

**E- NOTES**  
**on**  
**PROGRAMMING IN “C”**

## *Lecture Note: 1*

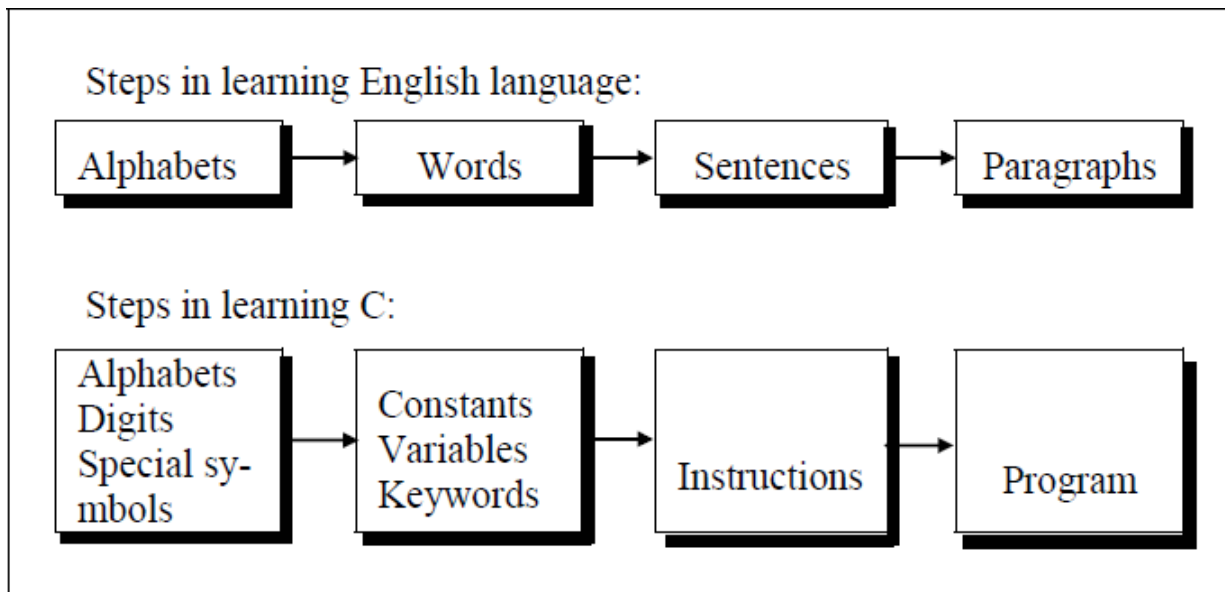
### Introduction to C

C is a programming language developed at AT & T's Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL, etc

ANSI C standard emerged in the early 1980s, this book was split into two titles: The original was still called *Programming in C*, and the title that covered ANSI C was called *Programming in ANSI C*. This was done because it took several years for the compiler vendors to release their ANSI C compilers and for them to become ubiquitous. It was initially designed for programming UNIX operating system. Now the software tool as well as the C compiler is written in C. Major parts of popular operating systems like Windows, UNIX, Linux is still written in C. This is because even today when it comes to performance (speed of execution) nothing beats C. Moreover, if one is to extend the operating system to work with new devices one needs to write device driver programs. These programs are exclusively written in C. C seems so popular is because it is **reliable**, **simple** and **easy** to use. often heard today is – “C has been already superceded by languages like C++, C# and Java.

## Program

There is a close analogy between learning English language and learning C language. The classical method of learning English is to first learn the alphabets used in the language, then learn to combine these alphabets to form words, which in turn are combined to form sentences and sentences are combined to form paragraphs. Learning C is similar and easier. Instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how using them constants, variables and keywords are constructed, and finally how are these combined to form an **instruction**. A group of instructions would be combined later on to form a **program**. So



a computer **program** is just a collection of the instructions necessary to solve a specific problem. The basic operations of a computer system form what is known as the computer's **instruction set**. And the approach or method that is used to solve the problem is known as an **algorithm**.

So far as programming language concern these are of two types.

- 1) Low level language
- 2) High level language

**Low level language:**

Low level languages are **machine level** and **assembly level language**. In machine level language computer only understand digital numbers i.e. in the form of 0 and 1. So, instruction given to the computer is in the form binary digit, which is difficult to implement instruction in binary code. This type of program is not portable, difficult to maintain and also error prone. The **assembly language** is on other hand modified version of machine level language. Where instructions are given in English like word as ADD, SUM, MOV etc. It is easy to write and understand but not understand by the machine. So the translator used here is assembler to translate into machine level. Although language is bit easier, programmer has to know low level details related to low level language. In the assembly level language the data are stored in the computer register, which varies for different computer. Hence it is not portable.

### **High level language:**

These languages are machine independent, means it is portable. The language in this category is Pascal, Cobol, Fortran etc. High level languages are understood by the machine. So it need to translate by the translator into machine level. A translator is software which is used to translate high level language as well as low level language in to machine level language.

Three types of translator are there:

### **Compiler**

### **Interpreter**

### **Assembler**

Compiler and interpreter are used to convert the high level language into machine level language. The program written in high level language is known as source program and the corresponding machine level language program is called as object program. Both compiler and interpreter perform the same task but there working is different. Compiler read the program at-a-time and searches the error and lists them. If the program is error free then it is converted into object program. When program size is large then compiler is preferred. Whereas interpreter read only one line of the source code and convert it to object code. If it check error, statement by statement and hence of take more time.

## **Integrated Development Environments (IDE)**

The process of editing, compiling, running, and debugging programs is often managed by a single integrated application known as an Integrated Development Environment, or IDE for short. An IDE is a windows-based program that allows us to easily manage large software programs, edit files in windows, and compile, link, run, and debug programs.

On Mac OS X, CodeWarrior and Xcode are two IDEs that are used by many programmers. Under Windows, Microsoft Visual Studio is a good example of a popular IDE. Kylix is a popular IDE for developing applications under Linux. Most IDEs also support program development in several different programming languages in addition to C, such as C# and C++.

**Structure of C Language program**

- 1 ) Comment line
- 2) Preprocessor directive
- 3 ) Global variable declaration
- 4) main function( )

```
    {  
        Local variables;  
  
        Statements;  
  
    }  
  
    User defined function  
  
    }  
  
}
```

**Comment line**

It indicates the purpose of the program. It is represented as

```
/*.....*/
```

Comment line is used for increasing the readability of the program. It is useful in explaining the program and generally used for documentation. It is enclosed within the decimeters. Comment line can be single or multiple line but should not be nested. It can be anywhere in the program except inside string constant & character constant.

**Preprocessor Directive:**

`#include<stdio.h>` tells the compiler to include information about the standard input/output library. It is also used in symbolic constant such as `#define PI 3.14(value)`. The `stdio.h` (standard input output header file) contains definition & declaration of system defined function such as `printf( )`, `scanf( )`, `pow( )` etc. Generally `printf()` function used to display and `scanf()` function used to read value

### **Global Declaration:**

This is the section where variable are declared globally so that it can be access by all the functions used in the program. And it is generally declared outside the function :

### **main()**

It is the user defined function and every function has one `main()` function from where actually program is started and it is enclosed within the pair of curly braces.

The `main( )` function can be anywhere in the program but in general practice it is placed in the first position.

Syntax :

```
main()  
{  
.....  
.....  
.....  
}
```

The `main( )` function return value when it declared by data type as

```
int main()  
{  
return 0
```

```
}
```

The main function does not return any value when void (means null/empty) as

```
void main(void ) or void main()
```

```
{
```

```
printf ("C language");
```

```
}
```

Output: C language

The program execution start with opening braces and end with closing brace.

And in between the two braces declaration part as well as executable part is mentioned. And at the end of each line, the semi-colon is given which indicates statement termination.

```
/*First c program with return statement*/
```

```
#include <stdio.h>
```

```
int main (void)
```

```
{
```

```
printf ("welcome to c Programming language.\n");
```

```
return 0;
```

```
}
```

Output: welcome to c programming language.

## **Steps for Compiling and executing the Programs**

A compiler is a software program that analyzes a program developed in a particular computer language and then translates it into a form that is suitable for execution



on a particular computer system. Figure below shows the steps that are involved in entering, compiling, and executing a computer program developed in the C programming language

**Step 1:** The program that is to be compiled is first typed into a *file* on the in Turbo C installed in computer system. There are various conventions that are used for naming files, typically be any name provided the last two characters are “.c” or file with extension .c. So, the file name **prog1.c** might be a valid filename for a C program. A text editor is usually used to enter the C program into a file. The program that is entered into the file is known as the *source program* because it represents the original form of the program expressed in the C language.

**Step 2:** After the source program has been entered into a file, then proceed to have it compiled. The compilation process is initiated by using keys i.e Alt+F9.

In the first step of the compilation process,

the compiler examines each program statement contained in the source program and checks it to ensure that it conforms to the syntax and semantics of the language. If any mistakes are discovered by the compiler during this phase, they are reported to the user and the compilation process ends right there. The errors then have to be corrected in the source program (with the use of an editor), and the compilation process must be restarted. Typical errors reported during this phase of compilation might be due to an expression that has unbalanced parentheses (**syntactic error**), or due to the use of a variable that is not “defined” (**semantic error**).

**Step 3:** When all the syntactic and semantic errors have been removed from the program, the compiler then proceeds to take each statement of the program and translate it into a “lower” form that is equivalent to assembly language program needed to perform the identical task.

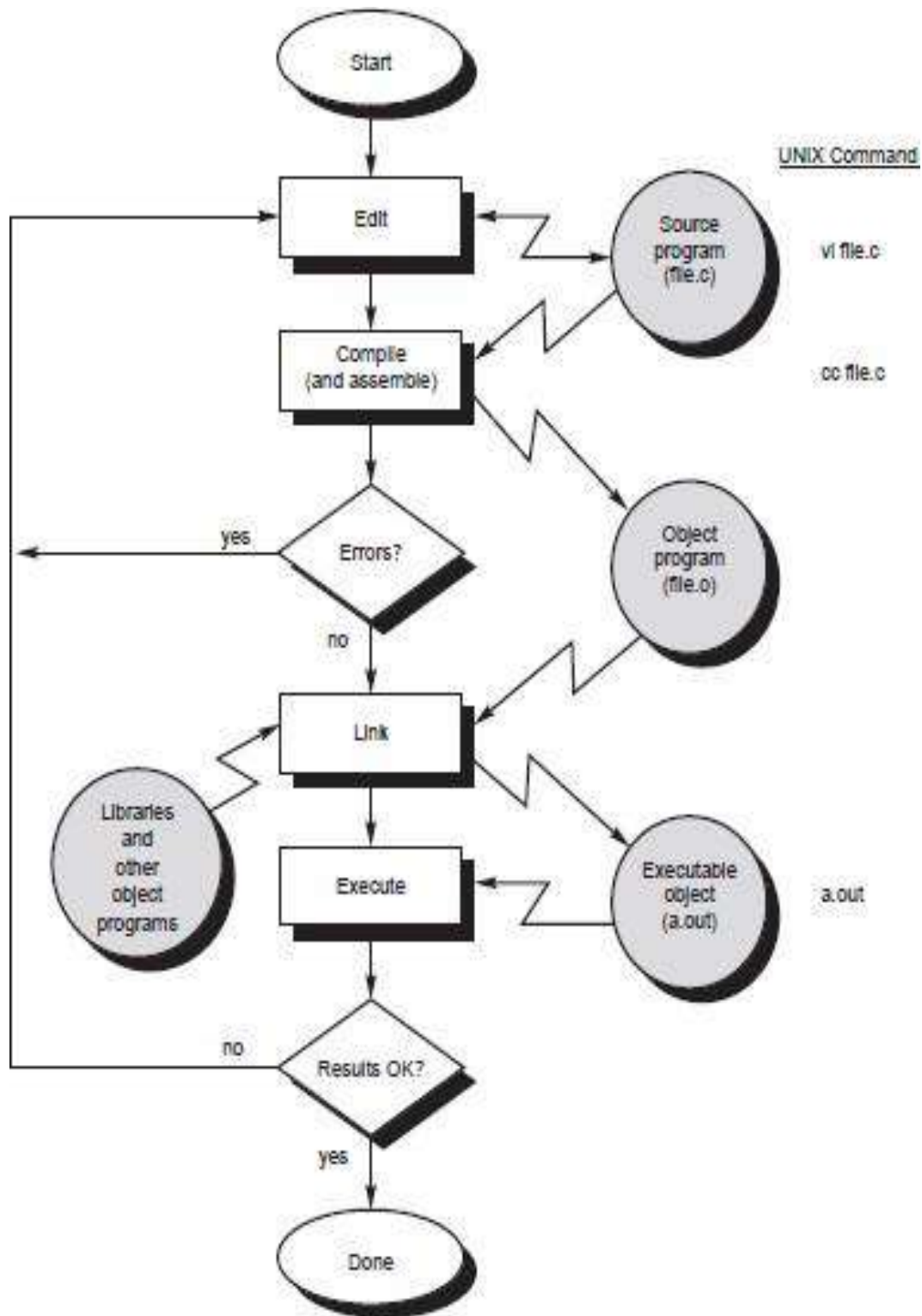
**Step 4:** After the program has been translated the next step in the compilation process is to translate the assembly language statements into actual machine instructions. The assembler takes each assembly language statement and converts it into a binary format known as *object code*, which is then written into another file on the system. This file has the same name as the source file under Unix, with the last letter an “o” (for *object*) instead of a “c”.

**Step 5:** After the program has been translated into object code, it is ready to be *linked*. The purpose of the linking phase is to get the program into a final form for execution on the computer. If the program uses other programs that were previously processed by the compiler, then during this phase the programs are linked together. Programs that are used from the system’s program *library* are also searched and linked together with the object program during this phase.

The process of compiling and linking a program is often called *building*. The final linked file, which is in an executable *object* code format, is stored in another file on the system, ready to be run or *executed*. Under Unix, this file is called **a.out** by default. Under Windows, the executable file usually has the same name as the source file, with the c extension replaced by an exe extension.

**Step 6:** To subsequently execute the program, the Keys Ctrl+F9 or Run from menu of Turbo C is used, initiating its execution.

When the program is executed, each of the statements of the program is sequentially executed in turn. If the program requests any data from the user, known as *input*, the program temporarily suspends its execution so that the input can be entered. Or, the program might simply wait for an *event*, such as a mouse being clicked, to occur. Results that are displayed by the program, known as *output*, appear in a window, sometimes called the *console*. If the program does not produce the desired results, it is necessary to go back and reanalyze the program's logic. This is known as the *debugging phase*, during which an attempt is made to remove all the known problems or *bugs* from the program. To do this, it will most likely be necessary to make changes to original source program.



/\* Simple program to add two numbers.....\*/

```
#include <stdio.h>

int main (void)
{
int v1, v2, sum;           //v1,v2,sum are variables and int is data type declared
v1 = 150;
v2 = 25;
sum = v1 + v2;
printf ("The sum of %i and %i is= %i\n", v1, v2, sum);
return 0;
}
```

Output:

The sum of 150 and 25 is=175

### **Lecture Note: 3**

#### **Character set**

A character denotes any alphabet, digit or special symbol used to represent information. Valid alphabets, numbers and special symbols allowed in C are

Alphabets	A, B, ....., Y, Z a, b, ....., y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ' ! @ # % ^ & * ( ) _ - + =   \ { } [ ] : ; " ' < > , . ? /

The alphabets, numbers and special symbols when properly combined form constants, variables and keywords.

## Identifiers

Identifiers are user defined word used to name of entities like variables, arrays, functions, structures etc. Rules for naming identifiers are:

- 1) name should only consists of alphabets (both upper and lower case), digits and underscore (\_) sign.
- 2) first characters should be alphabet or underscore
- 3) name should not be a keyword
- 4) since C is a case sensitive, the upper case and lower case considered differently, for example code, Code, CODE etc. are different identifiers.
- 5) identifiers are generally given in some meaningful name such as value, net\_salary, age, data etc. An identifier name may be long, some implementation recognizes only first eight characters, most recognize 31 characters. ANSI standard compiler recognize 31 characters. Some invalid identifiers are 5cb, int, res#, avg no etc.

## Keyword

There are certain words reserved for doing specific task, these words are known as **reserved word** or **keywords**. These words are predefined and always written in lower case or small letter. These keywords can't be used as a variable name as it assigned with fixed meaning. Some examples are **int, short, signed, unsigned, default, volatile, float, long, double, break, continue, typedef, static, do, for, union, return, while, do, extern, register, enum, case, goto, struct, char, auto, const** etc.

## Data types

Data types refer to an extensive system used for declaring variables or functions of different types before its use. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted. The value of a variable can be changed any time.

C has the following 4 types of data types

**basic built-in data types:** int, float, double, char

**Enumeration data type:** enum

**Derived data type:** pointer, array, structure, union

**Void data type:** void

A variable declared to be of type int can be used to contain integral values only—that is, values that do not contain decimal places. A variable declared to be of type float can be used for storing floating- point numbers (values containing decimal places). The double type is the same as type float, only with roughly twice the precision. The char data type can be used to store a single character, such as the letter *a*, the digit character *6*, or a semicolon similarly A variable declared char can only store character type value.

There are two types of type qualifier in c

**Size qualifier:** short, long

**Sign qualifier:** signed, unsigned

When the qualifier unsigned is used the number is always positive, and when signed is used number may be positive or negative. If the sign qualifier is not mentioned, then by default sign qualifier is assumed. The range of values for signed data types is less than that of unsigned data type. Because in signed type, the left most bit is used to represent sign, while in unsigned type this bit is also used to represent the value. The size and range of the different data types on a 16 bit machine is given below:

Basic data type	Data type with type qualifier	Size (byte)	Range
char	char or signed char	1	-128 to 127
	Unsigned char	1	0 to 255
int	int or signed int	2	-32768 to 32767
	unsigned int	2	0 to 65535
	short int or signed short int	1	-128 to 127
	unsigned short int	1	0 to 255
	long int or signed long int	4	-2147483648 to 2147483647
	unsigned long int	4	0 to 4294967295
float	float	4	-3.4E-38 to 3.4E+38
double	double	8	1.7E-308 to 1.7E+308
	Long double	10	3.4E-4932 to 1.1E+4932

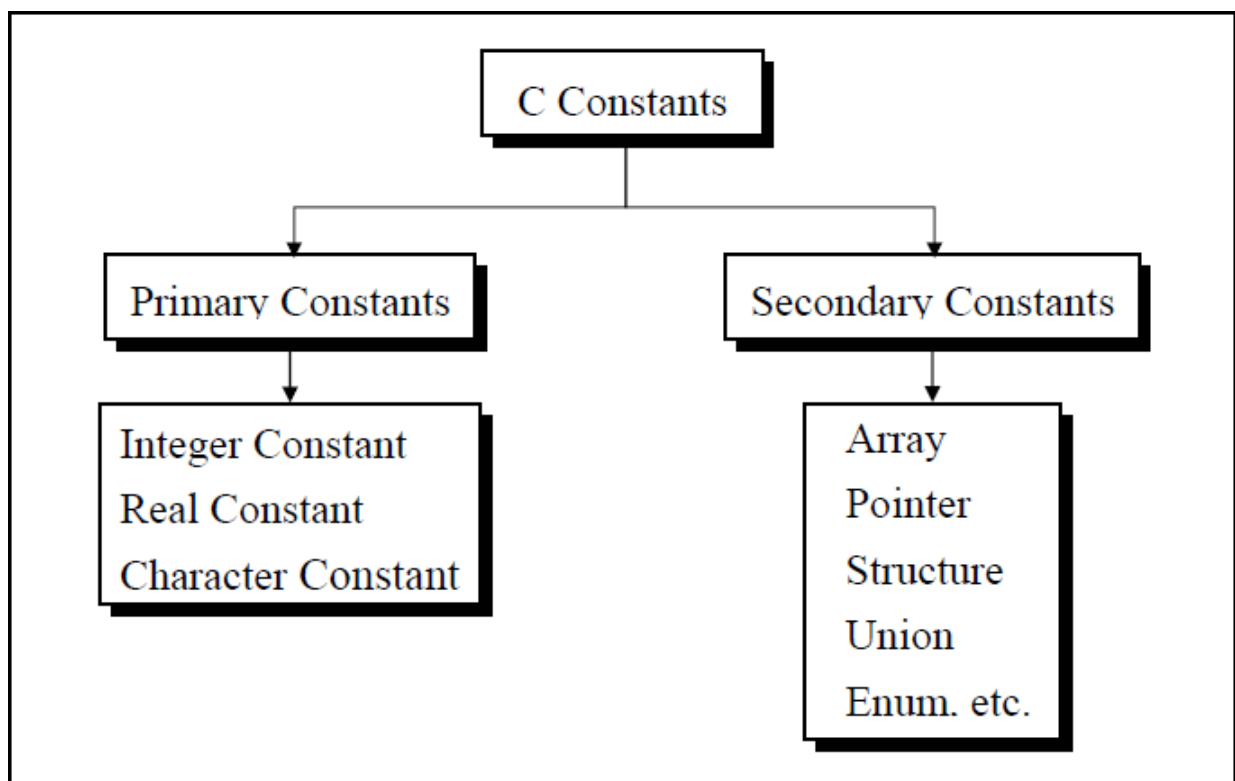


## Constants

Constant is a any value that cannot be changed during program execution. In C, any number, single character, or character string is known as a *constant*. A constant is an entity that doesn't change whereas a variable is an entity that may change. For example, the number 50 represents a constant integer value. The character string "Programming in C is fun.\n" is an example of a constant character string. C constants can be divided into two major categories:

Primary Constants  
Secondary Constants

These constants are further categorized as



**Numeric constant**  
**Character constant**  
**String constant**

**Numeric constant:** Numeric constant consists of digits. It required minimum size of 2 bytes and max 4 bytes. It may be positive or negative but by default sign is always positive. No comma or space is allowed within the numeric constant and it must have at least 1 digit. The allowable range for integer constants is -32768 to 32767. Truly speaking the range of an Integer constant depends upon the compiler. For a 16-bit compiler like Turbo C or Turbo C++ the range is -32768 to 32767. For a 32-bit compiler the range would be even greater. Mean by a 16-bit or a 32-bit compiler, what range of an Integer constant has to do with the type of compiler.

It is categorized a **integer constant** and **real constant**. An integer constant are whole number which have no decimal point. Types of integer constants are:

Decimal constant: 0----- 9(base 10)  
Octal constant: 0----- 7(base 8)  
Hexa decimal constant: 0----9, A ----- F(base 16)

In decimal constant first digit should not be zero unlike octal constant first digit must be zero(as 076, 0127) and in hexadecimal constant first two digit should be 0x/ 0X (such as 0x24, 0x87A). By default type of integer constant is integer but if the value of integer constant is exceeds range then value represented by integer type is taken to be unsigned integer or long integer. It can also be explicitly mention integer and unsigned integer type by suffix l/L and u/U.

**Real constant** is also called floating point constant. To construct real constant we must follow the rule of ,

- real constant must have at least one digit.
- It must have a decimal point.
- It could be either positive or negative.
- Default sign is positive.
- No commas or blanks are allowed within a real constant. Ex.: +325.34  
426.0  
-32.76

To express small/large real constant exponent(scientific) form is used where number is written in mantissa and exponent form separated by e/E. Exponent can be positive or negative integer but mantissa can be real/integer type, for example  $3.6 \times 10^5 = 3.6e+5$ . By default type of floating point constant is double, it can also be explicitly defined it by suffix of f/F.

## Character constant

Character constant represented as a single character enclosed within a single quote. These can be single digit, single special symbol or white spaces such as '9', 'c', '\$', ' ' etc. Every character constant has a unique integer like value in machine's character code as if machine using ASCII (American standard code for information interchange). Some numeric value associated with each upper and lower case alphabets and decimal integers are as:

A ----- Z ASCII value (65-90)  
a ----- z ASCII value (97-122)  
0-----9 ASCII value (48-59)  
; ASCII value (59)

## **String constant**

Set of characters are called string and when sequence of characters are enclosed within a double quote (it may be combination of all kind of symbols) is a string constant. String constant has zero, one or more than one character and at the end of the string null character(\0) is automatically placed by compiler. Some examples are “,sarathina” , “908”, “3”,” ”, “A” etc. In C although same characters are enclosed within single and double quotes it represents different meaning such as “A” and ‘A’ are different because first one is string attached with null character at the end but second one is character constant with its corresponding ASCII value is 65.

## **Symbolic constant**

Symbolic constant is a name that substitute for a sequence of characters and, characters may be numeric, character or string constant. These constant are generally defined at the beginning of the program as

#define name value , here name generally written in  
upper case for example

```
#define MAX 10  
  
#define CH 'b'  
  
#define NAME "sony"
```

## Variables

Variable is a data name which is used to store some data value or symbolic names for storing program computations and results. The value of the variable can be change during the execution. The rule for naming the variables is same as the naming identifier. Before used in the program it must be declared. Declaration of variables specify its name, data types and range of the value that variables can store depends upon its data types.

Syntax:

```
int a;
```

```
char c;
```

```
float f;
```

Variable initialization

When we assign any initial value to variable during the declaration, is called initialization of variables. When variable is declared but contain undefined value then it is called garbage value. The variable is initialized with the assignment operator such as

```
Data type variable name=constant;
```

```
Example: int a=20;
```

```
Or int a;
```

```
    a=20;
```

statements

## Expressions

An expression is a combination of variables, constants, operators and function call. It can be arithmetic, logical and relational for example:-

```
int z= x+y // arithmetic expression
```

```
a>b //relational
```

```
a==b // logical
```

```
func(a, b) // function call
```

Expressions consisting entirely of constant values are called *constant expressions*.

So, the expression

```
121 + 17 - 110
```

is a constant expression because each of the terms of the expression is a constant value. But if i were declared to be an integer variable, the expression

```
180 + 2 - j
```

would not represent a constant expression.

## Operator

This is a symbol use to perform some operation on variables, operands or with the constant. Some operator required 2 operand to perform operation or Some required single operation.

Several operators are there those are, arithmetic operator, assignment, increment , decrement, logical, conditional, comma, size of , bitwise and others.

### 1.Arithmetic Operator

This operator used for numeric calculation. These are of either Unary arithmetic operator, Binary arithmetic operator. Where Unary arithmetic operator required

only one operand such as +, -, ++, --, !, tiled. And these operators are addition, subtraction, multiplication, division. Binary arithmetic operator on other hand required two operand and its operators are +(addition), -(subtraction), \*(multiplication), /(division), %(modulus). But modulus cannot applied with floating point operand as well as there are no exponent operator in c.

Unary (+) and Unary (-) is different from addition and subtraction.

When both the operand are integer then it is called integer arithmetic and the result is always integer. When both the operand are floating point then it is called floating arithmetic and when operand is of integer and floating point then it is called mix type or mixed mode arithmetic . And the result is in float type.

## **2. Assignment Operator**

A value can be stored in a variable with the use of assignment operator. The assignment operator(=) is used in assignment statement and assignment expression. Operand on the left hand side should be variable and the operand on the right hand side should be variable or constant or any expression. When variable on the left hand side is occur on the right hand side then we can avoid by writing the compound statement. For example,

```
int x= y;
```

```
int Sum=x+y+z;
```

## **3. Increment and Decrement**

The Unary operator ++, --, is used as increment and decrement which acts upon single operand. Increment operator increases the value of variable by one .Similarly decrement operator decrease the value of the variable by one. And these operator can only used with the variable, but can't use with expression and constant as ++6 or ++(x+y+z).

It again categories into prefix post fix . In the prefix the value of the variable is incremented 1<sup>st</sup>, then the new value is used, where as in postfix the operator is written after the operand(such as m++,m--).

#### EXAMPLE

```
let y=12;
```

```
z= ++y;
```

```
y= y+1;
```

```
z= y;
```

Similarly in the postfix increment and decrement operator is used in the operation . And then increment and decrement is perform.

#### EXAMPLE

```
let x= 5;
```

```
y= x++;
```

```
y=x;
```

```
x= x+1;
```

### 4.Relational Operator

It is use to compared value of two expressions depending on their relation. Expression that contain relational operator is called relational expression.

Here the value is assign according to true or false value.

a.(a>=b) || (b>20)

b.(b>a) && (e>b)

c. 0(b!=7)

### 5.Conditional Operator

It sometimes called as ternary operator. Since it required three expressions as operand and it is represented as (? , :).

#### SYNTAX

```
exp1 ? exp2 :exp3
```

Here exp1 is first evaluated. It is true then value return will be exp2 . If false then exp3.

#### EXAMPLE

```
void main()
{
int a=10, b=2
int s= (a>b) ? a:b;
printf(“value is:%d”);
}
```

Output:

Value is:10

## 6.Comma Operator

Comma operator is use to permit different expression to be appear in a situation where only one expression would be used. All the expression are separator by comma and are evaluated from left to right.

#### EXAMPLE

```
int i, j, k, l;
for(i=1,j=2;i<=5;j<=10;i++;j++)
```



## 7.Sizeof Operator

Size of operator is a Unary operator, which gives size of operand in terms of byte that occupied in the memory. An operand may be variable, constant or data type qualifier.

Generally it is used make portable program(program that can be run on different machine) . It determines the length of entities, arrays and structures when their size are not known to the programmer. It is also use to allocate size of memory dynamically during execution of the program.

### EXAMPLE

```
main( )  
{  
int sum;  
float f;  
printf( "%d%d" ,size of(f), size of (sum) );  
printf("%d%d", size of(235 L), size of(A));  
}
```

## **1. Bitwise Operator**

Bitwise operator permit programmer to access and manipulate of data at bit level. Various bitwise operator enlisted are

one's complement	(~)
bitwise AND	(&)
bitwise OR	( )
bitwise XOR	(^)
left shift	(<<)
right shift	(>>)

These operator can operate on integer and character value but not on float and double. In bitwise operator the function showbits( ) function is used to display the binary representation of any integer or character value.

In one's complement all 0 changes to 1 and all 1 changes to 0. In the bitwise OR its value would obtaining by 0 to 2 bits.

As the bitwise OR operator is used to set on a particular bit in a number. Bitwise AND the logical AND.

It operate on 2operands and operands are compared on bit by bit basic. And hence both the operands are of same type.

### **Logical or Boolean Operator**

Operator used with one or more operand and return either value zero (for false) or one (for true). The operand may be constant, variables or expressions. And the expression that combines two or more expressions is termed as logical expression. C has three logical operators :

Operator	Meaning
----------	---------

&&	AND
	OR
!	NOT

Where logical NOT is a unary operator and other two are binary operator. Logical AND gives result true if both the conditions are true, otherwise result is false. And logical OR gives result false if both the condition false, otherwise result is true.

### Precedence and associativity of operators

Operators	Description	Precedence level	Associativity
()	function call	1	left to right
[]	array subscript		
→	arrow operator		
.	dot operator		
-----			
+	unary plus	2	right to left
-	unary minus		
++	increment		
--	decrement		
!	logical not		
~	1's complement		
*	indirection		
&	address		
(data type)	type cast		
sizeof	size in byte		
-----			
*	multiplication	3	left to right
/	division		
%	modulus		
-----			
+	addition	4	left to right

-	subtraction		
<<	left shift	5	left to right
>>	right shift		
<=	less than equal to	6	left to right
>=	greater than equal to		
<	less than		
>	greater than		
==	equal to	7	left to right
!=	not equal to		
&	bitwise AND	8	left to right
^	bitwise XOR	9	left to right
	bitwise OR	10	left to right
&&	logical AND	11	
	logical OR	12	
?:	conditional operator	13	
=, *=, /=, %=	} assignment operator	14	right to left
&=, ^=, <<=			
>>=			
,	comma operator	15	

## Control Statement

Generally C program statement is executed in a order in which they appear in the program. But sometimes we use decision making condition for execution only a part of program, that is called control statement. Control statement defined how the control is transferred from one part to the other part of the program. There are several control statement like if...else, switch, while, do. while, for loop,break, continue, goto etc.

## Loops in C

Loop:-it is a block of statement that performs set of instructions. In loops

Repeating particular portion of the program either a specified number of time or until a particular no of condition is being satisfied.

There are three types of loops in c

### 1.While loop

### 2.do while loop

### 3.for loop

#### While loop

```
while(condition)
{
Statement 1;
Statement 2;
}
Or      while(test condition)
        Statement;
```

The test condition may be any expression .when we want to do something a fixed no of times but not known about the number of iteration, in a program then while loop is used.

Here first condition is checked if, it is true body of the loop is executed else, If condition is false control will be come out of loop.

Example:-

```
/* wap to print 5 times welcome to C */  
  
#include<stdio.h>  
  
void main()  
{  
int p=1;  
While(p<=5)  
{  
printf(“Welcome to C\n”);  
P=p+1;  
}  
}
```

Output: Welcome to C

Welcome to C

Welcome to C

Welcome to C

Welcome to C

So as long as condition remains true statements within the body of while loop will get executed repeatedly.

### **do while loop**

This (do while loop) statement is also used for looping. The body of this loop may contain single statement or block of statement. The syntax for writing this statement is:

Syntax:-

```
Do
{
Statement;
}
while(condition);
```

Example:-

```
#include<stdio.h>

void main()
{
int X=4;

do
{
Printf(“%d”,X);
X=X+1;
```

```
}while(X<=10);  
    Printf(“ ”);  
}
```

Output: 4 5 6 7 8 9 10

Here firstly statement inside body is executed then condition is checked. If the condition is true again body of loop is executed and this process continue until the condition becomes false. Unlike while loop semicolon is placed at the end of while.

There is minor difference between while and do while loop, while loop test the condition before executing any of the statement of loop. Whereas do while loop test condition after having executed the statement at least one within the loop.

If initial condition is false while loop would not executed it's statement on other hand do while loop executed it's statement at least once even If condition fails for first time. It means do while loop always executes at least once. **Notes:**

Do while loop used rarely when we want to execute a loop at least once.



## **for loop**

In a program, for loop is generally used when number of iteration are known in advance. The body of the loop can be single statement or multiple statements. Its syntax for writing is:

Syntax:-

```
for(exp1;exp2;exp3)
{
Statement;
}
```

Or

```
for(initialized counter; test counter; update counter)
{
Statement;
}
```

Here exp1 is an initialization expression, exp2 is test expression or condition and exp3 is an update expression. Expression 1 is executed only once when loop started and used to initialize the loop variables. Condition expression generally uses relational and logical operators. And updation part executed only when after

body of the loop is executed.

Example:-

```
void main()
{
int i;
for(i=1;i<10;i++)
{
Printf(“ %d ”, i);
}
}
```

Output:-1 2 3 4 5 6 7 8 9

### **Nesting of loop**

When a loop written inside the body of another loop then, it is known as nesting of loop. Any type of loop can be nested in any type such as while, do while, for. For example nesting of for loop can be represented as :

```
void main()
{
int i,j;
for(i=0;i<2;i++)
for(j=0;j<5; j++)
printf(“%d %d”, i, j);
}
```

Output: i=0

j=0 1 2 3 4

i=1

j=0 1 2 3 4

### **Break statement(break)**

Sometimes it becomes necessary to come out of the loop even before loop condition becomes false then break statement is used. Break statement is used inside loop and switch statements. It cause immediate exit from that loop in which it appears and it is generally written with condition. It is written with the keyword as **break**. When break statement is encountered loop is terminated and control is transferred to the statement, immediately after loop or situation where we want to jump out of the loop instantly without waiting to get back to conditional state.

When break is encountered inside any loop, control automatically passes to the first statement after the loop. This break statement is usually associated with **if** statement.

Example :

```
void main()
{
int j=0;
for(;j<6;j++)
if(j==4)
break;
}
```

Output:

0 1 2 3

## **Continue statement (key word continue)**

Continue statement is used for continuing next iteration of loop after skipping some statement of loop. When it encountered control automatically passes through the beginning of the loop. It is usually associated with the if statement. It is useful when we want to continue the program without executing any part of the program.

The difference between break and continue is, when the break encountered loop is terminated and it transfer to the next statement and when continue is encounter control come back to the beginning position.

In while and do while loop after continue statement control transfer to the test condition and then loop continue where as in, for loop after continue control transferred to the updating expression and condition is tested.

Example:-

```
void main()
{
int n;
for(n=2; n<=9; n++)
{
if(n==4)
continue;
printf(“%d”, n);
}
}
Printf(“out of loop”);
}
```

Output: 2 3 5 6 7 8 9 out of loop

## **If statement**

Statement execute set of command like when condition is true and its syntax is

If (condition)

Statement;

The statement is executed only when condition is true. If the if statement body is consists of several statement then better to use pair of curly braces. Here in case condition is false then compiler skip the line within the if block.

```
void main()
{
    int n;
    printf (" enter a number:");
    scanf("%d",&n);
    If (n>10)
    Printf(" number is grater");
}
```

Output:

Enter a number:12

Number is greater

## **if.....else ... Statement**

it is bidirectional conditional control statement that contains one condition & two possible action. Condition may be true or false, where non-zero value regarded as true & zero value regarded as false. If condition are satisfy true, then a single or block of statement executed otherwise another single or block of statement is executed.

Its syntax is:-

```
if (condition)
{
Statement1;
Statement2;
}
else
{
Statement1;
Statement2;
}
```

Else statement cannot be used without if or no multiple else statement are allowed within one if statement. It means there must be a if statement with in an else statement.

Example:-

```
/* To check a number is eve or odd */
```

```
void main()
{
    int n;
    printf ("enter a number:");
    scanf ("%d", &n);
    If (n%2==0)
        printf ("even number");
    else
        printf("odd number");
}
```

Output: enter a number:121

odd number

## **Lecture Note: 10**

### **Nesting of if ...else**

When there are another if else statement in if-block or else-block, then it is called nesting of if-else statement.

Syntax is :-

```
if (condition)
{
    If (condition)
    Statement1;
else
    statement2;
}
Statement3;
```

**If....else LADDER**



In this type of nesting there is an if else statement in every else part except the last part. If condition is false control pass to block where condition is again checked with its if statement.

Syntax is :-

```
    if (condition)
        Statement1;
    else if (condition)
        statement2;
    else if (condition)
        statement3;
    else
        statement4;
```

This process continue until there is no if statement in the last block. if one of the condition is satisfy the condition other nested “else if” would not executed. But it has disadvantage over if else statement that, in if else statement whenever the condition is true, other condition are not checked. While in this case, all condition are checked.

## **ARRAY**

Array is the collection of similar data types or collection of similar entity stored in contiguous memory location. Array of character is a string. Each data item of an array is called an element. And each element is unique and located in separated memory location. Each of elements of an array share a variable but each element having different index no. known as subscript.

An array can be a single dimensional or multi-dimensional and number of subscripts determines its dimension. And number of subscript is always starts with zero. One dimensional array is known as vector and two dimensional arrays are known as matrix.

**ADVANTAGES:** array variable can store more than one value at a time where other variable can store one value at a time.

Example:

```
int arr[100];
```

```
int mark[100];
```

### **DECLARATION OF AN ARRAY :**

Its syntax is :

Data type array name [size];

```
int arr[100];
```

```
int mark[100];
```

```
int a[5]={10,20,30,100,5}
```

The declaration of an array tells the compiler that, the data type, name of the array, size of the array and for each element it occupies memory space. Like for int data type, it occupies 2 bytes for each element and for float it occupies 4 byte for each element etc. The size of the array operates the number of elements that can be

stored in an array and it may be a int constant or constant int expression.

We can represent individual array as :

```
int ar[5];  
  
ar[0], ar[1], ar[2], ar[3], ar[4];
```

Symbolic constant can also be used to specify the size of the array as:

```
#define SIZE 10;
```

### **INITIALIZATION OF AN ARRAY:**

After declaration element of local array has garbage value. If it is global or static array then it will be automatically initialize with zero. An explicitly it can be initialize that

```
Data type array name [size] = {value1, value2, value3...}
```

Example:

```
in ar[5]={20,60,90, 100,120}
```

Array subscript always start from zero which is known as lower bound and upper value is known as upper bound and the last subscript value is one less than the size of array. Subscript can be an expression i.e. integer value. It can be any integer, integer constant, integer variable, integer expression or return value from functional call that yield integer value.

So if i & j are not variable then the valid subscript are

```
ar [i*7],ar[i*i],ar[i++],ar[3];
```

The array elements are standing in continuous memory locations and the amount of storage required for hold the element depend in its size & type.

### **Total size in byte for 1D array is:**

```
Total bytes=size of (data type) * size of array.
```

Example : if an array declared is:

```
int [20];
```

Total byte= 2 \* 20 =40 byte.

### **ACCESSING OF ARRAY ELEMENT:**

/\*Write a program to input values into an array and display them\*/

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
int arr[5],i;
```

```
for(i=0;i<5;i++)
```

```
{
```

```
printf("enter a value for arr[%d] \n",i);
```

```
scanf("%d",&arr[i]);
```

```
}
```

```
printf("the array elements are: \n");
```

```
for (i=0;i<5;i++)
```

```
{
```

```
printf("%d\t",arr[i]);
```

```
}
```

```
return 0;
```

```
}
```

### **OUTPUT:**

Enter a value for arr[0] = 12

Enter a value for arr[1] =45

Enter a value for arr[2] =59

Enter a value for arr[3] =98

Enter a value for arr[4] =21

The array elements are 12 45 59 98 21

Example: From the above example value stored in an array are and occupy its memory addresses 2000, 2002, 2004, 2006, 2008 respectively.

a[0]=12, a[1]=45, a[2]=59, a[3]=98, a[4]=21

ar[0]	ar[1]	ar[2]	ar[3]	ar[4]
12	45	59	98	21
2000	2002	2004	2006	2008

Example 2:

```
/* Write a program to add 10 array elements */  
  
#include<stdio.h>  
  
void main()  
{  
    int i ;  
    int arr [10];  
    int sum=0;  
    for (i=0; i<=9; i++)  
    {  
        printf ("enter the %d element \n", i+1);  
        scanf ("%d", &arr[i]);
```

```
}  
for (i=0; i<=9; i++)  
{  
sum = sum + a[i];  
}  
printf (“the sum of 10 array elements is %d”, sum);  
}
```

OUTPUT:

```
Enter a value for arr[0] =5  
Enter a value for arr[1] =10  
Enter a value for arr[2] =15  
Enter a value for arr[3] =20  
Enter a value for arr[4] =25  
Enter a value for arr[5] =30  
Enter a value for arr[6] =35  
Enter a value for arr[7] =40  
Enter a value for arr[8] =45  
Enter a value for arr[9] =50  
  
Sum = 275
```

while initializing a single dimensional array, it is optional to specify the size of array. If the size is omitted during initialization then the compiler assumes the size of array equal to the number of initializers.

For example:-

```
int marks[]={99,78,50,45,67,89};
```

If during the initialization of the number the initializers is less than size of array, then all the remaining elements of array are assigned value zero .

For example:-

```
int marks[5]={99,78};
```

Here the size of the array is 5 while there are only two initializers so After this initialization, the value of the rest elements are automatically occupied by zeros such as

Marks[0]=99 , Marks[1]=78 , Marks[2]=0, Marks[3]=0, Marks[4]=0

Again if we initialize an array like

```
int array[100]={0};
```

Then the all the element of the array will be initialized to zero. If the number of initializers is more than the size given in brackets then the compiler will show an error.

For example:-

```
int arr[5]={1,2,3,4,5,6,7,8};//error
```

we cannot copy all the elements of an array to another array by simply assigning it to the other array like, by initializing or declaring as

```
int a[5] = {1,2,3,4,5};
```

```
int b[5];
```

```
b=a;//not valid
```

(**note**:-here we will have to copy all the elements of array one by one, using for loop.)

## Single dimensional arrays and functions

```
/*program to pass array elements to a function*/
```

```
#include<stdio.h>

void main()
{
int arr[10],i;
printf("enter the array elements\n");
for(i=0;i<10;i++)
{
scanf("%d",&arr[i]);
check(arr[i]);
}
}

void check(int num)
{
if(num%2=0)
{
printf("%d is even \n",num);
}
else
{
printf("%d is odd \n",num);
}
}
```



## Two dimensional arrays

Two dimensional array is known as matrix. The array declaration in both the array i.e. in single dimensional array single subscript is used and in two dimensional array two subscripts are used.

Its syntax is

Data-type array name[row][column];

Or we can say 2-d array is a collection of 1-D array placed one below the other.

Total no. of elements in 2-D array is calculated as **row\*column**

Example:-

```
int a[2][3];
```

Total no of elements=row\*column is  $2*3 = 6$

It means the matrix consist of 2 rows and 3 columns

For example:-

```
20  2   7
8   3  15
```

Positions of 2-D array elements in an array are as below

```
00  01  02
```

```
10  11  12
```

```
a [0][0]    a [0][0]    a [0][0]    a [0][0]    a [0][0]    a [0][0]
```

20	2	7	8	3	15
2000	2002	2004	2006		2008

### Accessing 2-d array /processing 2-d arrays

For processing 2-d array, we use two nested for loops. The outer for loop corresponds to the row and the inner for loop corresponds to the column.

For example

```
int a[4][5];
```

**for reading value:-**

```
for(i=0;i<4;i++)
{
    for(j=0;j<5;j++)
    {
        scanf("%d",&a[i][j]);
    }
}
```

**For displaying value:-**

```
for(i=0;i<4;i++)
{
    for(j=0;j<5;j++)
    {
        printf("%d",a[i][j]);
    }
}
```

}

### Initialization of 2-d array:

2-D array can be initialized in a way similar to that of 1-D array. for example:-

```
int mat[4][3]={11,12,13,14,15,16,17,18,19,20,21,22};
```

These values are assigned to the elements row wise, so the values of elements after this initialization are

Mat[0][0]=11,	Mat[1][0]=14,	Mat[2][0]=17	Mat[3][0]=20
Mat[0][1]=12,	Mat[1][1]=15,	Mat[2][1]=18	Mat[3][1]=21
Mat[0][2]=13,	Mat[1][2]=16,	Mat[2][2]=19	Mat[3][2]=22

While initializing we can group the elements row wise using inner braces.

for example:-

```
int mat[4][3]={{11,12,13},{14,15,16},{17,18,19},{20,21,22}};
```

And while initializing , it is necessary to mention the 2<sup>nd</sup> dimension where 1<sup>st</sup> dimension is optional.

```
int mat[][3];
```

```
int mat[2][3];
```

```
int mat[][];
int mat[2][]; } invalid
```

If we **initialize an array** as

```
int mat[4][3]={{11},{12,13},{14,15,16},{17}};
```

Then the compiler will assume its all rest value as 0, which are not defined.

Mat[0][0]=11,    Mat[1][0]=12,    Mat[2][0]=14,    Mat[3][0]=17  
 Mat[0][1]=0,    Mat[1][1]=13,    Mat[2][1]=15    Mat[3][1]=0  
 Mat[0][2]=0,    Mat[1][2]=0,    Mat[2][2]=16, Mat[3][2]=0

In memory map whether it is 1-D or 2-D, elements are stored in one contiguous manner.

We can also give the size of the 2-D array by using symbolic constant

Such as

```
#define ROW 2;

#define COLUMN 3;

int mat[ROW][COLUMN];
```

## String

Array of character is called a string. It is always terminated by the NULL character. String is a one dimensional array of character.

We can initialize the string as

```
char name[]={ 'j', 'o', 'h', 'n', '\0' };
```

Here each character occupies 1 byte of memory and last character is always NULL character. Where '\0' and 0 (zero) are not same, where **ASCII** value of '\0' is 0 and ASCII value of 0 is 48. Array elements of character array are also stored in contiguous memory allocation.

From the above we can represent as;

J	o	h	N	'\0'
---	---	---	---	------

The terminating NULL is important because it is only the way that the function that work with string can know, where string end.

String can also be **initialized** as;

```
char name[]="John";
```

Here the NULL character is not necessary and the compiler will assume it automatically.

### **String constant (string literal)**

A string constant is a set of character that enclosed within the double quotes and is also called a literal. Whenever a string constant is written anywhere in a program it is stored somewhere in a memory as an array of characters terminated by a NULL character ('\0').

Example – “m”

“Tajmahal”

“My age is %d and height is %f\n”

The string constant itself becomes a pointer to the first character in array.

Example-char crr[20]="Taj mahal";

1000	1001	1002	1003	1004	1005	1006	1007	100	1009
T	a	j		M	A	H	a	l	\0

It is called base address.

## String library function

There are several string library functions used to manipulate string and the prototypes for these functions are in header file “string.h”. Several string functions are

### **strlen()**

This function return the length of the string. i.e. the number of characters in the string excluding the terminating NULL character.

It accepts a single argument which is pointer to the first character of the string.

For example-

```
strlen(“suresh”);
```

It return the value 6.

### **In array version to calculate legnth:-**

```
int str(char str[])
{
    int i=0;
    while(str[i]!='\0')
    {
        i++;
    }
    return i;
}
```

Example:-

```
#include<stdio.h>
```

```
#include<string.h> void
```

```
main()
```

```
{
```

```
    char str[50];
```

```
    print("Enter a string:");
```

```
    gets(str);
```

```
    printf("Length of the string is %d\n",strlen(str));
```

```
}
```

Output:

Enter a string: C in Depth

Length of the string is 8

## **strcmp( )**

This function is used to compare two strings. If the two strings match, strcmp() returns a value 0 otherwise it returns a non-zero value. It compares the strings character by character and the comparison stops when the end of the string is reached or the corresponding characters in the two strings are not the same.

strcmp(s1,s2)

return a value:

<0 when s1<s2

=0 when s1=s2

>0 when s1>s2

The exact value returned in case of dissimilar strings is not defined. We only know that if s1<s2 then a negative value will be returned and if s1>s2 then a positive value will be returned.

For example



```

/*String comparison... ..... */
#include<stdio.h>
#include<string.h>
void main()
{
    char str1[10],str2[10];
    printf("Enter two strings:");
    gets(str1);
    gets(str2);
    if(strcmp(str1,str2)==0)
    {
        printf("String are same\n");
    }
    else
    {
        printf("String are not same\n")
    }
}

```

## **strcpy( )**

This function is used to copying one string to another string. The function strcpy(str1,str2) copies str2 to str1 including the NULL character. Here str2 is the source string and str1 is the destination string.

The old content of the destination string str1 are lost. The function returns a pointer to destination string str1.

Example:-

```
#include<stdio.h>

#include<string.h>

void main()
{

char str1[10],str2[10];

printf("Enter a string:");

scanf("%s",str2);

strcpy(str1,str2);

printf("First string:%s\t\tSecond string:%s\n",str1,str2);

strcpy(str,"Delhi");

strcpy(str2,"Bangalore");

printf("First string :%s\t\tSecond string:%s",str1,str2);
```

## **strcat( )**

This function is used to append a copy of a string at the end of the other string. If the first string is ""Purva" and second string is "Belmont" then after using this function the string becomes "PusvaBelmont". The NULL character from str1 is moved and str2 is added at the end of str1. The 2<sup>nd</sup> string str2 remains unaffected. A pointer to the first string str1 is returned by the function.

Example:-

```
#include<stdio.h>

#include<string.h>

void main()
{
    char str1[20],str[20];
    printf("Enter two strings:");
    gets(str1);
    gets(str2);
    strcat(str1,str2);
    printf("First string:%s\t second string:%s\n",str1,str2);
    strcat(str1,"-one");
    printf("Now first string is %s\n",str1);
}
```

Output

Enter two strings: data

Base

First string: database second string: database

Now first string is: database-one

## **Lecture Note: 14**

### **FUNCTION**

A function is a self contained block of codes or sub programs with a set of statements that perform some specific task or coherent task when it is called.

It is something like to hiring a person to do some specific task like, every six months servicing a bike and hand over to it.

Any 'C' program contain at least one function i.e main().

There are basically two types of function those are

#### **1. Library function**

#### **2. User defined function**

The user defined functions defined by the user according to its requirement

System defined function can't be modified, it can only read and can be used.

These function are supplied with every C compiler

Source of these library function are pre compiled and only object code get used by the user by linking to the code by linker

#### **Here in system defined function description:**

**Function definition** : predefined, precompiled, stored in the library

**Function declaration** : In header file with or function prototype.

**Function call** : By the programmer

### **User defined function**

Syntax:-

Return type      name of function (type 1 arg 1, type2 arg2, type3 arg3)

Return type      function name      argument list of the above syntax

So when user gets his own function three thing he has to know, these are.

### **Function declaration**

### **Function definition**

### **Function call**

These three things are represented like

```
int function(int, int, int);    /*function declaration*/

main()    /* calling function*/
{
    function(arg1,arg2,arg3);
}

int function(type 1 arg 1,type2 arg2,type3, arg3) /*function definition*/
{
    Local variable declaration;

    Statement;

    Return value;
}
```

### **Function declaration:-**

Function declaration is also known as function prototype. It inform the compiler about three thing, those are name of the function, number and type of argument received by the function and the type of value returned by the function.

While declaring the name of the argument is optional and the function prototype always terminated by the semicolon.

### **Function definition:-**

Function definition consists of the whole description and code of the function.

It tells about what function is doing what are its inputs and what are its out put

It consists of two parts function header and function body

Syntax:-

```
return type function(type 1 arg1, type2 arg2, type3 arg3) /*function header*/  
{  
    Local variable declaration;  
    Statement 1;  
    Statement 2;  
    Return value  
}
```

The return type denotes the type of the value that function will return and it is optional and if it is omitted, it is assumed to be int by default. The body of the function is the compound statements or block which consists of local variable declaration statement and optional return statement.

The local variable declared inside a function is local to that function only. It can't be used anywhere in the program and its existence is only within this function.

The arguments of the function **definition** are known as **formal arguments**.

## Function Call

When the function get called by the calling function then that is called, function call. The compiler execute these functions when the semicolon is followed by the function name.

Example:-

```
function(arg1,arg2,arg3);
```

The argument that are used inside the function call are called **actual argument**

Ex:-

```
int S=sum(a, b);           //actual arguments
```

## Actual argument

The arguments which are mentioned or used inside the function call is knows as actual argument and these are the original values and copy of these are actually sent to the called function

It can be written as constant, expression or any function call like

```
Function (x);
```

```
Function (20, 30);
```

```
Function (a*b, c*d);
```

```
Function(2,3,sum(a, b));
```

## Formal Arguments

The arguments which are mentioned in function definition are called formal arguments or dummy arguments.

These arguments are used to just hold the copied of the values that are sent by the calling function through the function call.

These arguments are like other local variables which are created when the function call starts and destroyed when the function ends.

The basic difference between the formal argument and the actual argument are

1) The formal argument are declared inside the parenthesis where as the local variable declared at the beginning of the function block.

2). The **formal argument** are automatically initialized when the copy of actual arguments are passed while other local variable are assigned values through the statements.

Order number and type of actual arguments in the function call should be match with the order number and type of the formal arguments.

### **Return type**

It is used to return value to the calling function. It can be used in two way as

```
return
```

```
Or return(expression);
```

```
Ex:- return (a);
```

```
return (a*b);
```

```
return (a*b+c);
```

Here the 1<sup>st</sup> return statement used to terminate the function without returning any value

```
Ex:- /*summation of two values*/
```

```
int sum (int a1, int a2);
```

```
main()
```



```

{
    int a,b;

    printf("enter two no");

    scanf("%d%d",&a,&b);

    int S=sum(a,b);

    printf("summation is = %d",s);

}

int sum(intx1,int y1)

{

int z=x1+y1;

Return z;

}

```

### **Advantage of function**

By using function large and difficult program can be divided in to sub programs and solved. When we want to perform some task repeatedly or some code is to be used more than once at different place in the program, then function avoids this repetition or rewritten over and over.

Due to reducing size, modular function it is easy to modify and test

### **Notes:-**

C program is a collection of one or more function.

A function is get called when function is followed by the semicolon.

A function is defined when a function name followed by a pair of curly braces

Any function can be called by another function even main() can be called by other function.

```
main()
{
function1()
}
function1()
{
Statement;
function2;
}
function 2()
{

}
```

So every function in a program must be called directly or indirectly by the main() function. A function can be called any number of times.

A function can call itself again and again and this process is called **recursion**.

A function can be called from other function **but** a function can't be defined in another function

### **Lecture Note: 15**

#### **Category of Function based on argument and return type**

##### **i) Function with no argument & no return value**

Function that have no argument and no return value is written as:- void

```
function(void);
```

```
    main()
    {
    void function()
    {
    Statement;
    }
```

Example:-

```
void me();
    main()
    {
    me();
    printf("in main");
    }
void me()
{
    printf("come on");
}
```

Output: come on

inn main

## ii) Function with no argument but return value

Syntax:-

```
int fun(void);

main()
{
    int r;
    r=fun();
}

int fun()
{
    reurn(exp);
}
```

Example:-

```
int sum();

main()
{
    int b=sum();
    printf("entered %d\n, b");
}

int sum()
{
    int a,b,s;
```

```
s=a+b;
return s;
}
```

Here called function is independent and are initialized. The values aren't passed by the calling function .Here the calling function and called function are communicated partly with each other.

### Lecture Note: 16

#### **iii ) function with argument but no return value**

Here the function have argument so the calling function send data to the called function but called function dose n't return value.

Syntax:-

```
void fun (int,int);
main()
{
int (a,b);
}
void fun(int x, int y);
{
Statement;
}
```

Here the result obtained by the called function.

#### iv) **function with argument and return value**

Here the calling function has the argument to pass to the called function and the called function returned value to the calling function.

Syntax:-

```
        fun(int,int);  
        main()  
    {  
        int r=fun(a,b);  
    }  
        int fun(intx,inty)  
    {  
        return(exp);  
    }
```

Example:

```
        main()  
    {  
        int fun(int);  
        int a,num;  
        printf("enter value:\n");  
        scanf("%d",&a)
```

```

int num=fun(a);

}

int fun(int x)
{
    ++x;

    return x;

}

```

## Call by value and call by reference

There are two way through which we can pass the arguments to the function such as **call by value** and **call by reference**.

### 1. Call by value

In the call by value copy of the actual argument is passed to the formal argument and the operation is done on formal argument.

When the function is called by ‘call by value’ method, it doesn’t affect content of the actual argument.

Changes made to formal argument are local to block of called function so when the control back to calling function the changes made is vanish.

Example:-

```

main()
{
    int x,y;

    change(int,int);
}

```

```

printf("enter two values:\n");
scanf("%d%d",&x,&y);
change(x ,y);
printf("value of x=%d and y=%d\n",x ,y);
}
change(int a,int b);
{
int k;
k=a;
a=b;
b=k;
}

```

Output: enter two values: 12

23

Value of x=12 and y=23

## 2. Call by reference

Instead of passing the value of variable, address or reference is passed and the function operate on address of the variable rather than value.

Here formal argument is alter to the actual argument, it means formal arguments calls the actual arguments.

Example:-

```
void main()
```



```

    {
        int a,b;
        change(int *,int*);
        printf("enter two values:\n");
        scanf("%d%d",&a,&b);
        change(&a,&b);
        printf("after changing two value of a=%d and b=%d\n:"a,b);
    }

    change(int *a, int *b)
    {
        int k;
        k=*a;
        *a=*b;
        *b= k;

        printf("value in this function a=%d and b=%d\n",*a,*b);
    }

```

Output: enter two values: 12

32

Value in this function a=32 and b=12

After changing two value of a=32 and b=12

So here instead of passing value of the variable, directly passing address of the variables. Formal argument directly access the value and swapping is possible even after calling a function.

## **Local, Global and Static variable**

### **Local variable: -**

variables that are defined within a body of function or block. The local variables can be used only in that function or block in which they are declared. Same variables may be used in different functions such as

```
function()
{
    int a,b;
    function 1();
}
function2 ()
{
    int a=0;
    b=20;
}
```

### **Global variable: -**

the variables that are defined outside of the function is called global variable. All functions in the program can access and modify global variables. Global variables are automatically initialized at the time of initialization.

Example:

```
#include<stdio.h>

void function(void);

void function1(void);

void function2(void);

int a, b=20;

void main()
{
    printf(“inside main a=%d,b=%d \n”,a,b);

function();
function1();
function2();
}

function()
{
    printf(“inside function a=%d,b=%d\n”,a,b);
}

function 1()
{
```

```

        printf("inside function a=%d,b=%d\n",a,b);
    }
function 2()
{
    printf("inside function a=%d,b=%d\n",a,);
}

```

**Static variables:**static variables are declared by writing the key word static.

-syntax:-

```
static data type variable name;
```

```
static int a;
```

-the static variables initialized only once and it retain between the function call. If its variable is not initialized, then it is automatically initialized to zero.

Example:

```
void fun1(void);
```

```
void fun2(void);
```

```
void main()
```

```
{
```

```
    fun1();
```

```
    fun2();
```

```
}
```

```
void fun1()
```

```
{
```

```

int a=10, static int b=2;

printf("a=%d, b=%d",a,b);

a++;

b++;

}

```

Output:a= 10 b= 2

a=10 b= 3

## Recursion

When function calls itself (inside function body) again and again then it is called as recursive function. In recursion calling function and called function are same. It is powerful technique of writing complicated algorithm in easiest way. According to recursion problem is defined in term of itself. Here statement with in body of the function calls the same function and same times it is called as circular definition. In other words recursion is the process of defining something in form of itself.

Syntax:

```

main ()
{
    rec(); /*function call*/
    rec();
    rec();
}

```

Ex:- /\*calculate factorial of a no.using recursion\*/

```

int fact(int);

void main()

```

```

    {
        int num;
        printf("enter a number");
        scanf("%d",&num);
        f=fact(num);
        printf("factorial is =%d\n",f);
    }
fact (int num)
{
    If (num==0||num==1)
return 1;
else
return(num*fact(num-1));
}

```

### **Lecture Note: 18**

#### **Monolithic Programming**

The program which contains a single function for the large program is called monolithic program. In monolithic program not divided the program, it is huge long pieces of code that jump back and forth doing all the tasks like single thread of execution, the program requires. Problem arise in monolithic program is that, when the program size increases it leads inconvenience and difficult to maintain

such as testing, debugging etc. Many disadvantages of monolithic programming are:

1. Difficult to check error on large programs size.
2. Difficult to maintain because of huge size.
3. Code can be specific to a particular problem. i.e. it cannot be reused.

Many early languages (FORTRAN, COBOL, BASIC, C) required one huge workspace with labelled areas that may does specific tasks but are not isolated.

## **Modular Programming**

The process of subdividing a computer program into separate sub-programs such as functions and subroutines is called Modular programming. **Modular programming sometimes also called as structured programming.** It enables multiple programmers to divide up the large program and debug pieces of program independently and tested.

. Then the linker will link all these modules to form the complete program. This principle dividing software up into parts, or modules, where a module can be changed, replaced, or removed, with minimal effect on the other software it works with. Segmenting the program into modules clearly defined functions, it can determine the source of program errors more easily. Breaking down program functions into modules, where each of which accomplishes one function and contains all the source code and variables needed to accomplish that function. Modular program is the solution to the problem of very large program that are difficult to debug, test and maintain. A program module may be rewritten while its inputs and outputs remain the same. The person making a change may only understand a small portion of the original program.

Object-oriented programming (OOP) is compatible with the modular programming concept to a large extent.

. , Less code has to be written that makes shorter.

- A single procedure can be developed for reuse, eliminating the need to retype the code many times.
- Programs can be designed more easily because a small team deals with only a small part of the entire code.
- Modular programming allows many programmers to collaborate on the same application.
- The code is stored across multiple files.
- Code is short, simple and easy to understand and modify, make simple to figure out how the program is operate and reduce likely hood of bugs.
- Errors can easily be identified, as they are localized to a subroutine or function or isolated to specific module.
- The same code can be reused in many applications.
- The scoping of variables and functions can easily be controlled.

#### Disadvantages

However it may takes longer to develop the program using this technique.

### Storage Classes

Storage class in c language is a specifier which tells the compiler where and how to store variables, its initial value and scope of the variables in a program. Or attributes of variable is known as storage class or in compiler point of view a variable identify some physical location within a computer where its string of bits value can be stored is known as storage class.

The kind of location in the computer, where value can be stored is either in the memory or in the register. There are various storage class which determined, in which of the two location value would be stored.

Syntax of declaring storage classes is:-

*storageclass datatype variable name;*

There are four types of storage classes and all are keywords:-

#### 1 ) Automatic (auto)



**2 ) Register (register)**

**3) Static (static)**

**4 ) External (extern)**

Examples:-

```
auto float x; or float x;
```

```
extern int x;
```

```
register char c;
```

```
static int y;
```

Compiler assume different storage class based on:-

**1 ) Storage class:-** tells us about storage place(where variable would be stored).

**2) Intial value :-**what would be the initial value of the variable.

If initial value not assigned, then what value taken by uninitialized variable.

**3) Scope of the variable:-**what would be the value of the variable of the program.

**4) Life time :-** It is the time between the creation and distribution of a variable or how long would variable exists.

### **1. Automatic storage class**

The keyword used to declare automatic storage class is auto.

Its features:-

**Storage-**memory location

**Default initial value:-**unpredictable value or garbage value.

**Scope:**-local to the block or function in which variable is defined.

**Life time:**-Till the control remains within function or block in which it is defined. It terminates when function is released.

The variable without any storage class specifier is called automatic variable.

Example:-

```
main( )  
{  
auto int i;  
printf(“i=”,i);  
}
```

## **Lecture Note: 19**

### **2. Register storage class**

The keyword used to declare this storage class is register.

The features are:-

**Storage:**-CPU register.

**Default initial value** :-garbage value

**Scope** :-local to the function or block in which it is defined.

**Life time** :-till controls remains within function or blocks in which it is defined.

Register variable don't have memory address so we can't apply address operator on it. CPU register generally of 16 bits or 2 bytes. So we can apply storage classes only for integers, characters, pointer type.

Variable stored in register storage class always access faster than, which is always stored in the memory. But to store all variable in the CPU register is not possible because of limitation of the register pair.

And when variable is used at many places like loop counter, then it is better to declare it as register class.

Example:-

```
main( )
{
register int i;
for(i=1;i<=12;i++)
printf(“%d”,i);
}
```

### **3 Static storage class**

The keyword used to declare static storage class is static.

Its feature are:-

**Storage:-**memory location

**Default initial value:-** zero

**Scope :-** local to the block or function in which it is defined.

**Life time:-** value of the variable persist or remain between different function call.

Example:-

```
main( )
```

```
{  
reduce();  
reduce();  
reduce ();  
}
```

```
reduce()
```

```
{  
static int x=10;  
printf(“%d”,x);  
x++;  
}
```

Output:-10,11,12

### **External storage classes**

The keyword used for this class is extern.

Features are:-

**Storage:-** memory area

**Default initial value:-**zero

**Scope :-** global

**Life time:-**as long as program execution remains it retains.

Declaration does not create variables, only it refer that already been created at somewhere else. So, memory is not allocated at a time of declaration and the external variables are declared at outside of all the function.

Example:-

```
int i,j;

void main( )

{
printf( "i=%d",i );
receive( );
receive ( );
reduce( );
reduce( );
}

receive( )

{
i=i+2;
printf("on increase i=%d",i);
}

reduce( )

{
i=i-1;
printf("on reduce i=%d",i);
}
```

Output:-i=0,2,4,3,2.

When there is large program i.e divided into several files, then external variable should be preferred. External variable extend the scope of variable.

## **Lecture Note: 20**

### **POINTER**

A pointer is a variable that store memory address or that contains address of another variable where addresses are the location number always contains whole number. So, pointer contain always the whole number. It is called pointer because it points to a particular location in memory by storing address of that location.

Syntax-

**Data type \*pointer name;**

Here \* before pointer indicate the compiler that variable declared as a pointer.

e.g.

```
int *p1; //pointer to integer type
```

```
float *p2; //pointer to float type
```

```
char *p3; //pointer to character type
```

When pointer declared, it contains garbage value i.e. it may point any value in the memory.

Two operators are used in the pointer i.e. **address operator(&)** and **indirection operator or dereference operator (\*)**.

Indirection operator gives the values stored at a particular address.

Address operator cannot be used in any constant or any expression.

Example:

```
void main()
{
    int i=105;
    int *p;
    p=&i;

    printf("value of i=%d",*p);
    printf("value of i=%d",*(&i));
    printf("address of i=%d",&i);
    printf("address of i=%d",p);
    printf("address of p=%u",&p);
}
```

## **Pointer Expression**

### **Pointer assignment**

```
int i=10;
```

```
int *p=&i;//value assigning to the pointer
```

Here declaration tells the compiler that P will be used to store the address of integer value or in other word P is a pointer to an integer and \*p reads the **value at the address contain in p.**

```
P++;
```

```
printf("value of p=%d");
```

We can assign value of 1 pointer variable to other when their base type and data type is same or both the pointer points to the same variable as in the array.

```
Int *p1,*p2;
```

```
P1=&a[1];
```

```
P2=&a[3];
```

We can assign constant 0 to a pointer of any type for that symbolic constant 'NULL' is used such as

```
*p=NULL;
```

It means pointer doesn't point to any valid memory location.

## **Pointer Arithmetic**

Pointer arithmetic is different from ordinary arithmetic and it is perform relative to the data type(base type of a pointer).

Example:-

If integer pointer contain address of 2000 on incrementing we get address of 2002 instead of 2001, because, size of the integer is of 2 bytes.

Note:-

When we move a pointer, somewhere else in memory by incrementing or decrement or adding or subtracting integer, it is not necessary that, pointer still pointer to a variable of same data, because, memory allocation to the variable are done by the compiler.



But in case of array it is possible, since there data are stored in a consecutive manner.

Ex:-

```
void main( )
{
static int a[ ]={20,30,105,82,97,72,66,102};
int *p,*p1;
P=&a[1];
P1=&a[6];
printf(“%d”,*p1-*p);
printf(“%d”,p1-p);
}
```

**Arithmetic operation never perform on pointer are:**

**addition, multiplication and division of two pointer.**

**multiplication between the pointer by any number.**

**division of pointer by any number**

**-add of float or double value to the pointer.**

Operation performed in pointer are:-

*/\* Addition of a number through pointer \*/*

Example

```
int i=100;
```

```
int *p;
```

```
p=&i;  
p=p+2;  
p=p+3;  
p=p+9;
```

ii /\* Subtraction of a number from a pointer' \*/

Ex:-

```
int i=22;  
*p1=&a;  
p1=p1-10;  
p1=p1-2;
```

iii- Subtraction of one pointer to another is possible when pointer variable point to an element of same type such as an array.

Ex:-

```
in tar[ ]={2,3,4,5,6,7};  
int *ptr1,*ptr1;  
ptr1=&a[3]; //2000+4  
ptr2=&a[6]; //2000+6
```

## Lecture Note: 21

### **Precedence of dereference (\*) Operator and increment operator and decrement operator**

The precedence level of dereference operator increment or decrement operator is same and their associativity from right to left.

Example :-

```
int x=25;
```

```
int *p=&x;
```

Let us calculate `int y=*p++;`

Equivalent to `*(p++)`

Since the operator associate from right to left, increment operator will applied to the pointer p.

i) `int y=*p++;` equivalent to `*(p++)`

`p =p++` or `p=p+1`

ii) `*++p;` → `*(++p)` → `p=p+1`

`y=*p`

iii) `int y=++*p`

equivalent to `++(*p)`

`p=p+1` then `*p`

iv) `y=(*p)++` → equivalent to `*p++`

`y=*p` then

`P=p+1 ;`

Since it is postfix increment the value of p.

## Pointer Comparison

Pointer variable can be compared when both variable, object of same data type and it is useful when both pointers variable points to element of same array.

Moreover pointer variable are compared with zero which is usually expressed as null, so several operators are used for comparison like the relational operator.

==, !=, <=, <, >, >=, can be used with pointer. Equal and not equal operators used to compare two pointer should finding whether they contain same address or not and they will equal only if are null or contains address of same variable.

Ex:-

```
void main()
{
static int arr[]={20,25,15,27,105,96}
int *x,*y;
x=&a[5];
y=&(a+5);
if(x==y)
printf("same");
else
printf("not");
}
```

## Lecture Note: 22

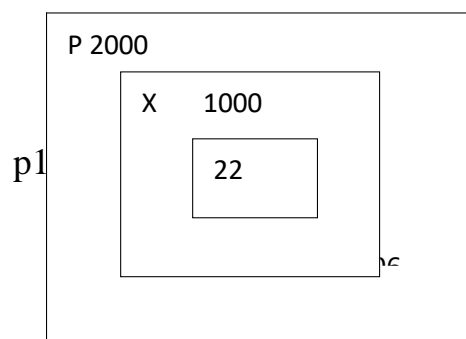
**Pointer to pointer:** Addition of pointer variable stored in some other variable is called pointer to pointer variable.

Or

Pointer within another pointer is called pointer to pointer.

Syntax:-

```
Data type **p;  
int x=22;  
int *p=&x;  
int **p1=&p;  
  
printf("value of x=%d",x);  
printf("value of x=%d",*p);  
printf("value of x=%d",&x);  
printf("value of x=%d",**p1);  
printf("value of p=%u",&p);  
printf("address of p=%u",p1);  
printf("address of x=%u",p);  
printf("address of p1=%u",&p1);  
printf("value of p=%u",p);  
printf("value of p=%u",&x);
```



3000

## Pointer vs array

Example :-

```
void main()
{
static char arr[]="Rama";
char*p="Rama";
printf("%s%s", arr, p);
```

In the above example, at the first time printf( ), print the same value array and pointer.

Here array arr, as **pointer to character** and **p act as a pointer to array of character** . When we are trying to increase the value of arr it would give the error because its known to compiler about an array and its base address which is always printed to base address is known as constant pointer and the base address of array which is not allowed by the compiler.

```
printf("size of (p)",size of (ar));
```

size of (p)            2/4 bytes

size of(ar)            5 bytes

## Structure

It is the collection of dissimilar data types or heterogenous data types grouped together. It means the data types may or may not be of same type.

Structure declaration-

```
struct tagname  
{  
Data type member1;  
Data type member2;  
Data type member3;  
.....  
.....  
Data type member n;  
};
```

OR

```
struct  
{  
Data type member1;  
Data type member2;
```

Data type member3;

.....

.....

Data type member n;

};

OR

struct tagname

{

struct element 1;

struct element 2;

struct element 3;

.....

.....

struct element n;

};

Structure variable declaration;

struct student

{

int age;

char name[20];

char branch[20];



```
}; struct student s;
```

### **Initialization of structure variable-**

Like primary variables structure variables can also be initialized when they are declared. Structure templates can be defined locally or globally. If it is local it can be used within that function. If it is global it can be used by all other functions of the program.

We cant initialize structure members while defining the structure

```
struct student
{
    int age=20;
    char name[20]="sona";
}s1;
```

The above is **invalid**.

A structure can be initialized as

```
struct student
{
    int age,roll;
    char name[20];
} struct student s1={16,101,"sona"};
    struct student s2={17,102,"rupa"};
```

If initialiser is less than no.of structure variable, automatically rest values are taken as zero.

## Accessing structure elements-

Dot operator is used to access the structure elements. Its associativity is from left to right.

```
structure variable ;
```

```
s1.name[];
```

```
s1.roll;
```

```
s1.age;
```

Elements of structure are stored in contiguous memory locations. Value of structure variable can be assigned to another structure variable of same type using assignment operator.

Example:

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main()
```

```
{
```

```
int roll, age;
```

```
char branch;
```

```
} s1,s2;
```

```
printf("\n enter roll, age, branch=");
```

```
scanf("%d %d %c", &s1.roll, &s1.age, &s1.branch);
```

```
s2.roll=s1.roll;
```

```
printf(" students details=\n");
```

```
printf("%d %d %c", s1.roll, s1.age, s1.branch);
```

```
printf("%d", s2.roll);
```

```
}
```

**Unary, relational, arithmetic, bitwise operators** are not allowed within structure variables.

## *Lecture Note:24*

### **Size of structure-**

Size of structure can be found out using sizeof() operator with structure variable name or tag name with keyword.

```
sizeof(struct student); or
```

```
sizeof(s1);
```

```
sizeof(s2);
```

Size of structure is different in different machines. So size of whole structure may not be equal to sum of size of its members.

### **Array of structures**

When database of any element is used in huge amount, we prefer Array of structures.

Example: suppose we want to maintain data base of 200 students, Array of structures is used.

```
#include<stdio.h>
```

```
#include<string.h>
```

```
struct student
```

```
{
```

```

char name[30];
char branch[25];
int roll;
};
void main()
{
struct student s[200];
int i;
s[i].roll=i+1;
printf("\nEnter information of students:");
for(i=0;i<200;i++)
{
printf("\nEnter the roll no:%d\n",s[i].roll);
printf("\nEnter the name:");
scanf("%s",s[i].name);
printf("\nEnter the branch:");
scanf("%s",s[i].branch);
printf("\n");
}
printf("\nDisplaying information of students:\n\n");
for(i=0;i<200;i++)
{
printf("\n\nInformation for roll no%d:\n",i+1);

```

```
printf("\nName:");  
puts(s[i].name);  
printf("\nBranch:");  
puts(s[i].branch);  
}  
}
```

In Array of structures each element of array is of structure type as in above example.

### **Array within structures**

```
struct student  
{  
char name[30];  
int roll,age,marks[5];  
}; struct student s[200];
```

We can also initialize using same syntax as in array.

### **Nested structure**

When a structure is within another structure, it is called Nested structure. A structure variable can be a member of another structure and it is represented as

```
struct student
```

```
{
element 1;
element 2;
.....
.....
struct student1
{
member 1;
member 2;
}variable 1;
.....
.....
element n;
}variable 2;
```

It is possible to define structure outside & declare its variable inside other structure.

```
struct date
{
int date,month;
};
struct student
{
```

```
char nm[20];  
int roll;  
struct date d;  
}; struct student s1;  
    struct student s2,s3;
```

Nested structure may also be initialized at the time of declaration like in above example.

```
struct student s={"name",200, {date, month}};  
                {"ram",201, {12,11}};
```

**Nesting of structure within itself** is not valid. Nesting of structure can be extended to any level.

```
struct time  
{  
int hr,min;  
};  
struct day  
{  
int date,month;  
struct time t1;  
};  
struct student
```

```
{  
char nm[20];  
struct day d;  
}stud1, stud2, stud3;
```

## Lecture Note: 25

### **Passing structure elements to function**

We can pass each element of the structure through function but passing individual element is difficult when number of structure element increases. To overcome this, we use to pass the whole structure through function instead of passing individual element.

```
#include<stdio.h>  
  
#include<string.h>  
  
void main()  
{  
struct student  
{  
char name[30];  
char branch[25];  
int roll;  
}struct student s;  
printf(“\n enter name=”);
```



```

gets(s.name);
printf("\nEnter roll:");
scanf("%d",&s.roll);
printf("\nEnter branch:");
gets(s.branch);
display(name,roll,branch);
}
display(char name, int roll, char branch)
{
printf("\n name=%s,\n roll=%d, \n branch=%s", s.name, s.roll. s.branch);
}

```

### **Passing entire structure to function**

```

#include<stdio.h>
#include<string.h>
struct student
{
char name[30];
int age,roll;
};
display(struct student); //passing entire structure
void main()

```

```

{
    struct student s1={"sona",16,101 };
    struct student s2={"rupa",17,102 };
display(s1);
display(s2);
}
display(struct student s)
{
printf("\n name=%s, \n age=%d ,\n roll=%d", s.name, s.age, s.roll);
}

```

Output: name=sona

roll=16

### **Lecture Note: 26**

## **UNION**

**Union** is derived data type contains collection of different data type or dissimilar elements. All definition declaration of union variable and accessing member is similar to structure, but instead of keyword struct the keyword union is used, the main difference between union and structure is

Each member of structure occupy the memory location, but in the unions members share memory. Union is used for saving memory and concept is useful when it is not necessary to use all members of union at a time.

Where union offers a memory treated as variable of one type on one occasion where (struct), it read number of different variables stored at different place of memory.

### **Syntax of union:**

```
union student
{
datatype member1;
datatype member2;
};
```

Like structure variable, union variable can be declared with definition or separately such as

```
union union name
{
Datatype member1;
}var1;
```

Example:- union student s;

Union members can also be accessed by the dot operator with union variable and if we have pointer to union then member can be accessed by using (arrow) operator as with structure.

Example:- struct student

```
struct student
```

```
{
```

```
int i;
```

```
char ch[10];
```

```
};struct student s;
```

Here datatype/member structure occupy 12 byte of location is memory, where as in the union side it occupy only 10 byte.

## Lecture Note:27

### **Nested of Union**

When one union is inside the another union it is called nested of union.

Example:-

```
union a
```

```
{
```

```
int i;
```

```
int age;
```

```
};
```

```
union b
```

```
{  
char name[10];  
union a aa;  
}; union b bb;
```

There can also be union inside structure or structure in union.

Example:-

```
void main()  
{  
    struct a  
    {  
int i;  
char ch[20];  
};  
    struct b  
    {  
int i;  
char d[10];  
};  
    union z  
    {  
struct a a1;  
struct b b1;
```

```
}; union z z1;  
z1.b1.j=20;  
z1.a1.i=10;  
z1.a1.ch[10]= " i";  
z1.b1.d[0]="j";  
printf(" ");
```

## Dynamic memory Allocation

The process of allocating memory at the time of execution or at the runtime, is called dynamic memory location.

Two types of problem may occur in static memory allocation.

If number of values to be stored is less than the size of memory, there would be wastage of memory.

If we would want to store more values by increase in size during the execution on assigned size then it fails.

Allocation and release of memory space can be done with the help of some library function called dynamic memory allocation function. These library function are called as **dynamic memory allocation function**. These library function prototype are found in the header file, "alloc.h" where it has defined.

Function take memory from memory area is called heap and release when not required.

Pointer has important role in the dynamic memory allocation to allocate memory.

**malloc():**

This function use to allocate memory during run time, its declaration is  
`void*malloc(size);`

### **malloc ()**

returns the pointer to the 1<sup>st</sup> byte and allocate memory, and its return type is void, which can be type cast such as:

```
int *p=(datatype*)malloc(size)
```

If memory location is successful, it returns the address of the memory chunk that was allocated and it returns null on unsuccessful and from the above declaration a pointer of type(**datatype**) and size in byte.

And **datatype** pointer used to typecast the pointer returned by malloc and this typecasting is necessary since, malloc() by default returns a pointer to void.

Example `int*p=(int*)malloc(10);`

So, from the above pointer p, allocated IO contiguous memory space address of 1<sup>st</sup> byte and is stored in the variable.

We can also use, the size of operator to specify the the size, such as  
`*p=(int*)malloc(5*size of int)` Here, 5 is the no. of data.

Moreover , it returns null, if no sufficient memory available , we should always check the malloc return such as, **if(p==null)**

```
printf(“not sufficient memory”);
```

Example:

```
/*calculate the average of mark*/
```

```
void main()
```

```
{
```

```
int n , avg,i,*p,sum=0;
```

```

printf("enter the no. of marks ");
scanf("%d",&n);
p=(int *)malloc(n*size(int));
if(p==null)
printf("not sufficient");
exit();
}
for(i=0;i<n;i++)
scanf("%d",(p+i));
for(i=0;i<n;i++)
Printf("%d",*(p+i));
sum=sum+*p;
avg=sum/n;
printf("avg=%d",avg);

```

### **Lecture Note: 28**

#### **calloc()**

Similar to malloc only difference is that calloc function use to allocate multiple block of memory .

two arguments are there

**1<sup>st</sup> argument specify number of blocks**



**2<sup>nd</sup> argument specify size of each block.**

Example:-

```
int *p= (int*) calloc(5, 2);
```

```
int*p=(int *)calloc(5, size of (int));
```

Another difference between malloc and calloc is by default memory allocated by malloc contains garbage value, where as memory allocated by calloc is initialised by zero(but this initialisation) is not reliable.

### **realloc()**

The function realloc use to change the size of the memory block and it alter the size of the memory block without loosing the old data, it is called reallocation of memory.

It takes two argument such as;

```
int *ptr=(int *)malloc(size);
```

```
int*p=(int *)realloc(ptr, new size);
```

The new size allocated may be larger or smaller.

If new size is larger than the old size, then old data is not lost and newly allocated bytes are uninitialized. If old address is not sufficient then starting address contained in pointer may be changed and this reallocation function moves content of old block into the new block and data on the old block is not lost.

Example:

```
#include<stdio.h>
```

```
#include<alloc.h>
```

```
void main()
```

```
int i,*p;
```

```

p=(int*)malloc(5*size of (int));
if(p==null)
{
printf("space not available");
exit();
printf("enter 5 integer");
for(i=0;i<5;i++)
{
scanf("%d",(p+i));
int*ptr=(int*)realloc(9*size of (int) );
if(ptr==null)
{
printf("not available");
exit();
}
printf("enter 4 more integer");
for(i=5;i<9;i++)
scanf("%d",(p+i));
for(i=0;i<9;i++)
printf("%d",*(p+i));
}

```

**free()**

Function free() is used to release space allocated dynamically, the memory released by free() is made available to heap again. It can be used for further purpose.

Syntax for free declaration .

```
void(*ptr)
```

Or

```
free(p)
```

When program is terminated, memory released automatically by the operating system. Even we don't free the memory, it doesn't give error, thus lead to memory leak.

We can't free the memory, those didn't allocated.

## *Lecture Note: 29*

### **Dynamic array**

Array is the example where memory is organized in contiguous way, in the dynamic memory allocation function used such as malloc(), calloc(), realloc() always made up of contiguous way and as usual we can access the element in two ways as:

### **Subscript notation**

### **Pointer notation**

Example:

```

#include<stdio.h>
#include<alloc.h>
void main()
{
printf(“enter the no.of values”);
scanf(“%d”,&n);
p=(int*)malloc(n*size of int);
If(p==null)
printf(“not available memory”);
exit();
}
for(i=0;i<n;i++)
{
printf(“enter an integer”);
scanf(“%d”,&p[i]);
for(i=0;i<n;i++)
{
printf(“%d”,p[i]);
}
}
}

```

## **File handling**

**File:** the file is a permanent storage medium in which we can store the data permanently.

### **Types of file can be handled**

we can handle three type of file as

(1) **sequential file**

(2) **random access file**

(3) **binary file**

### **File Operation**

#### **opening a file:**

Before performing any type of operation, a file must be opened and for this fopen() function is used.

#### **syntax:**

```
file-pointer=fopen("FILE NAME ", "Mode of open");
```

example:

```
FILE *fp=fopen("ar.c", "r");
```

If fopen() unable to open a file than it will return NULL to the file pointer.

**File-pointer:** The file pointer is a pointer variable which can be store the address of a special file that means it is based upon the file pointer a file gets opened.

#### **Declaration of a file pointer:-**

```
FILE* var;
```

#### **Modes of open**

The file can be open in three different ways as

**Read mode 'r'/rt**

**Write mode 'w'/wt**

**Appened Mode 'a'/at**

**Reading** a character from a file

**getc()** is used to read a character into a file

Syntax:

```
character_variable=getc(file_ptr);
```

**Writing** a character into a file

**putc()** is used to write a character into a file

```
puts(character-var,file-ptr);
```

## **CLOSING A FILE**

**fclose()** function close a file.

```
fclose(file-ptr);
```

**fcloseall ()** is used to close all the opened file at a time

## **File Operation**

The following file operation carried out the file

(1)creation of a new file

(3)writing a file

(4)closing a file

Before performing any type of operation we must have to open the file.c, language communicate with file using A new type called **file pointer**.

### **Operation with fopen()**

File pointer=fopen(“FILE NAME”,”mode of open”);

If **fopen()** unable to open a file then it will return **NULL** to the file-pointer.

## **Lecture Note: 30**

### **Reading and writing a characters from/to a file**

**fgetc()** is used for reading a character from the file

#### **Syntax:**

character variable= fgetc(file pointer);

**fputc()** is used to writing a character to a file

#### **Syntax:**

fputc(character,file\_pointer);

```

/*Program to copy a file to another*/
#include<stdio.h>

void main()
{
FILE *fs,*fd;
char ch;
If(fs=fopen("scr.txt","r")==0)
{
printf("sorry....The source file cannot be opened");
return;
}
If(fd=fopen("dest.txt","w")==0)
{
printf("Sorry.....The destination file cannot be opened");
fclose(fs);
return;
}
while(ch=fgets(fs)!=EOF)
fputc(ch,fd);
fcloseall();
}

```



## Reading and writing a string from/to a file

`getw()` is used for reading a string from the file

### Syntax:

```
gets(file pointer);
```

`putw()` is used to writing a character to a file

### Syntax:

```
fputs(integer,file_pointer);
```

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
void main()
```

```
{
```

```
FILE *fp;
```

```
int word;
```

```
/*place the word in a file*/
```

```
fp=fopen("dgt.txt","wb");
```

```
If(fp==NULL)
```

```
{
```

```
printf("Error opening file");
```

```
exit(1);
```

```
}
```

```
word=94;
```

```
putw(word,fp);
```

```
If(ferror(fp))
```

```
printf("Error writing to file\n");
else
printf("Successful write\n");
fclose(fp);
/*reopen the file*/
fp=fopen("dgt.txt","rb");
If(fp==NULL)
{
printf("Error opening file");
exit(1);
}

/*extract the word*/
word=getw(fp);
If(ferror(fp))
printf("Error reading file\n");
else
printf("Successful read:word=%d\n",word);
/*clean up*/
fclose(fp);
}
```

**Reading and writing a string from/to a file**

**fgets()** is used for reading a string from the file

**Syntax:**

```
fgets(string, length, file pointer);
```

**fputs()** is used to writing a character to a file

**Syntax:**

```
fputs(string,file_pointer);
```

```
#include<string.h>
```

```
#include<stdio.h>
```

```
void main(void)
```

```
{
```

```
FILE*stream;
```

```
char string[]="This is a test";
```

```
char msg[20];
```

```
/*open a file for update*/
```

```
stream=fopen("DUMMY.FIL","w+");
```

```
/*write a string into the file*/
```

```
fwrite(string,strlen(string),1,stream);
```

```
/*seek to the start of the file*/
```

```
fseek(stream,0,SEEK_SET);
```

```
/*read a string from the file*/  
fgets(msg,strlen(string)+1,stream);  
/*display the string*/  
printf("%s",msg);  
fclose(stream);  
}
```

#### BOOKS:

- 1 E.Balagurusamy "Programming in C". Tata McGraw Hill
- 2 Y. Kanetkar "Let Us C". BPB publication
- 3 Ashok N. Kamthane "Programming with ANSI and TURBO C". Pearson Education
- 4 Programming in C, a complete introduction to the programming language, Stephan G. Kocham, third edition
- 5 C in Depth, S.K Srivastava and Deepali Srivastava



# MULTIMEDIA APPLICATIONS

UNIT 1- Introduction to Multimedia Systems... 1-36

UNIT 2- Content and project planning, Designing and  
development .....37- 72

UNIT 3- Using Image Editing Tools.....73-152

UNIT 4- Multimedia Authoring Tools.....153 -220


# **UNIT 1- INTRODUCTION TO MULTIMEDIA SYSTEMS**



# What is Multimedia?

---

1. **Multimedia** means that computer information can be represented through audio, video, and animation in addition to traditional media (i.e., text, graphics/drawings, images).
2. Multimedia is the field concerned with the computer-controlled integration of text, graphics, drawings, still and moving images (Video),



animation, audio, and any other media where every type of information can be represented, stored, transmitted and processed digitally.

- 3.** Multimedia is a media that uses multiple form of information content and information processing.





---

# History of Multimedia

---



Newspaper were perhaps the first mass communication medium to employ Multimedia - they used mostly text, graphics, and images.

In 1895, Guglielmo Marconi sent his first wireless radio transmission at Pontecchio, Italy. A few years later (in 1901) he detected radio waves beamed across the Atlantic. Initially invented for telegraph, radio is now a major medium for audio broadcasting.

Television was the new media for the 20th century. It brings the video and has since changed the world of mass communications.

Some of the important events in relation to Multimedia in Computing include:

- 1945 - Bush wrote about Memex
- 1967 - Negroponte formed the Architecture Machine Group at MIT
- 1969 - Nelson & Van Dam hypertext editor at Brown
- Birth of The Internet
- 1971 - Email
- 1976 - Architecture Machine Group proposal to DARPA: Multiple Media
- 1980 - Lippman & Mohl: Aspen Movie Map
- 1983 - Backer: Electronic Book

- . 1985 - Negroponte, Wiesner: opened MIT Media Lab
- . 1989 - Tim Berners-Lee proposed the World Wide Web to CERN (European Council for Nuclear Research)
- . 1990 - K. Hooper Woolsey, Apple Multimedia Lab, 100 people, educ.
- . 1991 - Apple Multimedia Lab: Visual Almanac, Classroom MM Kiosk
- . 1992 - The first M-bone audio multicast on the Net
- . 1993 - United Illinois National Centre for Supercomputing Applications: NCSA Mosaic
- . 1994 - Jim Clark and Marc Andreessen: Netscape
- . 1995 - JAVA for platform-independent application development. Duke is the first applet.
- . 1996 - Microsoft, Internet Explorer.

# **The Basic Elements of Multimedia**

---

**Text**

**Graphic**

**Animation**

**Video**

**Audio**

# The Basic Elements of Multimedia

## TEXT

- characters that are used to create words, sentences, and paragraphs.

- Multimedia is a rich medium that accommodates numerous instructional strategies. Multimedia addresses many of the challenges of instruction in both the academic and corporate environments. It is accessible over distance and time and provides a vehicle for consistent delivery. Multimedia can provide the best medium with which to communicate a concept.

### Hardware

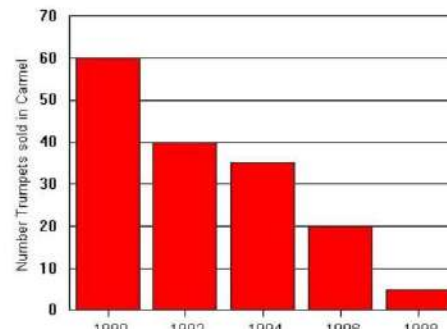
#### Components

- Monitor
- Keyboard
- Mouse
- Speaker

# The Basic Elements of Multimedia

## Graphics

- A digital representation of non-text information, such as a drawing, chart, or photograph.



# The Basic Elements of Multimedia

---

## Animation

- Flipping through a series of still images. It is a series of graphics that create an illusion of motion.



# The Basic Elements of Multimedia

---

## Video

- photographic images that are played back at speeds of 15 to 30 frames a second and they provide the appearance of full motion.





# The Basic Elements of Multimedia

---

## Audio

☞ music, speech, or any other sound.



# HARDWARE COMPONENTS OF MULTIMEDIA

---

## CPU- Central Processing Unit:

- The “brain” of the computer
- It’s the component that does all the heavy lifting (computational work)
- The faster the processor, the faster the computer can work
- Two factors – speed vs multi core.

## RAM- Random Access Memory:

- Temporary storage area
- This is where projects you are working on reside until you save to permanent storage
- Holds data for the CPU waiting for it to be processed.

- **The more RAM you have, the more applications you can work on at the same time**
- **The faster your RAM is, the more your processor can do.**
- **Modern multimedia systems will generally have at least 8GB of DDR3 RAM.**

### **Graphics Card:**

- **This is like the CPU but designed for graphics**
- **The GPU will often determine how fast images/graphics will move when animated**
- **Determines the computers maximum display resolution**
- **Can have more than one core (just like a CPU). Have RAM to store data being worked on**
- **As they can process a lot, they usually require lots of cooling**
- **Vital to any multimedia workstation.**

### **MotherBoard:**

- **Sometimes called Mainboard**

- **Component that connects everything together • Controllers for moving data around the computer**
- **Processor brand/type specific – usually can't mix and match between architectures**
- **Some will contain built-in components such as GPU or Sound Processor.**

### **SoundCard:**

- **Needed so you can hear sound from your computer**
- **Converts digital data into sound waves**
- **Most systems have a sound controller built into the Motherboard and use the CPU for processing.**

### **Hard Disk Drive(HDD):**

- **HDD – traditional storage medium for nearly all computers**
- **Mechanical disk made up of metal platters that are magnetised to hold data.**

## **Monitor (CRT) or Flat Panel Display:**

- Often overlooked but incredibly important
- A good quality monitor will represent colours correctly
- Larger sizes are important as they will support higher resolutions
- Most designers will usually have more than one connected to their computer and have a large screen capable of better than 1920x1080 resolution.

## **Keyboard & Mouse:**

- Vital input devices
- A good quality mouse can make the difference between clean, accurate designs and dodgy lines
- Keyboard types – traditional vs mechanical
- Mouse types – Optical vs laser.

## **Other Components:**

- **Chassis or case** – box that holds all the parts of a computer
- **Power supply** – device that converts AC into DC for the computer and powers all components
- **Optical Drives** – CD/DVD/Blu-Ray – input and output mediums used for many multimedia projects
- **Drawing tablets** – connect to your PC and allow you to draw using a stylus/pen
- **Headphones/speakers** – essential with multimedia presentations
- **External storage** – USB drives, hard disk drives, etc. for backup.

# **SOFTWARE COMPONENTS OF MULTIMEDIA**

---

The basic tool set for building a multimedia project can be divided into categories: –

- **Painting and Drawing Tools**
- **3-D Modeling**
- **Image editing tools**
- **Sound editing tools**
- **Animation Video**
- **Digital Movie tools**

## **Painting and Drawing Tools**

- Painting and drawing tools generally come with a graphical user interface with pull down menus for quick selection.
- You can create almost all kinds of possible shapes and resize them.
- Drawing file can be imported or exported in many image formats like .gif, .tiff, .jpg, .bmp, tec.
- Good drawing software – “Corel Draw”.

## **3D Modeling tools**

- Realism means that you depict things in the way they actually are.
- It tools the objects that appear in perception in your project can look realistic.



- These tools offer features like multiple windows to view your design in each dimension.
- Tools provide drag and drop menu.
- A good 3D modeling tool is “3D Studio Max”.

## **Image editing tools**

- These tools are used to edit existing bitmap images and pictures.
- They are similar to painting and drawing tools as they can also create images from scratch.
- It is also capable of converting the image data type file format.
- It is used usually for reinventing and recreating the image.
- Image process software – Adobe Photoshop & Paint Shop pro.

# **Animation, Video and digital movies editing tools**

- Animations are graphic scenes played back sequentially and rapidly.
- These tools enable you to edit and assemble video clips captured from camera, animations and other sources.
- The completed clip with added transition and visual effects could be played back.
- Adobe Premiere and Media Shop Pro are two good examples of these tools.

## **MULTIMEDIA FILE FORMATS**

### **1. Image File Formats:**

- a. JPEG (Joint Photographic Experts Group).
- b. PNG (Portable Network Graphics).
- c. TIFF (Tagged Image File Format).

- d. GIF (Graphic Interchange Format).
- e. BMP (BitMap).
- f. SVG (Scalable Vector Graphics).

## **2. Sound File Formats:**

- a. AIFF Audio
- b. MIDI
- c. MP3
- d. WAV
- e. AV Audio
- f. MPEG Audio
- g. Real Audio or RA
- h. VOC

## **3. Video File Formats**

- a. MPEG Video
- b. AVI
- c. Quick time format.

## **4. Animation File Formats**

- a. Director
- b. Animator Pro
- c. 3D studio max
- d. CompuServe GIF 89a
- e. Flash.
- f. MPEG.

# Usage of Multimedia

---

Multimedia finds its application in various areas including, but not limited to:

Advertisements

Art

Education

Entertainment

Engineering

Medicine

Mathematics

Business



# Scientific research

# Usage

---

In education, multimedia can be used as a source of information. Students can search encyclopaedias such as Encarta, which provide facts on a variety of different topics using multimedia presentations.

Teachers can use multimedia presentations to make lessons more interesting by using animations to highlight or demonstrate key points.

# Usage

---

A multimedia presentation can also make it easier for pupils to read text rather than trying to read a teacher's writing on the board.

Programs which show pictures and text whilst children are reading a story can help them learn to read; these too are a form of multimedia presentation.



# Usage

---

Multimedia is used for advertising and selling products on the Internet.

Some businesses use multimedia for training where CD-ROMs or on-line tutorials allow staff to learn at their own speed, and at a suitable time to the staff and the company.

Another benefit is that the company do not have to pay the additional expenses of an employee attending a course away from the workplace.

# Usage

---

People use the Internet for a wide range of reasons, including shopping and finding out about their hobbies.

The Internet has many multimedia elements embedded in web pages and web browsers support a variety of multimedia formats.

Many computer games use sound tracks, 3D graphics and video clips.

# Usage

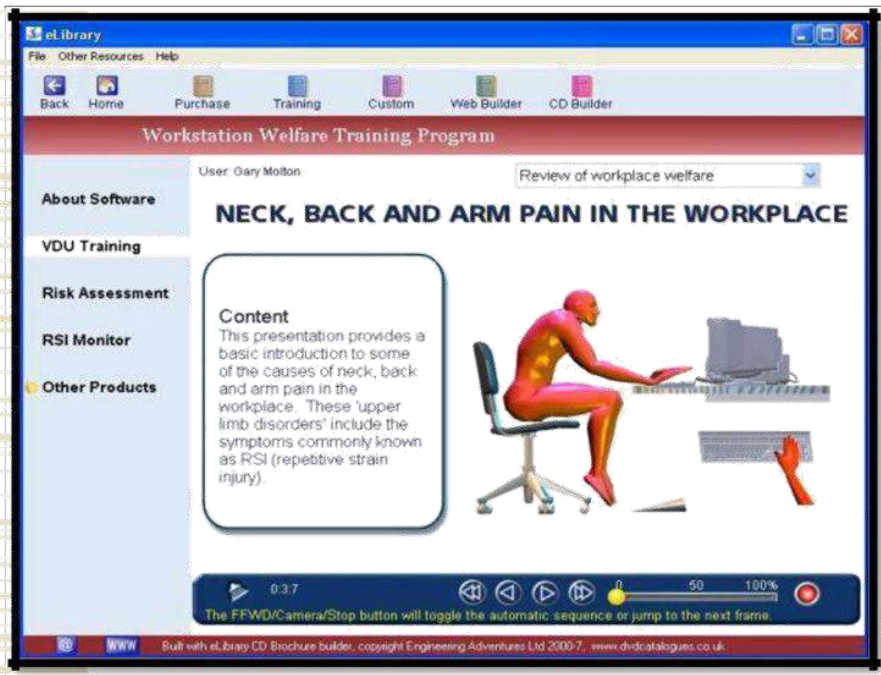


**Computer-Based Training**



**Teaching Aid**

# Usage



References

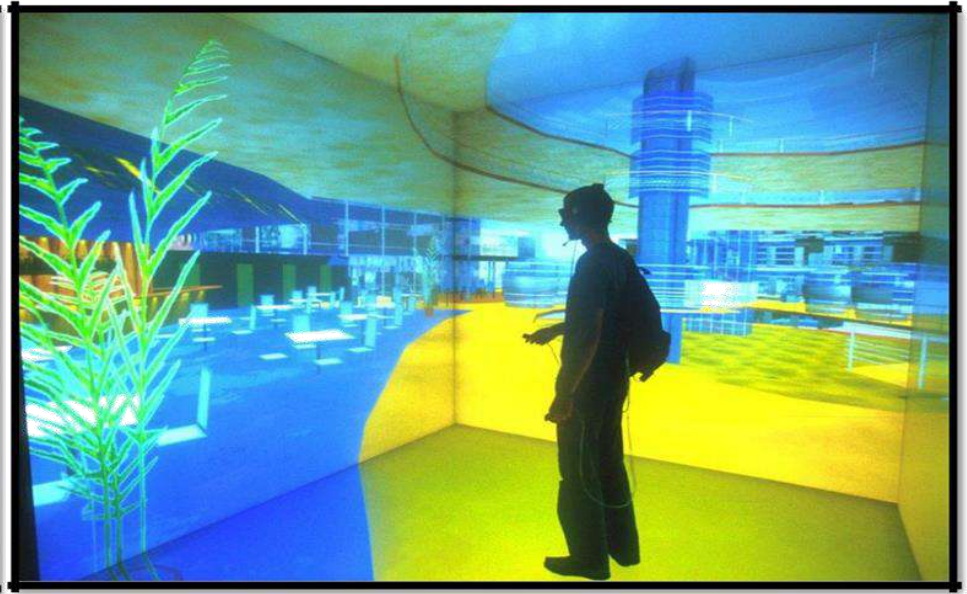


Entertainment

# Usage



Simulation



Virtual Reality

# Usage



**Virtual Surgery**



**Information Kiosk**



# Advantages of using Multimedia

---

- It is very user-friendly. It doesn't take much energy out of the user, in the sense that you can sit and watch the presentation, you can read the text and hear the audio.
- It is multi sensorial. It uses a lot of the user's senses while making use of multimedia, for example hearing, seeing and talking.
- It is integrated and interactive. All the different mediums are integrated through the digitisation process. Interactivity is heightened by the possibility of easy feedback.
- It is flexible. Being digital, this media can easily be changed to fit different situations and audiences.
- It can be used for a wide variety of audiences, ranging from one person to a whole group.

# Disadvantages of using Multimedia

---

- Information overload. Because it is so easy to use, it can contain too much information at once.
- It takes time to compile. Even though it is flexible, it takes time to put the original draft together.



- It can be expensive. As mentioned in one of my previous posts, multimedia makes use of a wide range of resources, which can cost you a large amount of money.
  - Too much makes it unpractical. Large files like video and audio has an effect of the time it takes for your presentation to load. Adding too much can mean that you have to use a larger computer to store the files.
-

# UNIT 2- CONTENT AND PROJECT PLANNING, DESIGNING AND DEVELOPMENT

**MULTIMEDIA PLANNING:** Multimedia projects are complex; they often involve the skills and efforts of multiple teams or people. During the development process, a project moves through the specialized parts of the team, from story creation to technical editing, with regular collective review sessions. Each stage is designed to refine the project with attention to the client's needs, technical requirements and audience preferences.

## **1. Planning Meeting**

A planning meeting is a crucial part of the multimedia development process; it creates a shared vision for everyone working on the project. The meeting usually kicks off a project, bringing together the team. During the meeting, the project manager communicates the major goals and lays out the milestones. The meeting may include a discussion of the target audience and how each division can help support the overarching goal.

## **2. Script Writing**

Most multimedia projects have a story behind them. After the initial meeting, the people in charge of the background story write a script, creative brief or outline. The text hits the main points of the project and uses language that appeals to the audience in jargon, tone and style.

## **3. Story Boarding**

A multimedia project usually includes multiple pieces: audio, video, imagery, text for voiceovers and on-screen titles. Story boarding ties everything together; a story board panel for a scene includes a sketch of the visual elements, the voiceover or title text, and any production notes. It guides the process, keeps everyone in check and gives structure to the project.

## **4. Designing**

During the design stage, designers take over the visual aspects of the project to determine how it looks and feels. Using the notes from the storyboard, they create graphics, design the navigation and give direction to photographers and videographers regarding the correct shots.

Depending on the project, the design stage might include graphic design, web design, information design, photography or image collection. Design is always done with an eye toward the audience

## 5. Editing

Editing is one of the most involved and complex stages of the multimedia development process. The people responsible for editing the project turn the various pieces into a cohesive product, taking into consideration the time constraints, story line and creative specifications. Depending on the scope of the project, pieces of the project may be edited separately. For projects with a large amount of video, editing is often the longest stage of the process; a minute of final video can take hours of editing. The editing stage usually involves internal review iterations and may also include rounds of client review and editing.

## 6. Production

The production stage is when all the parts of a multimedia project come together. The production staff gathers all of the edited assets in one place and puts them together in a logical sequence, using the story board as a guide. The rough draft is then put through rounds of review and final edits, both internally and with the client. To ensure that a project has the desired impact on the target audience, a company may engage in user testing as part of production. During this stage, test members of the audience use the multimedia piece while team members observe. Depending on the goals of the project, the staff might observe users' reactions or have them answer questions to see if the project hits the right marks. After user testing, there are usually further adjustments to the project. Once the team and clients are satisfied, the project goes out for distribution.

---

# Data compression

In signal processing, **datacompression**, **source coding**, or **bit-rate reduction** involves encoding information using fewer bits than the original representation. Compression can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating

statistical redundancy. No information is lost in lossless compression. Lossy compression reduces bits by removing unnecessary or less important information. The process of reducing the size of a data file is often referred to as data compression. In the context of data transmission, it is called source coding; encoding done at the source of the data before it is stored or transmitted. Source coding should not be confused with channel coding, for error detection and correction or line coding, the means for mapping data onto a signal. Compression is useful because it reduces resources required to store and transmit data. Computational resources are consumed in the compression process and, usually, in the reversal of the process (decompression). Data compression is subject to a space–time complexity trade-off. For instance, a compression scheme for video may require expensive hardware for the video to be decompressed fast enough to be viewed as it is being decompressed, and the option to decompress the video in full before watching it may be inconvenient or require additional storage. The design of data compression schemes involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (when using lossy data compression), and the computational resources required to compress and decompress the data.

## **Lossless Compression**

Lossless data compression algorithms usually exploit statistical redundancy to represent data without losing any information, so that the process is reversible. Lossless compression is possible because most real-world data exhibits statistical redundancy. For example, an image may have areas of colour that do not change over several pixels; instead of coding “red pixel, red pixel, ...” the data may be encoded as “279 red pixels”. This is a basic example of run-length encoding; there are many schemes to reduce file size by eliminating redundancy.

The Lempel–Ziv (LZ) compression methods are among the most popular algorithms for lossless storage. DEFLATE is a variation on LZ optimized for decompression speed and compression ratio, but compression can be slow. In the mid-1980s, following work by Terry Welch, the Lempel–Ziv–Welch (LZW) algorithm rapidly became the method of choice for most general-purpose compression systems. LZW is used in GIF images, programs such as PKZIP, and hardware devices such as modems. LZ methods use a table-based compression model where table entries are substituted for repeated strings of data. For most LZ methods, this table is generated dynamically from earlier data in the input. The table itself is

often Huffman encoded. The strongest modern lossless compressors use probabilistic models, such as prediction by partial matching. The Burrows–Wheeler transform can also be viewed as an indirect form of statistical modelling.

The class of grammar-based codes are gaining popularity because they can compress *highly repetitive* input extremely effectively, for instance, a biological data collection of the same or closely related species, a huge versioned document collection, internet archival, etc. The basic task of grammar-based codes is constructing a context-free grammar deriving a single string. Sequitur and Re-Pair are practical grammar compression algorithms for which software is publicly available. In a further refinement of the direct use of probabilistic modelling, statistical estimates can be coupled to an algorithm called arithmetic coding. Arithmetic coding is a more modern coding technique that uses the mathematical calculations of a finite-state machine to produce a string of encoded bits from a series of input data symbols. It can achieve superior compression to other techniques such as the better-known Huffman algorithm. It uses an internal memory state to avoid the need to perform a one-to-one mapping of individual input symbols to distinct representations that use an integer number of bits, and it clears out the internal memory only after encoding the entire string of data symbols. Arithmetic coding applies especially well to adaptive data compression tasks where the statistics vary and are context-dependent, as it can be easily coupled with an adaptive model of the probability distribution of the input data. An early example of the use of arithmetic coding was its use as an optional (but not widely used) feature of the JPEG image coding standard. It has since been applied in various other designs including H.263, H.264/MPEG-4 AVC and HEVC for video coding.

---

## Lossy Compression

Lossy data compression is the converse of lossless data compression. In the late 1980s, digital images became more common, and standards for compressing them emerged. In the early 1990s, lossy compression methods began to be widely used. In these schemes, some loss of information is acceptable. Dropping nonessential detail from the data source can save storage space. Lossy data compression schemes are designed by research on how people perceive the data in question. For example, the human eye is more sensitive to subtle variations in luminance than it is to the variations in color. JPEG image compression works in parts by rounding off non-essential bits of information. There is a corresponding trade-off between preserving information and reducing size. A number of popular compression formats exploit

these perceptual differences, including those used in music files, images, and video. Lossy image compression can be used in digital cameras, to increase storage capacities with minimal degradation of picture quality. Similarly, DVDs use the lossy MPEG-2 video coding format for video compression. In lossy audio compression, methods of psychoacoustics are used to remove non-audible (or less audible) components of the audio signal. Compression of human speech is often performed with even more specialized techniques; speech coding, or voice coding, is sometimes distinguished as a separate discipline from *audio compression*. Different audio and speech compression standards are listed under audio coding formats. *Voice compression* is used in internet telephony, for example, audio compression is used for CD ripping and is decoded by the audio players.

## **Theory**

The theoretical background of compression is provided by information theory (which is closely related to algorithmic information theory) for lossless compression and rate–distortion theory for lossy compression. These areas of study were essentially created by Claude Shannon, who published fundamental papers on the topic in the late 1940s and early 1950s. Coding theory is also related to this. The idea of data compression is also deeply connected with statistical inference.

## **Machine learning**

There is a close connection between machine learning and compression: a system that predicts the posterior probabilities of a sequence given its entire history can be used for optimal data compression (by using arithmetic coding on the output distribution) while an optimal compressor can be used for prediction (by finding the symbol that compresses best, given the previous history). The equivalence has been used as a justification for using data compression as a benchmark for “general intelligence.”

## **Feature space vectors**

However, a new, alternative view can show compression algorithms implicitly map strings into implicit feature space vectors, and compression-based similarity measures compute similarity within these feature spaces. For each compressor  $C(\cdot)$  we define an associated vector space  $\mathfrak{N}$ , such that  $C(\cdot)$  maps an input string  $x$ , corresponds to the vector norm  $\|\tilde{x}\|$ . An exhaustive examination of the feature spaces underlying all compression algorithms is

precluded by space; instead, feature vectors chooses to examine three representative lossless compression methods, LZW, LZ77, and PPM.

## **Data differencing**

Data compression can be viewed as a special case of data differencing: Data differencing consists of producing a *difference* given a *source* and a *target*, with patching producing a *target* given a *source* and a *difference*, while data compression consists of producing a compressed file given a target, and decompression consists of producing a target given only a compressed file. Thus, one can consider data compression as data differencing with empty source data, the compressed file corresponding to a “difference from nothing.” This is the same as considering absolute entropy (corresponding to data compression) as a special case of relative entropy (corresponding to data differencing) with no initial data.

When one wishes to emphasize the connection, one may use the term *differential compression* to refer to data differencing.

## **Uses**

### **Audio**

Audio data compression, not to be confused with dynamic range compression, has the potential to reduce the transmission bandwidth and storage requirements of audio data. Audio compression algorithms are implemented in software as audio codecs. Lossy audio compression algorithms provide higher compression at the cost of fidelity and are used in numerous audio applications. These algorithms almost all rely on psychoacoustics to eliminate or reduce fidelity of less audible sounds, thereby reducing the space required to store or transmit them.

In both lossy and lossless compression, information redundancy is reduced, using methods such as coding, pattern recognition, and linear prediction to reduce the uncompressed data.

The acceptable trade-off between loss of audio quality and transmission or storage size depends upon the application. For example, one 640 MB compact disc (CD) holds approximately one hour of uncompressed high fidelity music, less than 2 hours of music compressed lossless, or 7 hours of music compressed in the MP3 format at a medium bit

rate. A digital sound recorder can typically store around 200 hours of clearly intelligible speech in 640 MB.

Lossless audio compression produces a representation of digital data that decompress to an exact digital duplicate of the original audio stream, unlike playback from lossy compression techniques such as Vorbis and MP3. Compression ratios are around 50–60 % of original size, which is similar to those for generic lossless data compression. Lossless compression is unable to attain high compression ratios due to the complexity of waveforms and the rapid changes in sound forms. Codecs like FLAC, Shorten, and TTA use linear prediction to estimate the spectrum of the signal. Many of these algorithms use convolution with the filter  $[-1 \ 1]$  to slightly whiten or flatten the spectrum, thereby allowing traditional lossless compression to work more efficiently. The process is reversed upon decompression. When audio files are to be processed, either by further compression or for editing, it is desirable to work from an unchanged original (uncompressed or losslessly compressed). Processing of a lossily compressed file for some purpose usually produces a final result inferior to the creation of the same compressed file from an uncompressed original. In addition to sound editing or mixing, lossless audio compression is often used for archival storage, or as master copies.

A number of lossless audio compression formats exist. Shorten was an early lossless format. Newer ones include Free Lossless Audio Codec (FLAC), Apple's Apple Lossless (ALAC), MPEG-4 ALS, Microsoft's Windows Media Audio 9 Lossless (WMA Lossless), Monkey's Audio, TTA, and WavPack. See list of lossless codecs for a complete listing. Some audio feature a combination of lossy format and a lossless correction; this allows stripping the correction to easily obtain a lossy file. Such formats include MPEG-4 SLS (Scalable to Lossless), WavPack, and OptimFROG DualStream.

Other formats are associated with a distinct system, such as:

- Direct Stream Transfer
- Meridian Lossless Encoding

### **Lossy audio compression**

Lossy audio compression is used in a wide range of applications. In addition to the direct applications (MP3 players or computers), digitally compressed audio streams are used in most video DVDs, digital television, streaming media on the internet, satellite and cable radio, and increasingly in terrestrial radio broadcasts. Lossy compression typically achieves



---

far greater compression than lossless compression (data of 5 percent to 20 percent of the original stream, rather than 50 percent to 60 percent), by discarding less-critical data.

The innovation of lossy audio compression was to use psychoacoustics to recognize that not all data in an audio stream can be perceived by the human auditory system. Most lossy compression reduces perceptual redundancy by first identifying perceptually irrelevant sounds, that is, sounds that are very hard to hear. Typical examples include high frequencies or sounds that occur at the same time as louder sounds. Those sounds are coded with decreased accuracy or not at all. Due to the nature of lossy algorithms, audio quality suffers when a file is decompressed and recompressed (digital generation loss). This makes lossy compression unsuitable for storing the intermediate results in professional audio engineering applications, such as sound editing and multitrack recording. However, they are very popular with end users (particularly MP3) as a megabyte can store about a minute's worth of music at adequate quality.

### **Coding methods**

To determine what information in an audio signal is perceptually irrelevant, most lossy compression algorithms use transforms such as the modified discrete cosine transform (MDCT) to convert time domain sampled waveforms into a transform domain. Once transformed, typically into the frequency domain, component frequencies can be allocated bits according to how audible they are. Audibility of spectral components calculated using the absolute threshold of hearing and the principles of simultaneous masking—the phenomenon wherein a signal is masked by another signal separated by frequency—and, in some cases, temporal masking—where a signal is masked by another signal separated by time. Equal-loudness contours may also be used to weight the perceptual importance of components. Models of the human ear-brain combination incorporating such effects are often called psychoacoustic models.

Other types of lossy compressors, such as the linear predictive coding (LPC) used with speech, are source-based coders. These coders use a model of the sound's generator (such as the human vocal tract with LPC) to whiten the audio signal (i.e., flatten its spectrum) before quantization. LPC may be thought of as a basic perceptual coding technique: reconstruction of an audio signal using a linear predictor shapes the coder's quantization noise into the spectrum of the target signal, partially masking it.

Lossy formats are often used for the distribution of streaming audio or interactive applications (such as the coding of speech for digital transmission in cell phone networks). In such applications, the data must be decompressed as the data flows, rather than after the entire data stream has been transmitted. Not all audio codecs can be used for streaming applications, and for such applications a codec designed to stream data effectively will usually be chosen.

Latency results from the methods used to encode and decode the data. Some codecs will analyze a longer segment of the data to optimize efficiency, and then code it in a manner that requires a larger segment of data at one time to decode. (Often codecs create segments called a "frame" to create discrete data segments for encoding and decoding.) The inherent latency of the coding algorithm can be critical; for example, when there is a two-way transmission of data, such as with a telephone conversation, significant delays may seriously degrade the perceived quality.

In contrast to the speed of compression, which is proportional to the number of operations required by the algorithm, here latency refers to the number of samples that must be analysed before a block of audio is processed. In the minimum case, latency is zero samples (e.g., if the coder/decoder simply reduces the number of bits used to quantize the signal). Time domain algorithms such as LPC also often have low latencies, hence their popularity in speech coding for telephony. In algorithms such as MP3, however, a large number of samples have to be analyzed to implement a psychoacoustic model in the frequency domain, and latency is on the order of 23 ms (46 ms for two-way communication)).

### **Speech encoding**

Speech encoding is an important category of audio data compression. The perceptual models used to estimate what a human ear can hear are generally somewhat different from those used for music. The range of frequencies needed to convey the sounds of a human voice are normally far narrower than that needed for music, and the sound is normally less complex. As a result, speech can be encoded at high quality using a relatively low bit rate. If the data to be compressed is analog (such as a voltage that varies with time), quantization is employed to digitize it into numbers (normally integers). This is referred to as analog-to-digital (A/D) conversion. If the integers generated by quantization are 8 bits each, then the entire range of the analog signal is divided into 256 intervals and all the signal values within an interval are

quantized to the same number. If 16-bit integers are generated, then the range of the analog signal is divided into 65,536 intervals.

This relation illustrates the compromise between high resolution (a large number of analog intervals) and high compression (small integers generated). This application of quantization is used by several speech compression methods. This is accomplished, in general, by some combination of two approaches:

Only encoding sounds that could be made by a single human voice.

Throwing away more of the data in the signal—keeping just enough to reconstruct an “intelligible” voice rather than the full frequency range of human hearing.

Perhaps the earliest algorithms used in speech encoding (and audio data compression in general) were the A-law algorithm and the  $\mu$ -law algorithm.

## HISTORY

A literature compendium for a large variety of audio coding systems was published in the IEEE Journal on Selected Areas in Communications (JSAC), February 1988. While there were some papers from before that time, this collection documented an entire variety of finished, working audio coders, nearly all of them using perceptual (i.e. masking) techniques and some kind of frequency analysis and back-end noiseless coding.<sup>[24]</sup> Several of these papers remarked on the difficulty of obtaining good, clean digital audio for research purposes. Most, if not all, of the authors in the JSAC edition were also active in the MPEG-1 Audio committee.

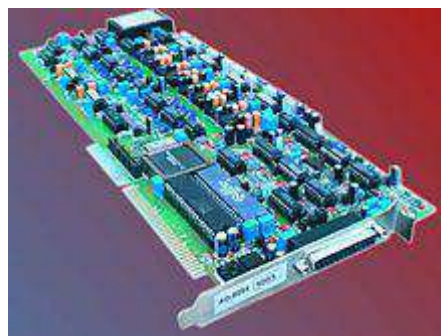


Figure: Solidyne 922: The world's first Commercial audio bit compression card for PC, 1990.

The world's first commercial broadcast automation audio compression system was developed by Oscar Bonello, an engineering professor at the University of Buenos Aires. In 1983, using the psychoacoustic principle of the masking of critical bands first published in 1967, he started developing a practical application based on the recently developed IBM PC computer, and the broadcast automation system was launched in 1987 under the name Audicom. Twenty years later, almost all the radio stations in the world were using similar technology manufactured by a number of companies.

## **Video**

Video compression is a practical implementation of source coding in information theory. In practice, most video codecs are used alongside audio compression techniques to store the separate but complementary data streams as one combined package using so-called *container formats*. Uncompressed video requires a very high data rate. Although lossless video compression codecs perform at a compression factor of 5 to 12, a typical MPEG-4 lossy compression video has a compression factor between 20 and 200.

## **Encoding theory**

Video data may be represented as a series of still image frames. Such data usually contains abundant amounts of spatial and temporal redundancy. Video compression algorithms attempt to reduce redundancy and store information more compactly.

Most video compression formats and codecs exploit both spatial and temporal redundancy (e.g. through difference coding with motion compensation). Similarities can be encoded by only storing differences between e.g. temporally adjacent frames (inter-frame coding) or spatially adjacent pixels (intra-frame coding). Inter-frame compression (a temporal delta encoding) is one of the most powerful compression techniques. It (re)uses data from one or more earlier or later frames in a sequence to describe the current frame. Intra-frame coding, on the other hand, uses only data from within the current frame, effectively being still-image compression. And the intra-frame coding always uses lossy compression algorithms.

A class of specialized formats used in camcorders and video editing use less complex compression schemes that restrict their prediction techniques to intra-frame prediction. Usually video compression additionally employs lossy compression techniques like quantization that reduce aspects of the source data that are (more or less) irrelevant to the

human visual perception by exploiting perceptual features of human vision. For example, small differences in color are more difficult to perceive than are changes in brightness. Compression algorithms can average a color across these similar areas to reduce space, in a manner similar to those used in JPEG image compression. As in all lossy compression, there is a trade-off between video quality, cost of processing the compression and decompression, and system requirements. Highly compressed video may present visible or distracting artifacts. \_\_\_\_\_

Other methods than the prevalent DCT-based transform formats, such as fractal compression, matching pursuit and the use of a discrete wavelet transform (DWT), have been the subject of some research, but are typically not used in practical products (except for the use of wavelet coding as still-image coders without motion compensation). Interest in fractal compression seems to be waning, due to recent theoretical analysis showing a comparative lack of effectiveness of such methods.

### **Inter-frame coding**

Inter-frame coding works by comparing each frame in the video with the previous one. Individual frames of a video sequence are compared from one frame to the next, and the video compression codec sends only the differences to the reference frame. If the frame contains areas where nothing has moved, the system can simply issue a short command that copies that part of the previous frame into the next one. If sections of the frame move in a simple manner, the compressor can emit a (slightly longer) command that tells the decompressor to shift, rotate, lighten, or darken the copy. This longer command still remains much shorter than intraframe compression. Usually the encoder will also transmit a residue signal which describes the remaining subtler differences to the reference imagery. Using entropy coding, these residue signals have a more compact representation than the full signal. In areas of video with more motion, the compression must encode more data to keep up with the larger number of pixels that are changing. Commonly during explosions, flames, flocks of animals, and in some panning shots, the high-frequency detail leads to quality decreases or to increases in the variable bitrate.

### **Hybrid block-based transform formats**

In the prediction stage, various deduplication and difference-coding techniques are applied that help decorrelate data and describe new data based on already transmitted data.

Then rectangular blocks of (residue) pixel data are transformed to the frequency domain to ease targeting irrelevant information in quantization and for some spatial redundancy reduction. The discrete cosine transform (DCT) that is widely used in this regard was introduced by N. Ahmed, T. Natarajan and K. R. Rao in 1974. In the main lossy processing stage that data gets quantized in order to reduce information that is irrelevant to human visual perception.

In the last stage statistical redundancy gets largely eliminated by an entropy coder which often applies some form of arithmetic coding. In an additional in-loop filtering stage various filters can be applied to the reconstructed image signal. By computing these filters also inside the encoding loop they can help compression because they can be applied to reference material before it gets used in the prediction process and they can be guided using the original signal. The most popular example are deblocking filters that blur out blocking artefacts from quantization discontinuities at transform block boundaries.

## **History**

All basic algorithms of today's dominant video codec architecture have been invented before 1979. In 1950, the Bell Labs filed the patent on DPCM which soon was applied to video coding. Entropy coding started in the 1940s with the introduction of Shannon–Fano coding on which the widely used Huffman coding is based that was developed in 1950; the more modern context-adaptive binary arithmetic coding (CABAC) was published in the early 1990s. Transform coding (using the Hadamard transform) was introduced in 1969, the popular discrete cosine transform (DCT) appeared in 1974 in scientific literature. The ITU-T's standard H.261 from 1988 introduced the prevalent basic architecture of video compression technology.

---

## **Genetics**

Genetics compression algorithms are the latest generation of lossless algorithms that compress data (typically sequences of nucleotides) using both conventional compression algorithms and genetic algorithms adapted to the specific datatype. In 2012, a team of scientists from Johns Hopkins University published a genetic compression algorithm that

does not use a reference genome for compression. HAPZIPPER was tailored for HapMap data and achieves over 20-fold compression (95% reduction in file size), providing 2- to 4-fold better compression and in much faster time than the leading general-purpose compression utilities. For this, Chanda, Elhaik, and Bader introduced MAF based encoding (MAFE), which reduces the heterogeneity of the dataset by sorting SNP by their minor allele frequency, thus homogenizing the dataset. Other algorithms in 2009 and 2013 (DNAZip and GenomeZip) have compression ratios of up to 1200-fold—allowing 6 billion basepair diploid human genomes to be stored in 2.5 megabytes (relative to a reference genome or averaged over many genomes).

---

## **CAPTURING IMAGES FROM DIGITAL CAMERA AND SCANNER**

### ***Scanners***

An image scanner is able to scan and capture text, imagery and objects and then turn them into a digital image, the most common scanners are flatbeds, this involves the object or document being placed on a horizontal glass surface whilst a bright light illuminates the surface and subsequently the object on the scanner is then converted to a digital image that can be seen and edited via the connected computer.

### ***Digital Cameras***

Digital Cameras like scanners are capable of image capture, except a digital camera is more transportable making them the preferable choice, this is also down to the fact that a Digital Camera is able to capture images of everyday life compared to the Scanner which requires invariably a flat document. Digital Cameras capture the image and then immediately present the image on the camera, the majority of modern cameras allow you to edit or filter images that have been stored.

### ***Resolution***

The resolution on an image file is entirely dependent on the Pixels per Inch, for example a digital camera that produces a 3000 x 2000-pixel image will only be this resolution provided that it is displayed at a certain size, however if the size is increased the image resolution will suffer, likewise if the image is decreased the resolution in theory should increase because the

digital image will always have 6 million captured pixels, no matter what this figure does not change.

### ***Storage***

Storage of images depends on the amount of memory that your computer has and the actual file size of the image, often to reduce the file size of the image in order to save the image, images can be changed to different formats such as JPEG's in order to reduce the size of the file. Asset management ensures that all of the documents or files that are especially important to you are correctly stored within the computer.

### **Where would you use this?**

Scanners would mainly be used when you are scanning a certain paper document or flat image, this is due to the fact that a scanner is a piece of hardware that is not very transportable and only has one glass screen for the document to be placed. However, it makes much more sense to scan a paper document as opposed to digitally capturing the document via a digital camera where uploading to a computer system takes much more effort. It is unlikely though that a scanner would be used for anything 3D, this is because the scanner has to be closed in order to scan properly, a 3D object would not allow this to happen.

Digital Cameras would be used when capturing any 3D image that is unable to be scanned, because of the versatility of Digital Cameras you can take them almost everywhere and capture images of everything that you are able to, essentially there is no limit to what you can capture with a Digital camera, however there is one major issue, this being that in order to transfer the imagery to a computer you will need either a transfer cable or a memory card that can be inserted into the computer, whereas a scanner is mainly universal to the majority of computers.

### **Benefits for user and business:**

The obvious key benefit to using a scanner for a business is that they can scan important documents that relate to the business, a scanner enables a business to turn any documents into a digital document that becomes available on the network, because the majority of business is conducted through the internet Scanners are becoming ever more important especially for businesses who have previously kept their documents on paper. Scanners essentially allow a business to save documents that were previously only on paper, this is also the main benefit for the majority of computer users.



Digital Cameras are a huge advantage for both businesses and computer users, for example computer users who may conduct a lot of online selling may choose to use a Digital Camera in order to capture an image of the product they are selling, by uploading this image on their computer they can quickly utilise the image by displaying it as their product. Likewise, Businesses will often hire professional photographers to take pictures and images for their website, the images usually consist of products the business is selling or general photos to liven up the webpage. More commonly businesses will use digitally produced imagery as their website background.

---

## **IMAGE EDITING:**

**Image editing** encompasses the processes of altering images, whether they are digital photographs, traditional photo-chemical photographs, or illustrations. Traditional analog image editing is known as photo retouching, using tools such as an airbrush to modify photographs, or editing illustrations with any traditional art medium. Graphic software programs, which can be broadly grouped into vector graphics editors, raster graphics editors, and 3D modelers, are the primary tools with which a user may manipulate, enhance, and transform images. Many image editing programs are also used to render or create computer art from scratch.

### **Basics of image editing**

Raster images are stored in a computer in the form of a grid of picture elements, or pixels. These pixels contain the image's color and brightness information. Image editors can change the pixels to enhance the image in many ways. The pixels can be changed as a group, or individually, by the sophisticated algorithms within the image editors. This article mostly refers to bitmap graphics editors, which are often used to alter photographs and other raster graphics. However, vector graphics software, such as Adobe Illustrator, CorelDRAW, Xara Designer Pro, PixelStyle Photo Editor, Inkscape or Vectr, are used to create and modify

vector images, which are stored as descriptions of lines, Bézier curves, and text instead of pixels. It is easier to rasterize a vector image than to vectorize a raster image; how to go about vectorizing a raster image is the focus of much research in the field of computer vision. Vector images can be modified more easily, because they contain descriptions of the shapes for easy rearrangement. They are also scalable, being rasterizable at any resolution.

### **Automatic image enhancement**

Camera or computer image editing programs often offer basic automatic image enhancement features that correct color hue and brightness imbalances as well as other image editing features, such as red eye removal, sharpness adjustments, zoom features and automatic cropping. These are called automatic because generally they happen without user interaction or are offered with one click of a button or mouse button or by selecting an option from a menu. Additionally, some automatic editing features offer a combination of editing actions with little or no user interaction.

### **Digital data compression**

Many image file formats use data compression to reduce file size and save storage space. Digital compression of images may take place in the camera or can be done in the computer with the image editor. When images are stored in JPEG format, compression has already taken place. Both cameras and computer programs allow the user to set the level of compression.

Some compression algorithms, such as those used in PNG file format, are **lossless**, which means no information is lost when the file is saved. By contrast, the JPEG file format uses a **lossy** compression algorithm by which the greater the compression, the more information is lost, ultimately reducing image quality or detail that cannot be restored. JPEG uses knowledge of the way the human brain and eyes perceive color to make this loss of detail less noticeable.

### **Image editor features**

Listed below are some of the most used capabilities of the better graphic manipulation programs. The list is by no means all inclusive. There is a myriad of choices associated with the application of most of these features.

### **Selection**

One of the prerequisites for many of the applications mentioned below is a method of selecting part(s) of an image, thus applying a change selectively without affecting the entire

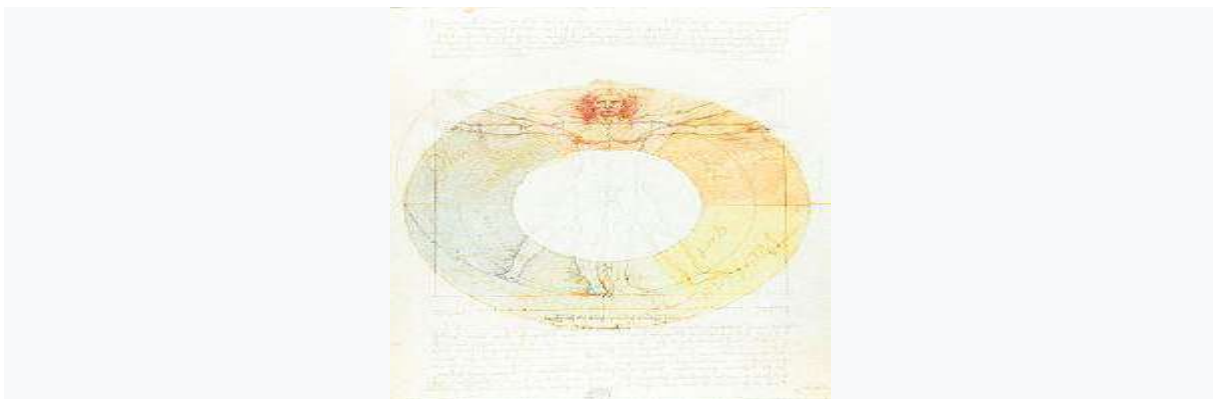
picture. Most graphics programs have several means of accomplishing this, such as:

- a marquee tool for selecting rectangular or other regular polygon-shaped regions,
- a lasso tool for freehand selection of a region,
- a magic wand tool that selects objects or regions in the image defined by proximity of color or luminescence,
- vector-based pen tools,

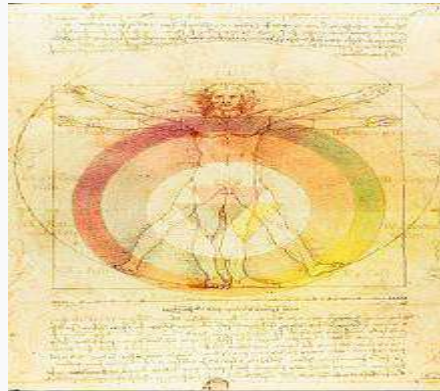
as well as more advanced facilities such as edge detection, masking, alpha compositing, and color and channel-based extraction. The border of a selected area in an image is often animated with the marching ants effect to help the user to distinguish the selection border from the image background.

## **Layers**

Another feature common to many graphics applications is that of Layers, which are analogous to sheets of transparent acetate (each containing separate elements that make up a combined picture), stacked on top of each other, each capable of being individually positioned, altered and blended with the layers below, without affecting any of the elements on the other layers. This is a fundamental workflow which has become the norm for the majority of programs on the market today, and enables maximum flexibility for the user while maintaining non-destructive editing principles and ease of use.



Leonardo da Vinci's Vitruvian Man overlaid with Goethe's Color Wheel using a screen layer in Adobe Photoshop. Screen layers can be helpful in graphic design and in creating multiple exposures in photography.



Leonardo da Vinci's Vitruvian Man overlaid a soft light layer Moses Harris's Color Wheel and a soft light layer of Ignaz Schiffermüller's Color Wheel. Soft light layers have a darker, more translucent look than screen layers.

## **Image Scaling**

Image editors can resize images in a process often called image scaling, making them larger, or smaller. High image resolution cameras can produce large images which are often reduced in size for Internet use. Image editor programs use a mathematical process called resampling to calculate new pixel values whose spacing is larger or smaller than the original pixel values. Images for Internet use are kept small, say 640 x 480 pixels which would equal 0.3 megapixels.

## **Cropping an image**

Digital editors are used to crop images. Cropping creates a new image by selecting a desired rectangular portion from the image being cropped. The unwanted part of the image is discarded. Image cropping does not reduce the resolution of the area cropped. Best results are obtained when the original image has a high resolution. A primary reason for cropping is to improve the image composition in the new image.

---

## **TEXT EDITOR:**

A text editor is a type of computer program that edits plain text. Such programs are sometimes known as “notepad” software, following the naming of Microsoft Notepad. Text editors are provided with operating systems and software development packages, and can be used to change configuration files, documentation files and programming language source code.

### **Plain text vs. rich text**

There are important differences between plain text (created and edited by text editors) and rich text (such as that created by word processors or desktop publishing software).

Plain text exclusively consists of character representation. Each character is represented by a fixed-length sequence of one, two, or four bytes, or as a variable-length sequence of one to four bytes, in accordance to specific character encoding conventions, such as ASCII, ISO/IEC 2022, UTF-8, or Unicode. These conventions define many printable characters, but also non-printing characters that control the flow of the text, such space, line break, and page break. Plain text contains no other information about the text itself, not even the character encoding convention employed. Plain text is stored in text files, although text files do not exclusively store plain text. In the early days of computers, plain text was displayed using a monospace font, such that horizontal alignment and columnar formatting were sometimes done using whitespace characters. For compatibility reasons, this tradition has not changed.

Rich text, on the other hand, may contain metadata, character formatting data (e.g. typeface, size, weight and style), paragraph formatting data (e.g. indentation, alignment, letter and word distribution, and space between lines or other paragraphs), and page specification data (e.g. size, margin and reading direction). Rich text can be very complex. Rich text can be saved in binary format (e.g. DOC), text files adhering to a markup language (e.g. RTF or HTML), or in a hybrid form of both (e.g. Office Open XML).

Text editors are intended to open and save text files containing either plain text or anything that can be interpreted as plain text, including the markup for rich text or the markup for something else (e.g. SVG).

### **Types of text editors**

Some text editors are small and simple, while others offer broad and complex functions. For example, Unix and Unix-like operating systems have the pico editor (or a variant), but many also include the vi and Emacs editors. Microsoft Windows systems come with the simple Notepad, though many people—especially programmers—prefer other editors with more features. Under Apple Macintosh's classic Mac OS there was the native SimpleText, which was replaced in Mac OS X by TextEdit, which combines features of a text editor with those typical of a word processor such as rulers, margins and multiple font selection. These features are not available simultaneously, but must be switched by user command, or through the program automatically determining the file type.

Most word processors can read and write files in plain text format, allowing them to open files saved from text editors. Saving these files from a word processor, however, requires ensuring the file is written in plain text format, and that any text encoding or BOM settings won't obscure the file for its intended use. Non-WYSIWYG word processors, such as WordStar, are more easily pressed into service as text editors, and in fact were commonly used as such during the 1980s. The default file format of these word processors often resembles a markup language, with the basic format being plain text and visual formatting achieved using non-printing control characters or escape sequences. Later word processors like Microsoft Word store their files in a binary format and are almost never used to edit plain text files.

Some text editors can edit unusually large files such as log files or an entire database placed in a single file. Simpler text editors may just read files into the computer's main memory. With larger files, this may be a slow process, and the entire file may not fit. Some text editors do not let the user start editing until this read-in is complete. Editing performance also often suffers in nonspecialized editors, with the editor taking seconds or even minutes to respond to keystrokes or navigation commands. By only storing the visible portion of large files in memory, editing performance improves.

Some editors are programmable, meaning, e.g., they can be customized for specific uses. With a programmable editor it is easy to automate repetitive tasks or, add new functionality or even implement a new application within the framework of the editor. One common motive for customizing is to make a text editor use the commands of another text editor with which the user is more familiar, or to duplicate missing functionality the user has come to depend on. Software developers often use editor customizations tailored to the programming language or development environment they are working in. The programmability of some

text editors is limited to enhancing the core editing functionality of the program, but Emacs can be extended far beyond editing text files—for web browsing, reading email, online chat, managing files or playing games. Emacs can even emulate Vi, its rival in the traditional editor wars of Unix culture.

An important group of programmable editors uses REXX<sup>[a]</sup> as a scripting language. These “orthodox editors” contain a “command line” into which commands and macros can be typed and text lines into which line commands and macros can be typed. Most such editors are derivatives of ISPF/PDF EDIT or of XEDIT, IBM’s flagship editor for VM/SP through z/VM. Among them are THE, KEDIT, X2, Uni-edit, and SEDIT.

A text editor written or customized for a specific use can determine what the user is editing and assist the user, often by completing programming terms and showing tooltips with relevant documentation. Many text editors for software developers include source code syntax highlighting and automatic indentation to make programs easier to read and write. Programming editors often let the user select the name of an include file, function or variable, then jump to its definition. Some also allow for easy navigation back to the original section of code by storing the initial cursor location or by displaying the requested definition in a popup window or temporary buffer. Some editors implement this ability themselves, but often an auxiliary utility like ctags is used to locate the definitions.

### **Typical features**

- Find and replace – Text editors provide extensive facilities for searching and replacing text, either on groups of files or interactively. Advanced editors can use regular expressions to search and edit text or code.
- Cut, copy, and paste – most text editors provide methods to duplicate and move text within the file, or between files.
- Ability to handle UTF-8 encoded text.
- Text formatting – Text editors often provide basic formatting features like line wrap, auto-indentation, bullet list formatting using ASCII characters, comment formatting, syntax highlighting and so on.
- Undo and redo – As with word processors, text editors provide a way to undo and redo the last edit. Often—especially with older text editors—there is only one level of edit history remembered and successively issuing the undo command will only “toggle” the last change. Modern or more complex editors usually provide a multiple level history

such that issuing the undo command repeatedly will revert the document to successively older edits. A separate redo command will cycle the edits "forward" toward the most recent changes. The number of changes remembered depends upon the editor and is often configurable by the user.

- Data transformation – Reading or merging the contents of another text file into the file currently being edited. Some text editors provide a way to insert the output of a command issued to the operating system's shell.
- Filtering – Some advanced text editors allow the editor to send all or sections of the file being edited to another utility and read the result back into the file in place of the lines being "filtered". This, for example, is useful for sorting a series of lines alphabetically or numerically, doing mathematical computations, indenting source code, and so on.
- Syntax highlighting – contextually highlights source code, markup languages, config files and other text that appears in an organized or predictable format. Editors generally allow users to customize the colors or styles used for each language element. Some text editors also allow users to install and use themes to change the look and feel of the editor's entire user interface.
- Extensibility - a text editor intended for use by programmers must provide some plugin mechanism, or be scriptable, so a programmer can customize the editor with features needed to manage individual software projects, customize functionality or key bindings for specific programming languages or version control systems, or conform to specific coding styles.

### **Specialised editors**

Some editors include special features and extra functions, for instance,

- Source code editors are text editors with additional functionality to facilitate the production of source code. These often feature user-programmable syntax highlighting and code navigation functions as well as coding tools or keyboard macros similar to an HTML editor (see below).
- Folding editors. This subclass includes so-called "orthodox editors" that are derivatives of Xedit. Editors that implement folding without programming-specific features are usually called outliners (see below).



- IDEs (integrated development environments) are designed to manage and streamline large programming projects. They are usually only used for programming as they contain many features unnecessary for simple text editing.
- World Wide Web authors are offered a variety of HTML editors dedicated to the task of creating web pages. These include: Dreamweaver, KompoZer and E Text Editor. Many offer the option of viewing a work in progress on a built-in HTML rendering engine or standard web browser. Most web development is done in a dynamic programming language such as Ruby or PHP using a source code editor or IDE. The HTML delivered by all but the simplest static web sites is stored as individual template files that are assembled by the software controlling the site and do not compose a complete HTML document.
- Mathematicians, physicists, and computer scientists often produce articles and books using TeX or LaTeX in plain text files. Such documents are often produced by a standard text editor, but some people use specialized TeX editors.
- Outliners. Also called tree-based editors, because they combine a hierarchical outline tree with a text editor. Folding (see above) can be considered a specialized form of outlining.
- Collaborative editors allow multiple users to work on the same document simultaneously from remote locations over a network. The changes made by individual users are tracked and merged into the document automatically to eliminate the possibility of conflicting edits. These editors also typically include an online chat component for discussion among editors.
- Simultaneous editing is a technique in End-user development research to edit all items in a multiple selection. It allows the user to manipulate all the selected items at once through direct manipulation. The Lapis text editor<sup>[13][14]</sup> and the *multi edit*<sup>[15]</sup> plugin for gedit are examples of this technique. The Lapis editor can also create an automatic multiple selection based on an example item.
- Distraction-free editors provide a minimalistic interface with the purpose of isolating the writer from the rest of the applications and operating system, thus being able to focus on the writing without distractions from interface elements like a toolbar or notification area.

Programmable editors can usually be enhanced to perform any or all of these functions, but simpler editors focus on just one, or, like gPHPedit, are targeted at a single programming language.

## **AUDIO EDITING:**

**Audio editing software** is software which allows editing and generating of audio data. Audio editing software can be implemented completely or partly as library, as computer application, as Web application or as a loadable kernel module. **Wave Editors** are digital audio editors and there are many sources of software available to perform this function. Most can edit music, apply effects and filters, adjust stereo channels etc.

A digital audio workstation (DAW) consists of software to a great degree, and usually is composed of many distinct software suite components, giving access to them through a unified graphical user interface using GTK+, Qt or some other library for the GUI widgets.

### **For use with music**

Editors designed for use with music typically allow the user to do the following:

- The ability to import and export various audio file formats for editing.
- Record audio from one or more inputs and store recordings in the computer's memory as digital audio
- Edit the start time, stop time, and duration of any sound on the audio timeline
- Fade into or out of a clip (e.g. an S-fade out during applause after a performance), or between clips (e.g. crossfading between takes)
- Mix multiple sound sources/tracks, combine them at various volume levels and pan from channel to channel to one or more output tracks
- Apply simple or advanced effects or filters, including compression, expansion, flanging, reverb, audio noise reduction and equalization to change the audio
- Playback sound (often after being mixed) that can be sent to one or more outputs, such as speakers, additional processors, or a recording medium

- Conversion between different audio file formats, or between different sound quality levels

Typically, these tasks can be performed in a manner that is non-linear. Audio editors may process the audio data non-destructively in real-time, or destructively as an "off-line" process, or a hybrid with some real-time effects and some off-line effects.

Comparison of destructive and real-time editing

Destructive editing modifies the data of the original audio file, as opposed to just editing its playback parameters. Destructive editors are also known as "sample editors".

Destructive editing applies edits and processing directly to the audio data, changing the data immediately. If, for example, part of a track is deleted, the "deleted" audio data is immediately removed from that part of the track.

Real-time editing does not apply changes immediately but applies edits and processing on the fly during playback. If, for example, part of a track is deleted, the "deleted" audio data is not actually removed from the track but is hidden and will be skipped on playback.

### **Advantages of destructive editing**

- In graphical editors, all changes to the audio is usually visible immediately as the visible waveform is updated to match the audio data.
- The number of effects that may be applied is virtually unlimited (though may be limited by disk space available for "undo" data).
- Editing is usually precise down to exact sample intervals.
- Effects may be applied to a precisely specified selected region.
- Mixing down or exporting the edited audio is usually relatively quick as little additional processing is required.

### **Limitations of destructive editing**

- Once an effect has been applied, it cannot usually be changed. This is usually mitigated by the ability to "undo" the last performed action. Typically, a destructive audio editor will maintain many levels of "undo history" so that multiple actions may be undone in the reverse order that they were applied.

- Edits can only be undone in the reverse order that they were applied (undoing the most recent edit first).

### **Advantages of real-time editing**

- Effects can usually be adjusted during playback, or at any other time.
- Edits may be undone or adjusted at any time in any order.
- Multiple effects and edits may be 'stacked' so that they are applied to the audio as an effect chain.
- A stack of effects may be changed so that effects are applied in a different order, or effects inserted or removed from the chain.
- Some real-time editors support effect automation so that changes to effect parameters may be programmed to occur at specified times during audio playback.

### **Limitations of real-time editing**

- The waveform does not usually show the effect of processing until the audio has been mixed-down or "bounced" (rendered) to another track.
- The number of effects that may be applied is limited by the available processing power of the computer or editing hardware. In some editors this may be mitigated by "freezing" the track (applying the effect stack destructively).
- It is not usually possible to have an effect only on part of a track. To apply a real-time effect to part of a track usually required that the effect is set to turn on at one point and turn off at another.
- In multi-track editors, if audio is copied or moved from one track to another, the audio in the new track may sound different from how it sounded in the original track as there may be different real-time effects in each track.
- In some applications, mixing down or exporting the edited audio may be slow as all effects and processing needs to be applied.

### **For use with speech**

Editors designed for use in speech research add the ability to make measurements and perform acoustic analyses such as extracting and displaying a fundamental frequency contour or spectrogram. They typically lack most or all of the effects that interest musicians.

## **VIDEO EDITING:**

**Video editing** is the manipulation and arrangement of video shots. Video editing is used to structure and present all video information, including films and television shows, video advertisements and video essays. Video editing has been dramatically democratized in recent years by editing software available for personal computers.

### **Types of editing**

Once the province of expensive machines called video editors, video editing software is now available for personal computers and workstations. Video editing includes cutting segments (trimming), re-sequencing clips, and adding transitions and other Special Effects.

- Linear video editing, using video tape and is edited in a very linear way. Several video clips from different tapes are recorded to one single tape in the order that they will appear.
- Non-linear editing system (NLE), This is edited on computers with specialised software. These are non-destructive to the video being edited and use programs such as Adobe Premiere Pro, Final Cut Pro and Avid.
- Offline editing is the process in which raw footage is copied from an original source, without affecting the original film stock or video tape. Once the editing has been completely edited, the original media is then re-assembled in the online **editing** stage.
- Online editing is the process of reassembling the edit to full resolution video after an offline edit has been performed and is done in the final stage of a video production.

- Vision mixing, when working within live television and video production environments. A vision mixer is used to cut live feed coming from several cameras in real time.

## **Introduction**

Video editing is the process of editing segments of motion video production footage, special effects and sound recordings in the post-production process. Motion picture film editing is a predecessor to video editing and, in several ways, video editing simulates motion picture film editing, in theory and the use of linear video editing and video editing software on non-linear editing systems (NLE). Using video, a director can communicate non-fictional and fictional events. The goals of editing are to manipulate these events to bring the communication closer to the original goal or target. It is a visual art.

Early 1950s video tape recorders (VTR) were so expensive, and the quality degradation caused by copying was so great, that 2 inch Quadruplex videotape was edited by visualizing the recorded track with ferrofluid and cutting with a razor blade or guillotine cutter and splicing with video tape. The two pieces of tape to be joined were painted with a solution of extremely fine iron filings suspended in carbon tetrachloride, a toxic and carcinogenic compound. This “developed” the magnetic tracks, making them visible when viewed through a microscope so that they could be aligned in a splicer designed for this task.

Improvements in quality and economy, and the invention of the flying erase-head, allowed new video and audio material to be recorded over the material already recorded on an existing magnetic tape and was introduced into the linear editing technique. If a scene closer to the beginning of the video tape needed to be changed in length, all later scenes would need to be recorded onto the video tape again in sequence. In addition, sources could be played back simultaneously through a vision mixer (video switcher) to create more complex transitions between scenes. A popular 1970-80s system for doing that was the U-matic equipment (named for the U-shaped tape path). That system used two tape players and one tape recorder, and edits were done by automatically having the machines back up, then speed up together in synchrony, so the edit didn't roll or glitch. Later, 1980-90's came the smaller beta equipment (named for the B-shaped tape path), and more complex controllers, some of which did the synchronizing electronically.

There was a transitional analog period using multiple source videocassette recorder (VCR)s with the EditDroid using LaserDisc players, but modern NLE systems edit video digitally captured onto a hard drive from an analog video or digital video source. Content is ingested

and recorded natively with the appropriate codec that the video editing software uses to process captured footage. High-definition video is becoming more popular and can be readily edited using the same video editing software along with related motion graphics programs. Video clips are arranged on a timeline, music tracks, titles, digital on-screen graphics are added, special effects can be created, and the finished program is “rendered” into a finished video. The video may then be distributed in a variety of ways including DVD, web streaming, QuickTimeMovies, iPod, CD-ROM, or video tape.

### **Home video editing**

Like some other technologies, the cost of video editing has declined by an order of magnitude or more. The original 2” Quadruplex system cost so much that many television production facilities could only afford a single unit and editing was a highly involved process requiring special training. In contrast to this, nearly any home computer sold since the year 2000 has the speed and storage capacity to digitize and edit standard-definition television (SDTV). The two major retail operating systems include basic video editing software – Apple’s iMovie and Microsoft's Windows Movie Maker. Additional options exist, usually as more advanced commercial products. As well as these commercial products, there are opensource video-editing programs. Automatic video editing products have also emerged, opening up video editing to a broader audience of amateurs and reducing the time it takes to edit videos. These exist usually as media storage services, such as Google with its Google Photos or smaller companies like Vidify.

---

## **CODING TECHNIQUES FOR MOVING IMAGES:**

The digital representation of an image requires a very large number of bits. The goal of image coding is to reduce this number, as much as possible, and reconstruct a faithful duplicate of the original picture. Early efforts in image coding, solely guided by information theory, led to a plethora of methods. The compression ratio, starting at 1 with the first digital picture in the early 1960s, reached a saturation level around 10:1 a couple of years ago. This certainly does not mean that the upper bound given by the entropy of the source has also been reached. First, this entropy is not known and depends heavily on the model used for the source, i.e., the digital image. Second, the information theory does not take into account what the human

eye sees and how it sees. Recent progress in the study of the brain mechanism of vision has opened new vistas in picture coding. Directional sensitivity of the neurons in the visual pathway combined with the separate processing of contours and textures has led to a new class of coding methods capable of achieving compression ratios as high as 70:1. Image quality, of course, remains as an important problem to be investigated. This class of methods, that we call second generation, is the subject of this paper. Two groups can be formed in this class: methods using local operators and combining their output in a suitable way and methods using contour-texture descriptions. Four methods, two in each class, are described in detail. They are applied to the same set of original pictures to allow a fair comparison of the quality in the decoded pictures. If more effort is devoted to this subject, a compression ratio of 100:1 is within reach.

The MPEG standards consist of different *Parts*. Each *part* covers a certain aspect of the whole specification. The standards also specify *Profiles* and *Levels*. *Profiles* are intended to define a set of tools that are available, and *Levels* define the range of appropriate values for the properties associated with them. Some of the approved MPEG standards were revised by later amendments and/or new editions. MPEG has standardized the following compression formats and ancillary standards:

- MPEG-1 (1993): *Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s* (ISO/IEC 11172). This initial version is known as a lossy fileformat and is the first MPEG compression standard for audio and video. It is commonly limited to about 1.5 Mbit/s although the specification is capable of much higher bit rates. It was basically designed to allow moving pictures and sound to be encoded into the bitrate of a Compact Disc. It is used on Video CD and can be used for low-quality video on DVD Video. It was used in digital satellite/cable TV services before MPEG-2 became widespread. To meet the low bit requirement, MPEG-1 downsamples the images, as well as uses picture rates of only 24–30 Hz, resulting in a moderate quality. It includes the popular MPEG-1 Audio Layer III (MP3) audio compression format.
- MPEG-2 (1995): *Generic coding of moving pictures and associated audio information* (ISO/IEC 13818). Transport, video and audio standards for broadcast-quality television. MPEG-2 standard was considerably broader in scope and of wider appeal – supporting interlacing and high definition. MPEG-2 is considered important because it



has been chosen as the compression scheme for over-the-air digital television ATSC, DVB and ISDB, digital satellite TV services like Dish Network, digital cable television signals, SVCD and DVD Video. It is also used on Blu-ray Discs, but these normally use MPEG-4 Part 10 or SMPTE VC-1 for high-definition content.

- **MPEG-3:** MPEG-3 dealt with standardizing scalable and multi-resolution compression and was intended for HDTV compression but was found to be redundant and was merged with MPEG-2; as a result, there is no MPEG-3 standard. MPEG-3 is not to be confused with MP3, which is MPEG-1 or MPEG-2 Audio Layer III.
- **MPEG-4 (1998):** *Coding of audio-visual objects.* (ISO/IEC 14496) MPEG-4 uses further coding tools with additional complexity to achieve higher compression factors than MPEG-2. In addition to more efficient coding of video, MPEG-4 moves closer to computer graphics applications. In more complex profiles, the MPEG-4 decoder effectively becomes a rendering processor and the compressed bitstream describes three-dimensional shapes and surface texture. MPEG-4 supports Intellectual Property Management and Protection (IPMP), which provides the facility to use proprietary technologies to manage and protect content like digital rights management. It also supports MPEG-J, a fully programmatic solution for creation of custom interactive multimedia applications (Java application environment with a Java API) and many other features

## **User Interface Design Basics:**

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

### **Choosing Interface Elements**

Users have become familiar with interface elements acting in a certain way, so try to be consistent and predictable in your choices and their layout. Doing so will help with task completion, efficiency, and satisfaction.

Interface elements include but are not limited to:

- **Input Controls:** buttons, text fields, checkboxes, radio buttons, dropdown lists, list boxes, toggles, date field

- **Navigational Components:** breadcrumb, slider, search field, pagination, slider, tags, icons
- **Informational Components:** tooltips, icons, progress bar, notifications, message boxes, modal windows
- **Containers:** accordion

There are times when multiple elements might be appropriate for displaying content. When this happens, it's important to consider the trade-offs. For example, sometimes elements that can help save you space, put more of a burden on the user mentally by forcing them to guess what is within the dropdown or what the element might be.

### **Best Practices for Designing an Interface**

Everything stems from knowing your users, including understanding their goals, skills, preferences, and tendencies. Once you know about your user, make sure to consider the following when designing your interface:

- **Keep the interface simple.** The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use on labels and in messaging.
- **Create consistency and use common UI elements.** By using common elements in your UI, users feel more comfortable and are able to get things done more quickly. It is also important to create patterns in language, layout and design throughout the site to help facilitate efficiency. Once a user learns how to do something, they should be able to transfer that skill to other parts of the site.
- **Be purposeful in page layout.** Consider the spatial relationships between items on the page and structure the page based on importance. Careful placement of items can help draw attention to the most important pieces of information and can aid scanning and readability.
- **Strategically use color and texture.** You can direct attention toward or redirect attention away from items using color, light, contrast, and texture to your advantage.
- **Use typography to create hierarchy and clarity.** Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text to help increase scalability, legibility and readability.
- **Make sure that the system communicates what's happening.** Always inform your users of location, actions, changes in state, or errors. The use of various UI elements to communicate status and, if necessary, next steps can reduce frustration for your user.

- **Think about the defaults.** By carefully thinking about and anticipating the goals people bring to your site, you can create defaults that reduce the burden on the user. This becomes particularly important when it comes to form design where you might have an opportunity to have some fields pre-chosen or filled out.
- 

## QR code (quick response code)

A QR code (quick response code) is a type of 2D bar code that is used to provide easy access to information through a smartphone.



*Example of a QR code*

In this process, known as mobile tagging, the smartphone's owner points the phone at a QR code and opens a barcode reader app which works in conjunction with the phone's camera. The reader interprets the code, which typically contains a call to action such as an invitation to download a mobile application, a link to view a video or an SMS message inviting the viewer to respond to a poll. The phone's owner can choose to act upon the call to action or click cancel and ignore the invitation.

Static QR codes, the most common type, are used to disseminate information to the general public. They are often displayed in advertising materials in the environment (such as billboards and posters), on television and in newspapers and magazines. The code's creator can track information about the number of times a code was scanned and its associated action taken, along with the times of scans and the operating system of the devices that scanned it.

Dynamic QR codes (sometimes referred to as unique QR codes) offer more functionality. The owner can edit the code at any time and can target a specific individual for personalized marketing. Such codes can track more specific information, including the scanners names and

email address, how many times they scanned the code and, in conjunction with tracking codes on a website, conversion rates.

The technology for QR codes was developed by Densa-Wave, a Toyota subsidiary. The codes were originally used for tracking inventory.

**Here are a few examples of QR codes in current use:**

- QR codes on business cards link to the individual's full resume or website.
- A Starbucks promotion featured a QR code-enabled scavenger hunt involving hints accessed through QR codes in the stores.
- Quiring Monuments in Seattle puts QR code on gravestones to connects people to an online obituary or a website about the deceased.
- In Florida, the J.N. "Ding" Darling National Wildlife Refuge puts QR codes on signs to connect people to informational videos about wildlife along the trails.

**Barcode reader (POS scanner, bar code reader, price scanner)**

A barcode reader, also called a price scanner or point-of-sale ( POS ) scanner, is a hand-held or stationary input device used to capture and read information contained in a bar code. A barcode reader consists of a scanner , a decoder (either built-in or external), and a cableused to connect the reader with a computer. Because a barcode reader merely captures and translates the barcode into numbers and/or letters, the data must be sent to a computer so that a software application can make sense of the data. Barcode scanners can be connected to a computer through a serial port , keyboard port , or an interface device called a wedge . A barcode reader works by directing a beam of light across the bar code and measuring the amount of light that is reflected back. (The dark bars on a barcode reflect less light than the white spaces between them.) The scanner converts the light energy into electrical energy, which is then converted into data by the decoder and forwarded to a computer.

There are five basic kinds of barcode readers -- pen wands, slot scanners, Charge-Couple Device ( CCD ) scanners, image scanners, and laser scanners.

- A pen wand is the simplest barcode reader. It contains no moving parts and is known for its durability and low cost. A pen wand can present a challenge to the user, however, because it has to remain in direct contact with the bar code, must be held at a certain angle, and has to be moved over the bar code at a certain speed.
  - A slot scanner remains stationary and the item with the bar code on it is pulled by hand through the slot. Slot scanners are typically used to scan bar codes on identification cards.
  - A CCD scanner has a better read-range than the pen wand and is often used in retail sales. Typically, a CCD scanner has a "gun" type interface and has to be held no more than one inch from the bar code. Each time the bar code is scanned, several readings are taken to reduce the possibility of errors. A disadvantage of the CCD scanner is that it cannot read a bar code that is wider than its input face.
  - An image scanner, also called a camera reader, uses a small video camera to capture an image of the bar code and then uses sophisticated digital image processing techniques to decode the bar code. It can read a bar code from about 3 to 9 inches away and generally costs less than a laser scanner.
  - A laser scanner, either hand-held or stationary, does not have to be close to the bar code in order to do its job. It uses a system of mirrors and lenses to allow the scanner to read the bar code regardless of orientation and can easily read a bar code up to 24 inches away. To reduce the possibility of errors, a laser scanning may perform up to 500 scans per second. Specialized long-range laser scanners are capable of reading a bar code up to 30 feet away.
-

# UNIT – 3: USING IMAGE PROCESSING TOOLS

## Photoshop workshop

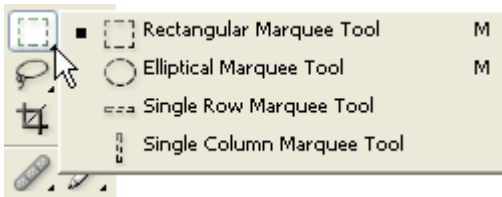
**Adobe Photoshop** is the predominant photo editing and manipulation software on the market. Its uses range from full featured editing of large batches of photos to creating intricate digital paintings and drawings that mimic those done by hand.

Image editing tools

### Selection and move Tools

**Adobe Photoshop** also offers a number of **selection tools**: Rectangular marquee, Elliptical marquee, Lasso, Polygonal Lasso, Magnetic Lasso, Magic Wand.


The **Rectangular marquee** and **Elliptical marquee tools** are hidden in the Toolbox under one and the same icon. The icon on the Toolbox displays the last tool used. To open the floating menu right-click on the arrow in the lower right corner of the displayed icon.



- **Rectangular marquee**

This tool selects rectangular and square areas.

To select a rectangular area, you should:

**Step 1.** Activate the Rectangular marquee tool by clicking on the icon , or (if the Rectangular marquee was not the last tool applied) select it from the floating window.

**Step 2.** Bring the mouse cursor to the point of the image where the corner of an imaginary rectangle should be and press the left mouse button.


**Step 3.** Keeping the left button pressed, move the cursor diagonally to the opposite corner and release the button.

To select a square area of the image, make a selection keeping the Shift key pressed. Take into account that if you already have a selected area the new selection will be added to the previous one. To avoid it you should press the Shift key only when you start selecting a new area.

- **Elliptical marquee**

This tool selects ellipses and circles.

To select an elliptical area, you should:

**Step 1.** Select the Elliptical marquee tool from the Toolbox by clicking on the icon , or (if the Elliptical marquee was not the last tool applied) select it from the floating window.

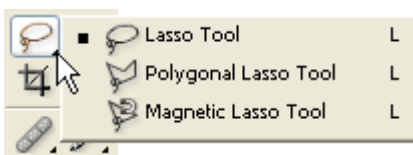
**Step 2.** Bring the mouse cursor to the point of the image where the corner of an imaginary rectangle with an inscribed ellipse should be and press the left button.

**Step 3.** Keeping the left button pressed, move the cursor diagonally to the opposite corner and release the button.

To select a circular area of the image make a selection keeping the Shift key pressed. Take into account that if you already have a selected area the new selection will be added to the previous one. To avoid it you should press the Shift key only when you start selecting a new area.

If you keep the Alt (Option in Mac) key pressed when selecting an elliptical or a rectangular area, the selection is generated from the center to borders, not from one corner to another.


The **Lasso, Polygonal Lasso, Magnetic Lasso tools** are hidden in the Toolbox under one and the same icon. The icon on the Toolbox displays the last tool selected. To open the floating menu right-click on the arrow in the lower right corner of the displayed icon.



- **Lasso**

The tool allows creating freehand selections.

To make a freehand selection you should:

**Step 1.** Select the Lasso tool from the Toolbox by left-clicking on the icon , or (if Lasso was not the last tool applied) select it from the floating window.


**Step 2.** Bring the mouse cursor to the object that must be selected and outline it keeping the left button pressed.

- **Polygonal Lasso**

The tool makes freehand selections, but its contour is made up of straight segments.

To make a selection you should:



**Step 1.** Select the Polygonal Lasso tool from the Toolbox by clicking on the icon , or (if Polygonal Lasso was not the last tool applied) select it from the floating window.

**Step 2.** Bring the cursor to any point near the object to be outlined and press the left mouse button - it'll be the first point of the contour.

**Step 3.** Move the cursor to the next point of the contour not far from the first one and left-click it again. The program will automatically draw a straight line between the two points.


**Step 4.** Keep putting points in this way until the whole object is outlined and close the contour.

- **Magnetic Lasso**

This tool makes a freehand selection.

When you use Magnetic Lasso, you do not need to follow the contour of the object precisely. If the object stands out against the background the border of the selected area will be traced automatically as you move the cursor along the object.

To select an area using Magnetic lasso you should:

**Step 1.** Select the Magnetic Lasso tool from the Toolbox by clicking on the icon , or (if Magnetic Lasso was not the last tool applied) select it from the floating window.

**Step 2.** Bring the mouse cursor to the border of the object that should be selected.


**Step 3.** Press the left button and start dragging the cursor along the object. Pay attention to fastening points that appear as you outline the object and when you make a click. If a fastening point is irrelevant you can remove it by pressing the Delete key and return to the previous fastening point to continue outlining the object.

**Step 4.** Close the contour, that is join the first fastening point with the last one by bringing the cursor to the first point or by making a double-click.

- **Magic Wand**

This tool selects a consistently colored area. You can set Tolerance in the Options palette of the Magic Wand tool. The higher is the value, the more colors will fall into the selected area. The Tolerance value ranges from 0 to 255. At Tolerance equal to 0 the selected area will be represented only by one color, at Tolerance equal to 255 - all colors of the image will be selected, that is the whole image.

To select a consistently colored area, you should:

**Step 1.** Select the Magic Wand tool in the Toolbox by clicking the icon .

**Step 2.** Bring the cursor to the pixel of the image that must be included into the selection and left-click it. As a result an outline appears around the pixel. It includes colors of the image similar to the color of the selected pixel according to the specified Tolerance value.

These selection tools are efficient due to the flexibility of their usage: **you can add to, subtract from or intersect a selection.**

To add an area to the previous selection you should press the Shift key before you use a selection tool and, keeping it pressed, make a new selection.

To subtract an area from the previous selection you should press the Alt (Option in Mac) key before you use a selection tool and, keeping it pressed, make a new selection.

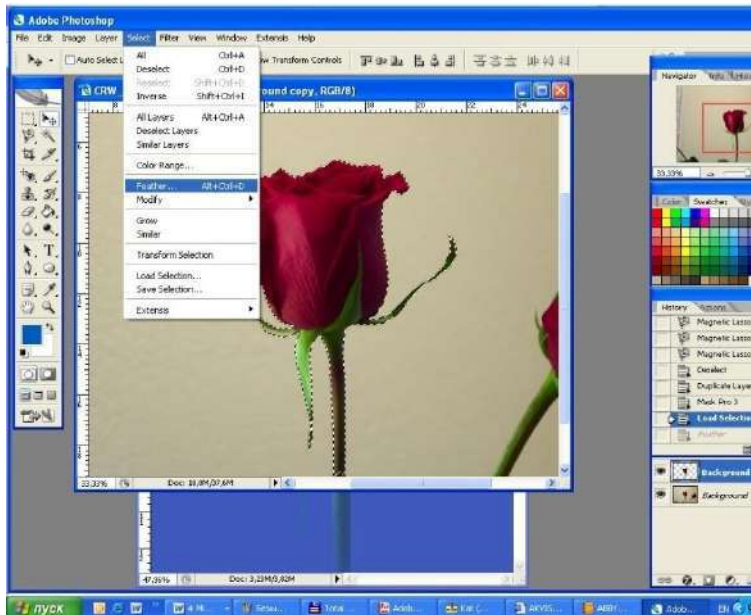
If you press Shift and Alt (Shift and Option in Mac) keys simultaneously you obtain an intersection of the old and new selections.

### **Selection Shading**

Whether you've made a selection with Photoshop tools or used a plugin you'd want your selection to look nice and smooth on a new background. This happens when the edges of the object blend with the background colors. If the edges of the selected image were not softened or feathered during the process of selection you may shade them in Photoshop.

All you have to do is:

- **Step 1.** In Adobe Photoshop have your selection loaded. Open **Select** menu and choose **Feather**.



- **Step 2.** In the opening dialogue window type in a number of pixels to be feathered (2-5 are usually enough).



- **Step 3.** Now you can copy your selection, paste it into a new environment and evaluate the result.



No shading



5 px shading



15 px shading



50 px shading

- **Moving an object in Adobe Photoshop**

To move an object, we can either move the layer with the inserted object or select a layer fragment and move this immediate fragment.

Select the **Move**  tool from the Toolbar to move the layer or its fragment.

First of all, activate the layer with the inserted object and follow the instructions:

- **Step 1.** If you want to move a fragment of the layer not the layer itself, you can select the necessary area using any selection tool.

- **Step 2.** Select the Move tool from the Toolbar.
  - **Step 3.** Bring the cursor inside the selected fragment to move it or on any point of the layer to move the layer itself.
  - **Step 4.** Drag the object. For this purpose, press the left mouse button and keeping it pressed drag the mouse cursor.
- **Transformation of objects in Adobe Photoshop photo editor**

After an object is inserted in a new layer, we can use Layer transformation commands to transform the object. To transform the layer or the selected fragment we can use one of the commands from the menu **Edit - Transform**. For example, the following commands: **Scale, Rotate, Skew, Distort, Perspective, rotate 180°, Rotate 90° CW, rotate 90° CCW, Flip Horizontal, Flip Vertical**.

You can also use the command **Free Transform** from the Edit menu, which will help you to perform all transformations at one time. If you select the command Edit – Free Transform, the layer or the selected fragment will be enclosed in a frame with eight markers, and in the Options palette a number of parameters for adjustment will appear.



<="" div=""

**Scale** – to adjust the scale of the image within the area you should move one of the eight markers. To change the scale proportionally you should drag the marker in the corner keeping the Shift key pressed. You can enter values for the W and H parameters in percentage from the original size directly in the Options palette. For the size to be changed proportionally, you should activate the relation sign between the W and H parameters.


**Flip** – to flip an image you should move one of the markers behind the opposite marker. For example, if you drag the left marker all the way to the right side of the right marker, the image will flip horizontally. However, if all you want is to flip the image, you'd better use the commands Flip Horizontal and Flip Vertical from the menu Edit – Transform.


**Rotate** – to rotate an image you should bring the cursor to the marker in the corner so that the cursor transforms in a two-side rounded arrow, press the left mouse and keeping it pressed, drag the cursor. You can set the rotation angle in the Options palette using the Rotate parameter.

**Skew** – to skew an image you should drag the marker on the side, the upper and the lower marker keeping the Ctrl key pressed (Command in Mac). You can adjust the Skew transformation option in the Options palette setting the H and V parameters.

**Distort** – it is possible to distort an image by dragging a corner marker keeping the Ctrl key pressed (Cmd in Mac).

**Perspective** – to create a perspective you should drag a corner marker keeping the Ctrl and Shift keys pressed (Cmd and Shift in Mac). If you want to drag two points at a time, you should drag a corner marker keeping the Ctrl, Alt and Shift keys pressed (Cmd, Option and Shift in Mac).

To confirm the transformation press, enter (Return in Mac) or double-click with the left mouse button inside the object. You can as well press the button  in the **Options** palette.


To cancel the transformation press Esc or the button  in the **Options** palette.

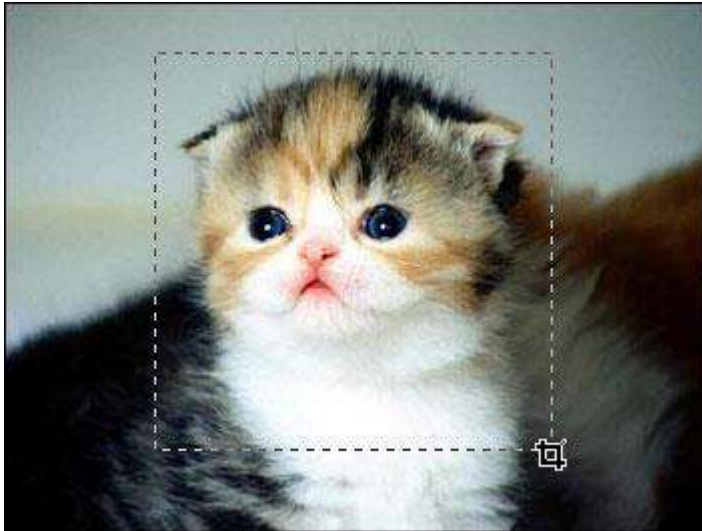
## **Cropping an Image in Adobe Photoshop**

**Cropping** - This changes the number of pixels in an image by "cropping" away the pixels from the surrounding area.

In the photo editor **Adobe Photoshop** an image can be cropped with the **Crop** tool or the **Crop** command.

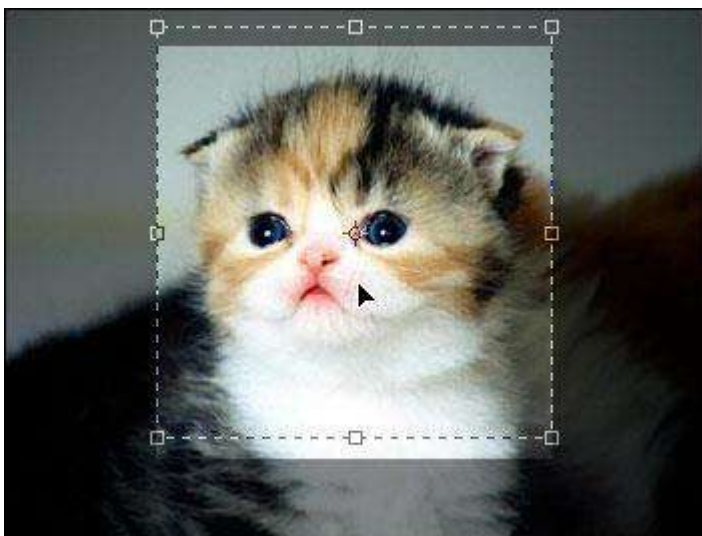
To crop an image with the **Crop** tool, follow these instructions:

- **Step 1.** Choose the **Crop** tool  from the **Tool Panel** or press C.
- **Step 2.** Bring the cursor to a point on the image, where a corner of the cropped image will be, and left-click the mouse.
- **Step 3.** Moving the cursor diagonally, keep the left mouse button pressed.



**Step 4.** Release the left mouse button. A box will appear over the image with marked corners, indicating the area that will be preserved. This box can be moved, resized, and rotated.

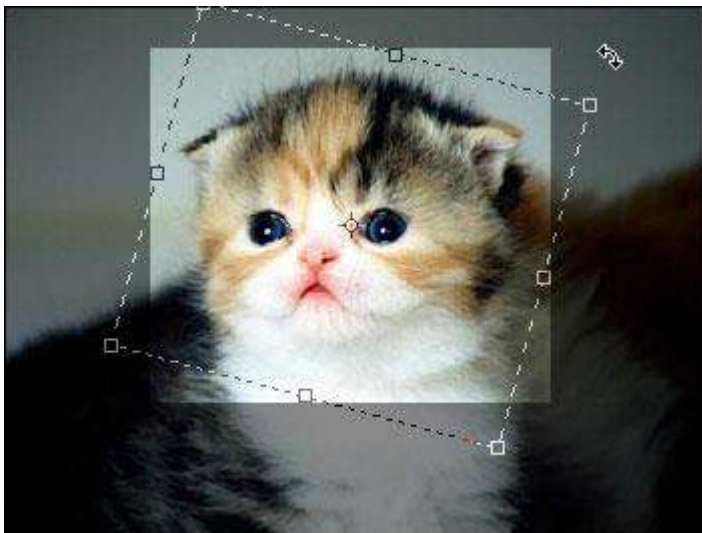
- To move the crop box, move the cursor completely inside the selected area, press the left mouse button, and while keeping the left mouse button pressed, drag the box.





- To change the size of the selected area move the cursor to one of the corner markers, press the left mouse button and drag the marker. If the cursor is dragged while pressing Shift, the size of the box will be changed proportionally.



- To rotate the crop box move the cursor to one of the corner markers and drag the cursor.



- **Star 5.** Press Enter (Return on Mac) or press the  button in the Options Panel to crop the image.

To close the crop box without cropping the image press Esc or press the  button in the Options Panel.

To crop an image with the **Crop** command from the Photoshop menu, follow these steps:

- **Step 1.** Choose the **Rectangular marquee** tool from the Tool Panel.
- **Step 2.** Select a rectangular area on the image **Selection Tools in Adobe Photoshop**).

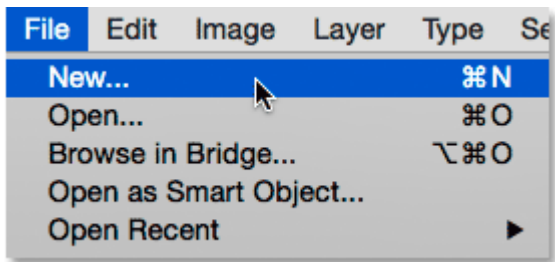


- **Step 3.** Choose **Crop** from the **Image** menu.

Drawing Gradients in Photoshop

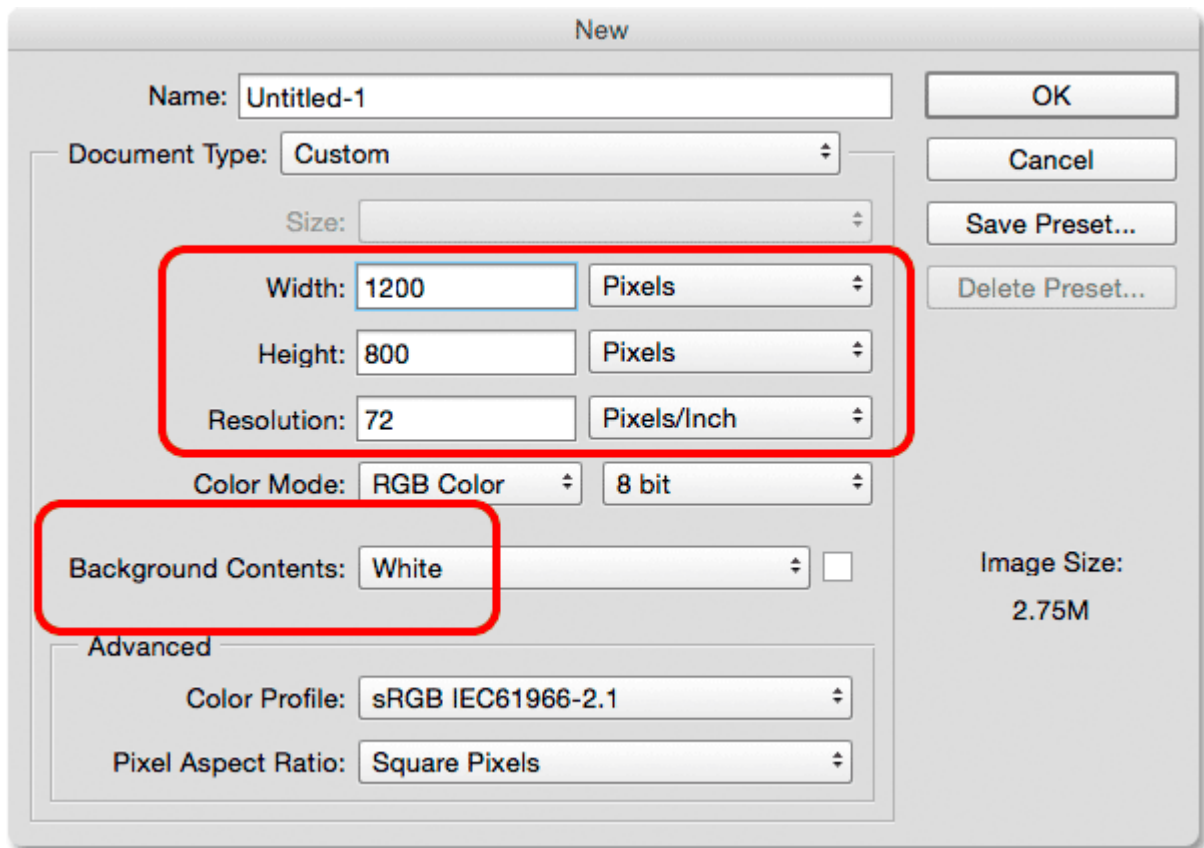
## Creating A New Document

Let's start by creating a new Photoshop document. To do that, I'll go up to the **File** menu in the Menu Bar along the top of the screen and choose **New**:



Going to **File > New**.

This opens the New dialog box. For this tutorial, I'll set the **Width** of my document to **1200 pixels** and the **Height** to **800 pixels**. There's no particular reason why I'm using this size, so if you're working along with me and have a different size in mind, feel free to use it. I'll leave the **Resolution** value set to its default of **72 pixels/inch**, and I'll make sure **Background Contents** is set to **White**. I'll click **OK** when I'm done to close out of the dialog box, at which point a new white-filled document appears on the screen:



The New dialog box.

### Selecting the Gradient Tool

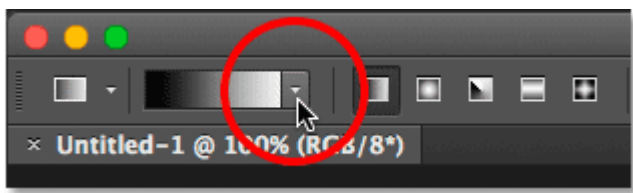
Photoshop's **Gradient Tool** is found in the **Tools panel** along the left of the screen. I'll select it by clicking on its icon. You can also select the Gradient Tool simply by pressing the letter **G** on your keyboard:



Selecting the Gradient Tool from the Tools panel.

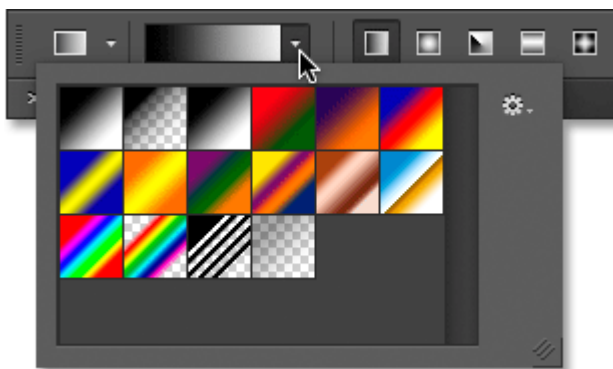
### The Gradient Picker

With the Gradient Tool selected, the next thing we need to do is choose a gradient, and there's a couple of ways we can do that. One is by opening Photoshop's **Gradient Picker**; the other is by opening the larger **Gradient Editor**. The difference between the two is that the Gradient Picker simply allows us to choose from ready-made preset gradients, while the Gradient Editor, as its name implies, is where we can edit and customize our own gradients. When you just want to choose one of Photoshop's preset gradients, or one that you've previously created on your own and saved as a custom preset (again, we'll learn how to do that in the next tutorial), click on the small **arrow** to the right of the **gradient preview bar** in the Options Bar. Make sure you click on the arrow itself, *not* on the preview bar (clicking the preview bar will open the Gradient Editor and we don't want to go there just yet):



**Clicking the arrow to open the Gradient Picker.**

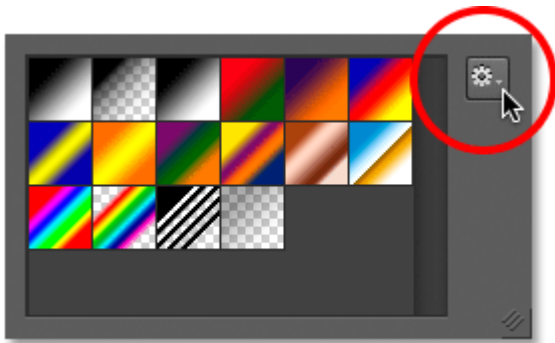
Clicking the arrow opens the Gradient Picker, with thumbnails of all the preset gradients we can choose from. To choose a gradient, click on its thumbnail, then press **Enter** (Win) / **Return** (Mac) on your keyboard, or click on any empty space in the Options Bar, to close the Gradient Picker. You can also **double-click** on the thumbnail, which will both select the gradient and close out of the Gradient Picker:



**The Gradient Picker.**

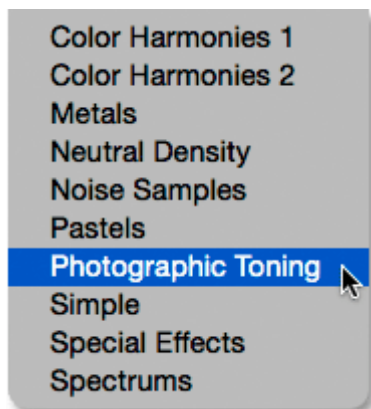
## Loading More Gradients

By default, only a small number of preset gradients are available, but Photoshop includes other **gradient sets** that we can choose from. All we need to do is load them in. To do that, click on the **gear icon** in the upper right:



Clicking the gear icon in the Gradient Picker.

If you look in the bottom half of the menu that appears, you'll find the list of additional gradient sets, each based on a specific theme, like color harmonies, metals, pastels, and more. If you're a photographer, the Neutral Density and Photographic Toning gradients are particularly useful:



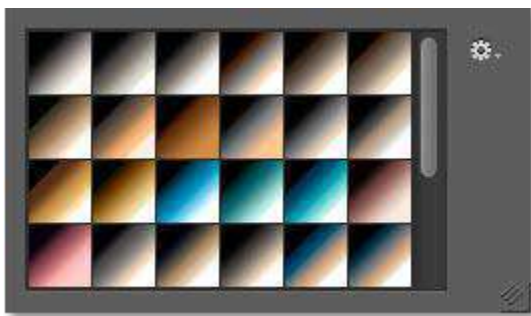
The other gradient sets we can choose from.

To load any of the sets, click on its name in the list. I clicked on the Photographic Toning set. Photoshop will ask if you want to replace the current gradients with the new ones. If you click **Append**, rather than replacing the original gradients, it will simply add the new ones below the originals. As we'll see in a moment, it's easy to restore the originals, so I'll click **OK** to replace them with the Photographic Toning set:



Clicking OK to replace the original gradients with the new set.

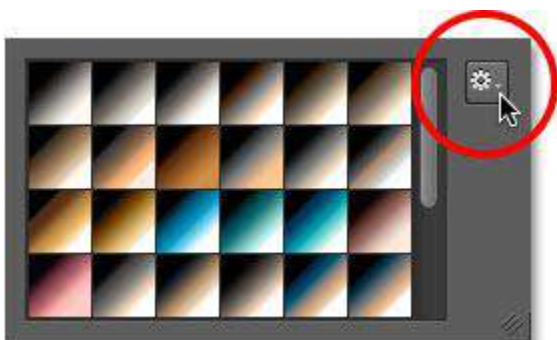
And now, we see in the Gradient Picker that the original gradients have been replaced with the Photographic Toning gradients.



The original gradients have been replaced with the new set.

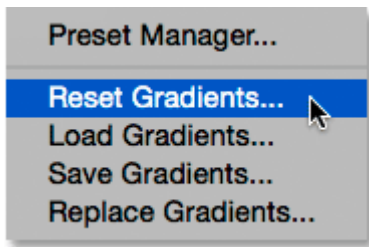
### Restoring the Default Gradients

To keep us focused on the basics, we'll stick with the original default gradients for now. To restore them, click once again on the **gear icon** in the Gradient Picker:



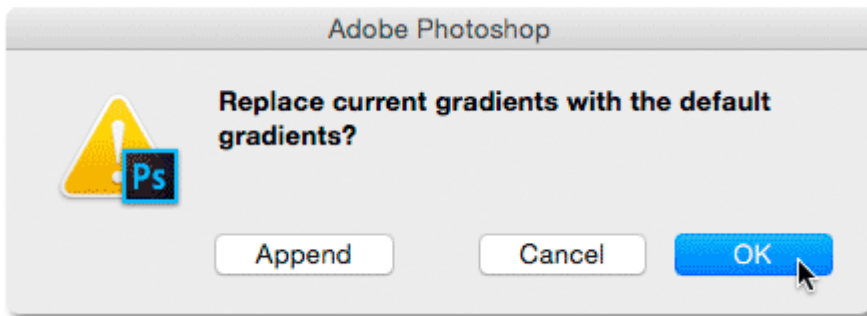
Clicking the gear icon.

Then choose **Reset Gradients** from the menu:



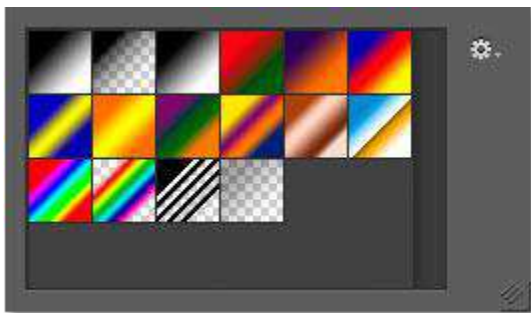
Choosing "Reset Gradients".

Photoshop will ask if you want to replace the current gradients with the defaults. Click **OK**:



Replacing the current gradients with the defaults.

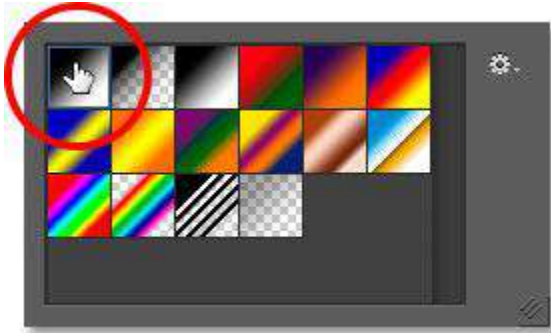
And now, we're back to the originals:



The default gradients have been restored.

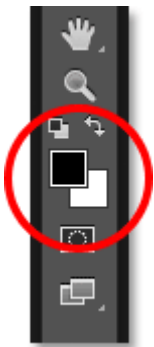
### The Foreground To Background Gradient

Before we learn how to draw gradients, let's quickly look at one gradient in particular - the **Foreground to Background** gradient. It's the one that Photoshop selects for us by default, but you can also select it manually if you need to by clicking on its thumbnail (first one on the left, top row):



### Selecting the Foreground to Background gradient.

As you may have guessed, the Foreground to Background gradient gets its colors from your Foreground and Background colors. You can see your current Foreground and Background colors in the **color swatches** near the bottom of the Tools panel. The swatch in the **upper left** shows the **Foreground** color, and the one in the **lower right** shows the **Background** color. By default, the Foreground color is set to **black** and the Background color is set to **white**:



### The current Foreground (upper left) and Background (lower right) colors.

Since it's based on your current Foreground and Background colors, the Foreground to Background gradient is the easiest of all the gradients to customize and the one that often proves most useful. Let's use it to help us learn how to actually draw a gradient, and along the way, we'll see how easy it is to change its colors to whatever we need!

### Specifying and Adjusting Colors color model

Probably everyone who is engaged in advertising, had to hear such phrases as “light model”, “print file should be in CMYK, and for posting on the site – RGB”. Some may even know about the existence of such color models as GreyScale, LAB, HSB and HLS. But what exactly are these “Color models”? How is CMYK color model different from RGB or LAB?

We live in this white light. And this light can be divided into many different hues. As far as we know, the first who came up with this idea was Isaac Newton. He divided the light through the prism to seven primary colors: red, orange, yellow, green, blue, indigo and violet. We'll talk about this phenomenon afterward. But now we divide the light into three basic hue (color), because it is convenient.

### RGB color model

Each color TV or monitor of your computer is based on the principle of the division of the light. If say roughly, the monitor on which you now see is a huge number of points (their number determines the horizontal and vertical resolution of the monitor), and each point of light has three "bulbs": red, green and blue. Each "bulb" can shine with different brightness and cannot shine at all. If only shines blue "bulb" – we see the blue dot. If only the red – we can see the red dot. Similarly, with green. If all the lights are shining with full brightness at one point, then this point turns white, as all gradations of white come together again. If no light shines, the point seems to us black. Since black – the absence of light. Combining the colors of these "bulbs", glowing with different brightness, we can obtain different colors and hues.

Brightness of each bulb is determined by the intensity (division) from 0 ("bulb" is off) to 255 ("bulb", that luminous with full power). This division is called – RGB color model, from the first letters of the words "RED", "GREEN" and "BLUE".

Thus, **white color** of our dot in RGB color model can be written as:

**R – 255, G – 255, B – 255**

"Saturated" red can be written as:

**R – 255, G – 0, B – 0**

Black:

**R – 0, G – 0, B – 0**

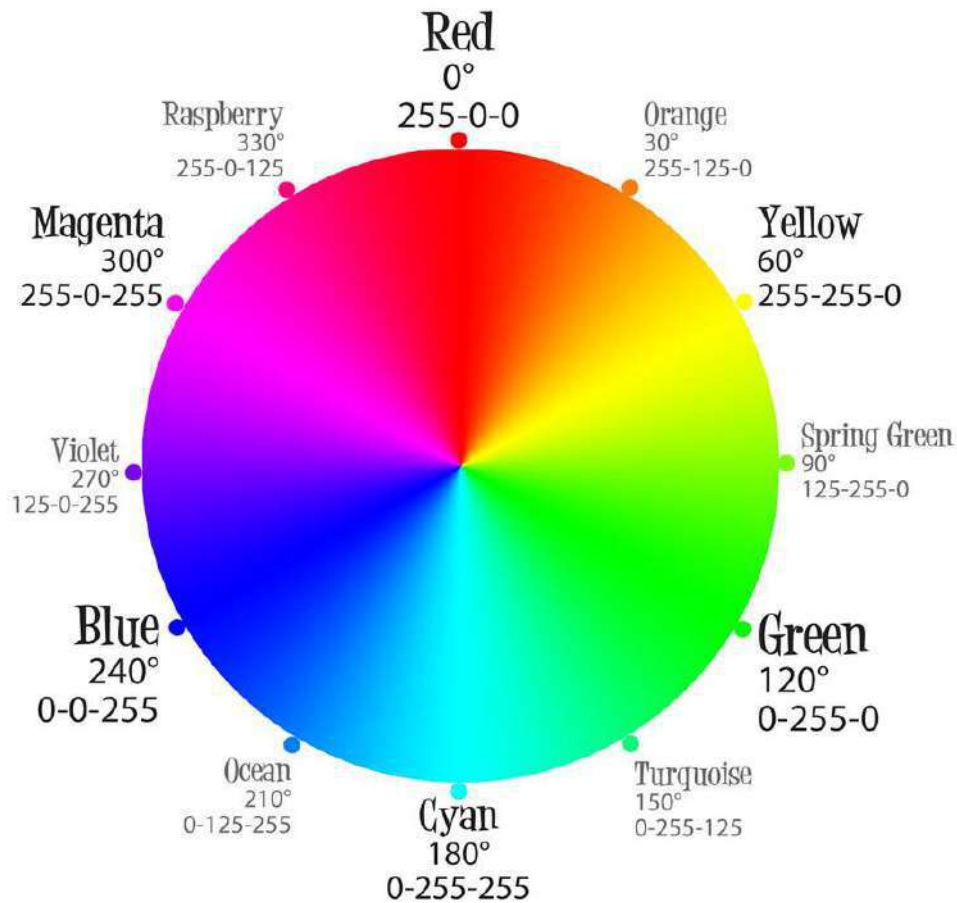
Yellow will be the following:

**R – 255, G – 255, B – 0**



You also need to know, that in order to record colors in rgb, we usually use the hexadecimal system. Intensity indicators are recorded in order of #RGB:

**White – #ffffff, Red – #ff0000, Black – #000000, Yellow – #ffff00**



### CMYK color model

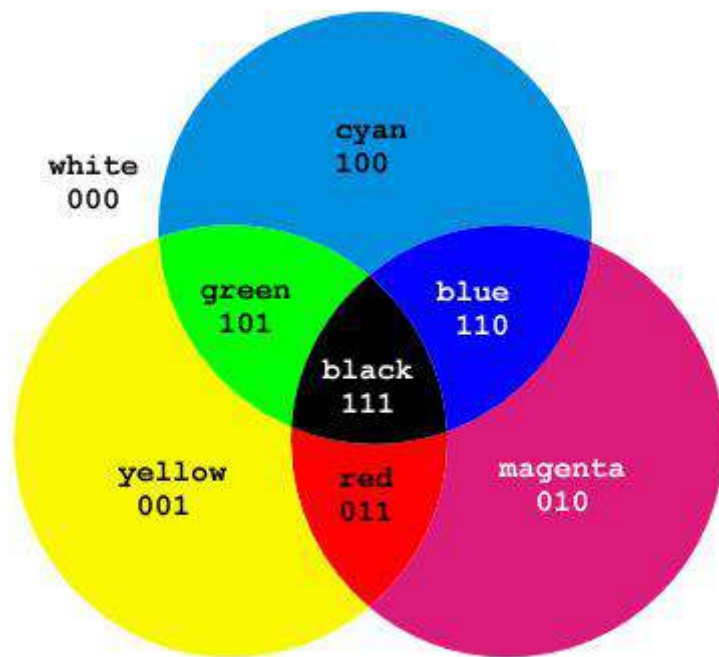
So now we know the tricky way our computer sends us the color of a particular point. Let's now use the acquired knowledge and try to get the white color with paints. Let's buy three jars of paints: red, blue and green, and mix them. Did it work? I did not get. What is the problem?

The problem is that our monitor emits light, so the color is lit. But the nature of many of the objects do not have this property. They simply reflect the white light that falls on them. And if the subject reflects the entire spectrum of white light, we see it in white, but if some of that light is absorbed by them – we see it is not all white.

Something like this: we shine on the red thing with white light. White light can be represented as R-255 G-255 B-255. But the thing does not want to reflect all the light that we

have sent to it, and it is brazenly stealing from us all hues of green and blue. As a result, it reflects only the R-255 G-0 B-0. That is why it seems to us as red.

So, it is very problematic to use the RGB color model for printing on paper. For printing, as a rule, we use another color model – CMY or CMYK. CMY color model is based on the fact that initially we have a white sheet of paper, and it reflects (virtually) the entire spectrum of RGB, and all inks applied to it, act as filters, each of which “steals” own color (red, or green, or blue). Thus, these color inks are determined by subtracting one from the white colors RGB. We get colors: Cyan (light blue), Magenta (or we can say pink) and Yellow.



As you remember, gradation of each color (in RGB color model) is brightness (from 0 to 255). However, in CMYK color model, the value of each color – is “opacity” (amount of paint) and determines the percentage from 0% to 100%.

Thus, white color can be described as follows:

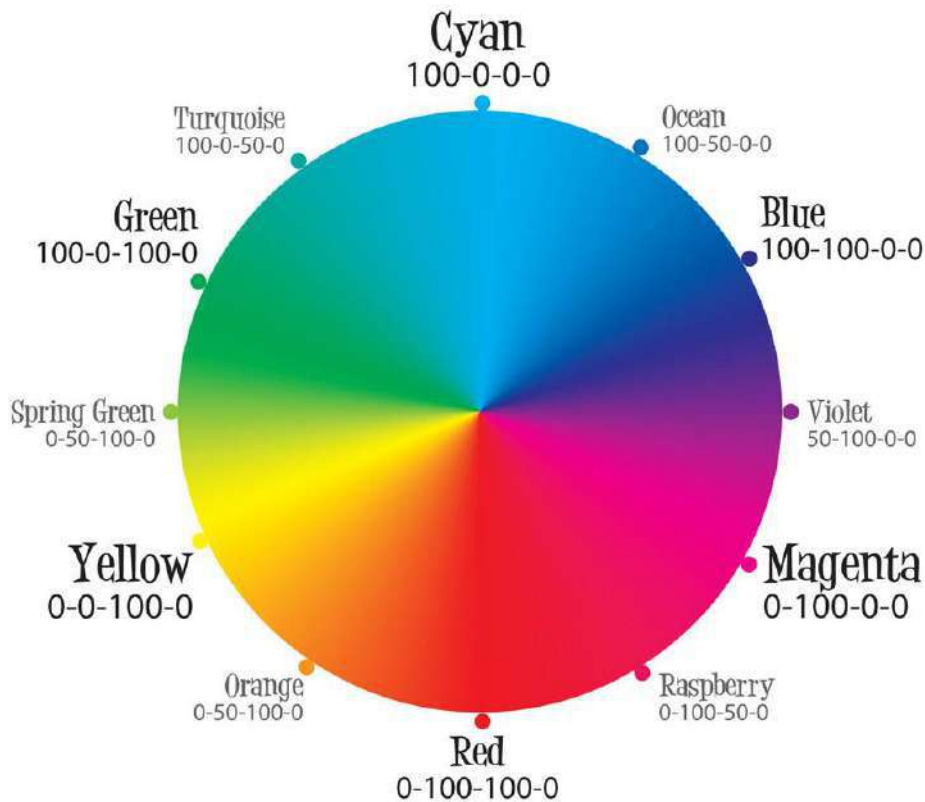
**C (cyan) – 0%; M (magenta) – 0%; Y (yellow) – 0%.**

**Red: C – 0%; M – 100%; Y – 100%.**

**Green: C – 100%; M – 0%; Y – 100%.**

**Blue: C – 100%; M – 100%; Y – 0%.**

**Black: C – 100%; M – 100%; Y – 100%.**



However, this is possible only in theory. But in practice, we can not use CMY colors. Black color turns muddy brown (printing), gray is not similar to the gray, it is problematic to create dark hues. For the settlement of the final color, another color is used (see the last letter in the name CMYK). Transcription of this letter may be different:

\* This may be an abbreviation of black. And it used the last letter, so as not to confuse it with the color Blue in RGB color model

\* Printers often use the word “Contour” on this color. So, it is possible that the letter K in abbreviation CMYK – an abbreviation of the German word “Kontur”

\* It could be a short for Key-color.

\* However, it is difficult to name it as ‘the key’, as it is rather complementary. And this black color is not really black. If only print this paint, it is rather gray than black. Therefore, some people think that the letter K in CMYK stands obreviature “Kobalt” (German).

As a rule, the “black” term is used to define this color

Print using CMYK colors is called “full-color” or “triad.”

We need to note, that during printing, CMYK inks do not mix. They lay down on paper as “spots” (raster) next to each other and mixed already in our’s mind, because these “spots” are very small. That is, the image is rasterized, as otherwise the paint, getting on one another, and spreads produced moire or d

### Grayscale color model

Image in grayscale color model many people mistakenly called ‘black and white’. But it is not. Black-and-white image consists only of black and white tones. While both grayscale has 101 hue. This is gradation of Kobalt color from 0% to 100%.



### Device-dependent and device-independent color model

Color model CMYK and RGB are device-dependent, ie they depend on the way they transfer a color. They point to a specific device, how to use the corresponding colors, but do not have information about the perception of color of the final person. Depending on the settings for brightness, contrast and sharpness of your computer monitor, ambient light and the angle at which we look at the monitor, the color with the same parameters RGB is perceived by us in different ways. A person’s perception of color in the color model “CMYK” depends on an even greater number of conditions, such as the properties of the printed material (for example, glossy paper absorbs less ink than the matte color on it accordingly are more vivid and rich), especially paint, humidity in which the paper is dried up, the characteristics of the printing press and so on

To transfer more reliable information about the color to a person, we attach (so-called) color profiles to the hardware-dependent color models. Each of such profile contains information about a particular way of human transmission of color and adjusts the final color by the addition or removal of any constituent of the original color settings. For example, to print on glossy film uses a color profile, cleaning 10% Cyan and adding 5% Yellow to the original

color, because of the specific features of the printing press, the film itself and other conditions. However, even attached profiles do not solve all the problems of transfer color.

Device-independent color models do not contain any information to transfer a color to a person. They mathematically describe color perceived by man with normal color vision.

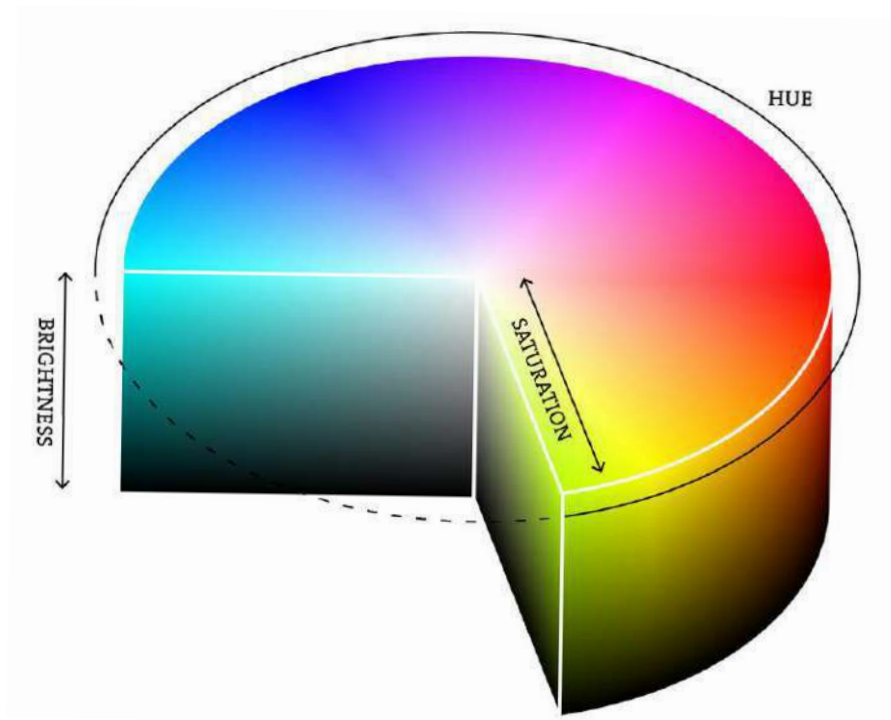
### HSB and HLS color models

The basis of this color space is already familiar rainbow wheel of RGB colorspace. Color is controlled by changing parameters such as Hue, Saturation and Brightness

Parameter hue – the color. Defined by degrees from 0 to 360 based on the colors of the rainbow wheel.

Parameter saturation – percentage of adding to this color of the white paint (has a value from 0% to 100%).

Parameter brightness – the percentage of adding black ink also varies from 0% to 100%.



The principle is similar to one of the representations of the world in terms of art. When we add white or black paint into an existing color.

This is the easiest-to-understand color model, this is why many web-designers like it very much. However, it has several drawbacks:

The human eye perceives colors of the rainbow wheel, as colors with different brightness. For example, spectral green has the greater brightness than the spectral blue. In the HSB color model all the colors of the circle are considered to have a brightness of 100%, which, unfortunately, is not true.

Since it is based on the color model RGB, it is still a hardware-dependent.

This color model is converted to CMYK for printing and converted to RGB for display on the monitor. So, it is very problematic to guess what color you get in the long run.

HLS color model is very similar, it has the following meaning: Hue, Lightness and Saturation

This color model is sometimes used for light and color correction in an image

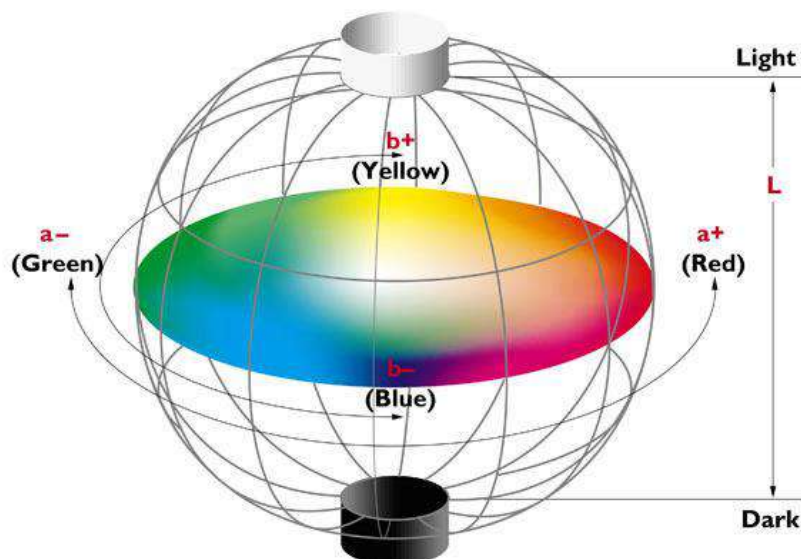
LAB color models

In this color model the color consists of:

**Luminance** – this is set of notions lightness and chroma

**A** – this is color range from green to purple

**B** – this is color range from blue to yellow



That is, the two indicators determine the color (in the aggregate) and one indicator measures the light.

LAB is a device-independent color model, ie it does not depend on how we transfer color. It contains color as RGB and CMYK, and grayscale, which allows it to convert with minimal loss of image quality from one color model to another.

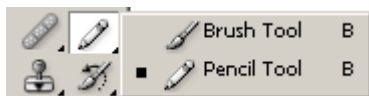
Another advantage is that it, unlike the color model HSB, corresponding to features of perception of color by the human eye.

This model is often used to improve image quality and converting images from one color space to another.

### Painting and Drawing Tools

The main painting tools in **Adobe Photoshop** are the **Pencil** and the **Brush**. The **Pencil** draws free form lines with a hard edge, and the **Brush** - draws lines with a softer edge. It is possible to draw distinct or fuzzy lines with the **Brush**, but they will always be a little soft, because its edges are indistinguishable from the background. Lines drawn with the **Pencil** always have a sharp edge, because there is no interaction with the background.

Both of these tools occupy one cell in the Toolbar and are always portrayed with a pictogram of the last tool used. To choose another tool, press the right mouse button on the arrow next to the tool and choose the required tool from the menu that appears. The menu will also appear if you press and hold the tool's button.



To draw free lines with the **Pencil** or **Brush**, follow these steps:

- **Step 1.** Choose a tool from the Toolbar.
- **Step 2.** Set the color with which colors will be drawn.
- **Step 3.** Choose the parameters for the chosen tool in the Options Panel.
- **Step 4.** Bring the cursor over the image in the photo editor.
- **Step 5.** Press the left mouse button and, while keeping the button pressed, move the cursor across the image.

Most of the parameters used to define the **Pencil** and **Brush** tools in the Options panel are the same, but there are some small differences.

**Brush.** The Brush parameter shows the current shape and size of the brush. To change the shape and size of the brush:

- left-click on the triangle to open the drop-down palette;
- in this palette set the size and hardness of the brush or choose the shape of the brush from the list of presets.


This menu can also be brought up by right-clicking anywhere in the image window. In addition the shape and size of the brushes can be changed through the Brushes palette, which can be opened by pressing or by using the command Window - Brushes.

**Mode.** The Mode parameter sets the mode used by the brush. These modes affect how the colors applied by the brush interact with the colors of the background.

**Opacity.** The Opacity parameter affects the level of opacity in which a line is drawn. To change this parameter, enter a value from 1 to 100 in the parameter's field, or press the triangle button and move the slider. At lower values of Opacity, the color of the lines drawn by the tool blend more intensely with the backgrounds' colors.

**Auto Eraser.** The Auto Eraser parameter is only available for the Pencil tool. When Auto Eraser is checked, the Pencil tool draws over areas of the backgrounds' colors with the primary color and in areas occupied by the primary color in the backgrounds' colors, but if strokes begin where the primary color is not present, then the tool will use the primary color, which is shown at the bottom of the toolbar.

**Flow.** The Flow parameter affects every point of color applied by the tool. Each new application of color is opaquer than the one before. This parameter is only available for the Brush tool.

**Airbrush.** The Airbrush option can also only be set for the Brush tool. When the Airbrush  is pressed the Brush tool paints a line with a border like that of a airbrush.

The **Pencil** and **Brush** tools not only can be used to draw free form lines, but also for drawing straight lines. To draw a straight line with either tool (vertically or horizontally) press Shift and, while holding it down, start to move the cursor in either a vertical or horizontal direction.

## Retouching Images



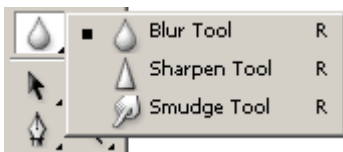
The editing tools do not apply paint to an image, but rather affect the colors already in an image.

**Adobe Photoshop** editing tools are: **Blur, Sharpen, Smudge, Dodge, Burn** and **Sponge**.

To use any of these tools, follow these steps:

- **Step 1.** Choose a tool from the Toolbar.
- **Step 2.** Set the parameters for the chosen tool in the Options Panel.
- **Step 3.** Bring the cursor into the image window.
- **Step 4.** Press the left mouse button and, while keeping it pressed, move the cursor over the image.

The **Blur, Sharpen** and **Smudge** occupy one cell in the Toolbar, represented by the icon of the last tool used. To choose another instrument, press the triangle next to the icon and choose the desired icon from the menu that appears. This menu can also be accessed from the screen, if the icon is pressed and held for a few moments.



**Blur.** The Blur tool reduces the sharpness (focus) of an image. It does this by reducing the color contrast of neighboring pixels.

**Sharpen.** The Sharpen tool increases the sharpness (focus) of an image, by increasing the contrast of neighboring pixels. This results in increased clearness and contrast of borders, and heightened detail in the image.

**Smudge.** The Smudge tool spreads color in an image, displacing pixels of corresponding colors. It is similar to the effect created by smearing your fingers through wet paint. Smudge works by "grasping" a color at the beginning of a stroke and then mixing it with other colors as it is dragged across the image.

The following parameters can be changed for these tools in the Options Panel: **Brush, Mode, Strength, Use All Users, Finger Painting**.

**Brush.** The Brush parameter displays the current shape and size of a brush. To change the shape and size of the brush:

- left-click on the triangular button to open the drop-down palette;

- change the tool's size and hardness in the Options panel or choose its shape from a selection of presets.

The palette can also be accessed by right-clicking anywhere in the image window. In addition the shape and size of the brush can be set in the Brushes palette, which can be opened by pressing or with the command Window - Brushes.

**Mode.** The Mode parameter affects how a tool is applied.

**Strength.** When using the Blur or Sharpen tools the Strength parameter affects the extent to which the tool changes the focus (sharpness) of an image. When using the Smudge tool this parameter defines the distance which the tool smears color in the image.

**Use All Users.** If All New Users is checked when using the Blur or Sharpen tools, it is possible to increase or decrease the sharpness of colors on all visible layers of an image.

**Finger Painting.** This parameter is only available for the Smudge tool. If Finger Painting is checked, it will appear as if before smearing, the "finger" has been dipped into the main color. The result is that not only are colors smeared but an additional hue is added.

### Using Toning Tool

**Dodge, Burn and Sponge** are tools that affect tone. They are used for lightening or darkening parts of an image.

These tools occupy one cell in the Toolbar and are represented by the icon of the last tool used. To choose another tool, right-click on the triangle next to the tool and choose the desired tool from the menu that appears. This menu can also be accessed from the screen, if you click on the icon and hold the button down for a few moments.



**Dodge.** This tool lightens a part of an image, if the cursor is dragged across it.

**Burn.** This tool darkens a part of an image.

**Sponge.** The Sponge tool affects the saturation and contrast of an image.

In the options panel, the following parameters can be adjusted for Dodge and Burn: **Brush, Range, Exposure** and **Airbrush**.

**Brush.** The Brush parameter affects the shape and size of the tool. To change the tools' shape and size:

- press the triangular button with the left mouse button, to open the drop-down palette;
- Set the size and hardness of the tool in the palette or choose its shape from the selection of presets.

**Range.** This parameter affects the mode in which the tool is applied. In Midtones mode dark and light areas are affected equally. In Shadows mode, pixels in darker areas (shadows) are affected more. In Highlights mode pixels in lighter areas are affected more.

**Exposure.** This parameter affects the degree of darkening for Burn and the degree of lightening for Dodge. A value of 100% leads to the maximum degree of darkening or lightening.

**Airbrush.** When the  button is pressed the tool works in dispersion mode.

The Sponge tool can be set in the Options panel with the following parameters: **Brush, Mode, Flow** and **Airbrush**.

**Brush.** The Brush parameter sets the shape and size of the tool. To change the tool's shape and size:

- left-click on the triangular button to open the drop-down palette;
- set the tool's size and hardness in the palette or choose a shape for the tool from the selection of pre-sets.

**Mode.** This parameter switches between the modes in which the tool works. When Sponge is in Desaturation mode the saturation of the primary color decreases, while in Saturation mode it increases.

**Flow.** Flow affects every point of color applied with the tool. With each application of the tool, the color applied becomes opaquer.

**Airbrush.** When the button  is pressed the tool begins to work in dispersion mode

## Zoom tool

When the Zoom tool is active, you also have additional zoom tools on the Options bar. Selecting plus (+) or minus (-) sets the default magnification of the Zoom tool to either enlarge or reduce the image.

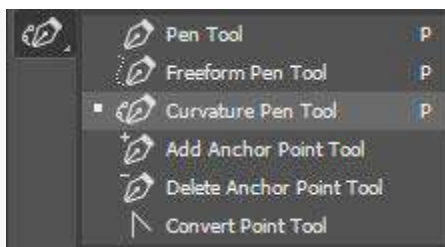
When you zoom in and out, Photoshop doesn't alter the size of the document window, so your image may become too large for its window (in which case, scroll bars appear so you can view the rest of the image) or too small (in which case, a gray border appears around the image).

## PenTool

Photoshop provides multiple Pen tools to suit your use cases and creative style:

- The Curvature Pen tool lets you intuitively draw curves and straight segments.
- The standard Pen tool lets you draw straight segments and curves with great precision.
- The Freeform Pen tool lets you draw paths as if you were drawing with pencil on a piece of paper.
- The Magnetic Pen options let you draw a path that snaps to the edges of the defined areas in your image.

Use the **Shift+P** key combination to cycle through the tools in the Pen group.



*Cycle through the Pen tools using the Shift+P key combination*

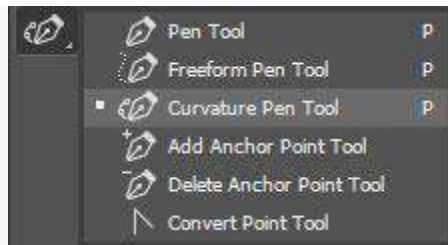
You can use the pen tools in conjunction with the shape tools to create complex shapes. For more information about the modes in which you can draw with the Pen tools.

Use the Curvature Pen tool

The Curvature Pen tool lets you draw smooth curves and straight-line segments with equal ease. Create custom shapes in your designs or define precise paths to effortlessly fine-tune

your images using this intuitive tool. While doing so, create, toggle, edit, add, or remove smooth or corner points without ever having to switch tools.

1. From the Pen tools group, select the Curvature Pen tool.



2. To create the first anchor point, click or tap anywhere in the document.

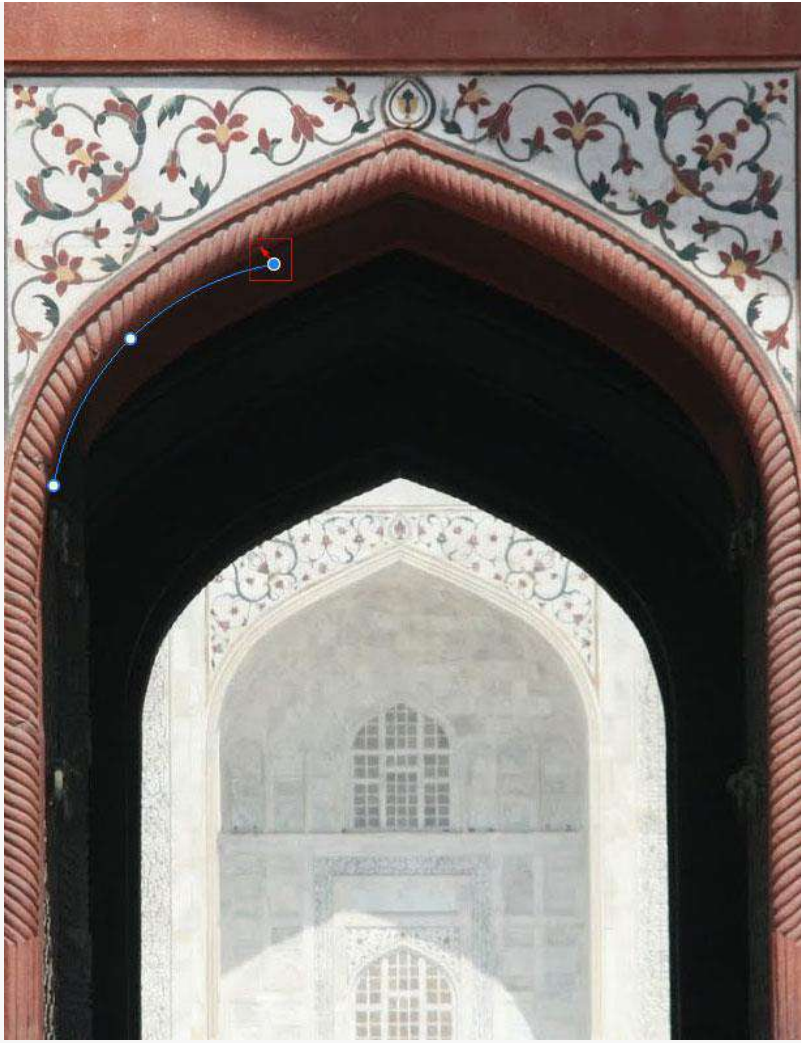


3. Click/tap again to define the second anchor point and complete the first segment of the path. Click once (default) if you want the next segment of your path to be curved. Double-click if you want to draw a straight segment next.



4.
5.  (*Curved path*) Using a mouse or on a touch device, drag the pointer to draw the next segment of your path. While the mouse button is pressed down, optimize the curve of the segment. The previous segment is automatically adjusted to keep the curve smooth.

eep the curve smooth.





*Optimize the curve of the segment while the mouse button is pressed down.*

- (Curved path)* Release the mouse button to drop the anchor point and complete the second segment.
- Draw additional segments and complete the path.





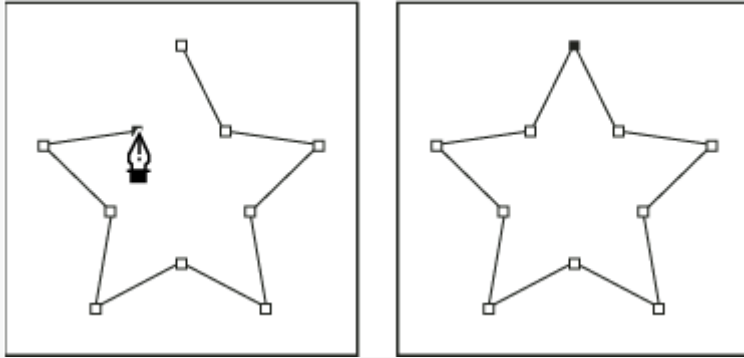
- When you're done drawing, press the **Esc** key.

---

Use the standard Pen tool

### **Draw straight line segments**


The simplest path you can draw with the standard Pen tool is a straight line, made by clicking the Pen tool to create two anchor points. By continuing to click, you create a path made of straight line segments connected by corner points.



*Clicking the Pen tool creates straight segments.*

1. Select the Pen tool.
2. Position the Pen tool where you want the straight segment to begin, and click to define the first anchor point (do not drag).
1. Click again where you want the segment to end (Shift-click to constrain the angle of the segment to a multiple of 45°).
2. Continue clicking to set anchor points for additional straight segments.

The last anchor point you add always appears as a solid square, indicating that it is selected. Previously defined anchor points become hollow, and deselected, as you add more anchor points.

3. Complete the path by doing one of the following:
  - To close the path, position the Pen tool over the first (hollow) anchor point. A small circle appears next to the Pen tool pointer  when it is positioned correctly. Click or drag to close the path.
  - To leave the path open, Ctrl-click (Windows) or Command-click (Mac OS) anywhere away from all objects.

To leave the path open, you can also select a different tool.

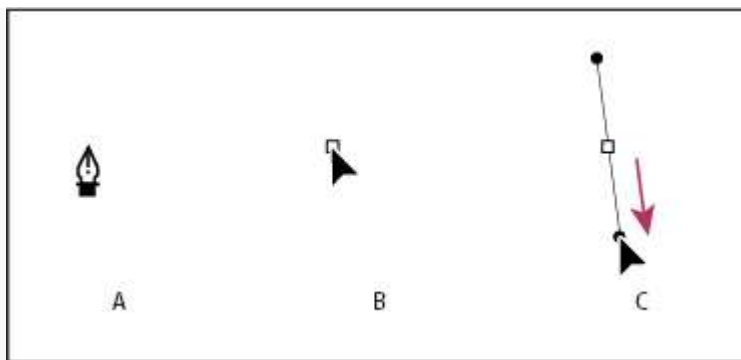
### **Draw curves with the standard Pen tool**

You create a curve by adding an anchor point where a curve changes direction, and dragging the direction lines that shape the curve. The length and slope of the direction lines determine the shape of the curve.

Curves are easier to edit and your system can display and print them faster if you draw them using as few anchor points as possible. Using too many points can also introduce unwanted bumps in a curve. Instead, draw widely spaced anchor points, and practice shaping curves by adjusting the length and angles of the direction lines.

1. Select the Pen tool.
2. Position the Pen tool where you want the curve to begin and hold down the mouse button. The first anchor point appears, and the Pen tool pointer changes to an arrowhead. (In Photoshop, the pointer changes only after you've started dragging.)
3. Drag to set the slope of the curve segment you're creating, and then release the mouse button. In general, extend the direction line about one third of the distance to the next anchor point you plan to draw. (You can adjust one or both sides of the direction line later.)

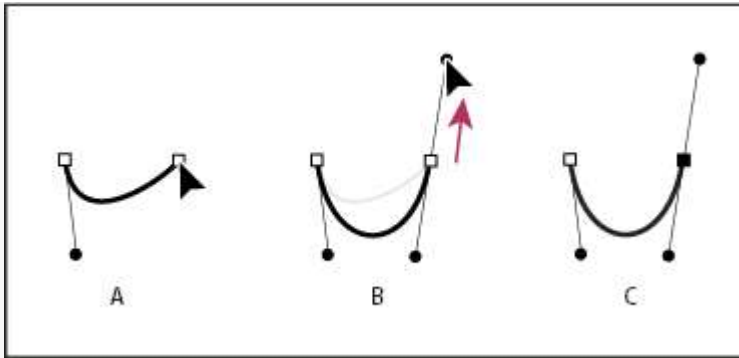
Hold down the Shift key to constrain the tool to multiples of 45°.



*Drawing the first point in a curve*

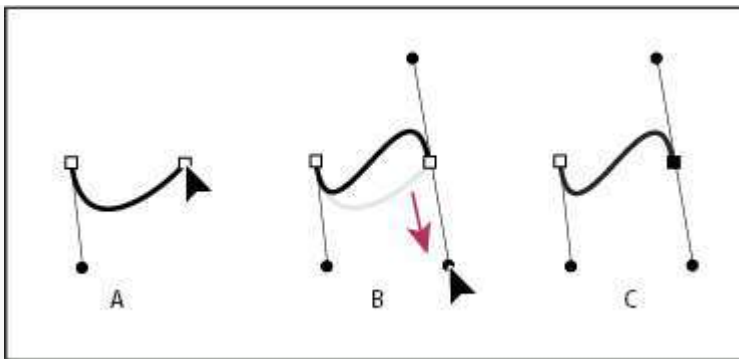
**A.** Positioning Pen tool **B.** Starting to drag (mouse button pressed) **C.** Dragging to extend direction lines

4. Position the Pen tool where you want the curve segment to end, and do one of the followings:
  - To create a C-shaped curve, drag in a direction opposite to the previous direction line. Then release the mouse button.




*Drawing the second point in a curve*

- To create an S-shaped curve, drag in the same direction as the previous direction line. Then release the mouse button.



*Drawing an S curve*

### **Finish drawing a path**

1. Complete a path in one of the following ways:
  - To close a path, position the Pen tool over the first (hollow) anchor point. A small circle appears next to the Pen tool pointer  when it is positioned correctly. Click or drag to close the path.
  - To leave a path open, Ctrl-click (Windows) or Command-click (Mac OS) anywhere away from all objects.

### **Settings in the Options bar**


When you use the standard Pen tool, the following options are available in the options bar:

- Auto Add/Delete, which lets you add an anchor point when you click a line segment or delete an anchor point when you click it.
- Rubber Band, which lets you preview path segments as you move the pointer between clicks. To access this option, click the pop-up menu to the right of the Custom Shape icon.

---

### Use the Freeform Pen tool

The Freeform Pen tool lets you draw as if you were drawing with a pencil on paper. Anchor points are added automatically as you draw. You do not determine where the points are positioned, but you can adjust them once the path is complete. To draw with greater precision, use the Pen tool.

1. Select the Freeform Pen tool .
2. To control how sensitive the final path is to the movement of your mouse or stylus, click the inverted arrow next to the shape buttons in the options bar, and enter a value between 0.5 and 10.0 pixels for Curve Fit. A higher value creates a simpler path with fewer anchor points.
3. Drag the pointer in the image. As you drag, a path trails behind the pointer. When you release the mouse, a work path is created.
4. To continue the existing freehand path, position the pen pointer on an end point of the path, and drag.
5. To complete the path, release the mouse. To create a closed path, drag the line to the initial point of the path (a circle appears next to the pointer when it is aligned).

### Using channels

*Channels* are grayscale images that store different types of information:

- *Color information channels* are created automatically when you open a new image. The image's color mode determines the number of color channels created. For example, an RGB image has a channel for each color (red, green, and blue) plus a composite channel used for editing the image.
- *Alpha channels* store selections as grayscale images. You can add alpha channels to create and store masks, which let you manipulate or protect parts of an image.

- *Spot color channels* specify additional plates for printing with spot color inks. An image can have up to 56 channels. All new channels have the same dimensions and number of pixels as the original image.

The file size required for a channel depends on the pixel information in the channel. Certain file formats, including TIFF and Photoshop formats, compress channel information and can save space. The size of an uncompressed file, including alpha channels and layers, appears as the right-most value in the status bar at the bottom of the window when you choose Document Sizes from the pop-up menu.

---

### Channels panel overview

The Channels panel lists all channels in the image—composite channel first (for RGB, CMYK, and Lab images). A thumbnail of the channel’s contents appears to the left of the channel name; the thumbnail is automatically updated as you edit the channel.



### Channel types

**A.** Color channels **B.** Spot channels **C.** Alpha channels

### Display the Channels panel

1. Choose Windows > Channels.

### Resize or hide channel thumbnails

1. Choose Panel Options from the Channels panel menu. Click a thumbnail size or click None to turn off the display of thumbnails.

Viewing thumbnails is a convenient way of tracking channel contents; however, turning off the display of thumbnails can improve performance.

---


#### Show or hide a channel

You can use the Channels panel to view any combination of channels in the document window. For example, you can view an alpha channel and the composite channel together to see how changes made in the alpha channel relate to the entire image.

1. Click in the eye column next to the channel to show or hide that channel. (Click the composite channel to view all default color channels. The composite channel is displayed whenever all the color channels are visible.)
- 

#### Show color channels in color

Individual channels are displayed in grayscale. In RGB, CMYK, or Lab images, you can view the individual channels in color. (In Lab images, only the *a* and *b* channels appear in color.) If more than one channel is active, the channels always appear in color.

You can change the default to show the individual color channels in color. When a channel is visible in the image, an eye icon  appears to its left in the panel.

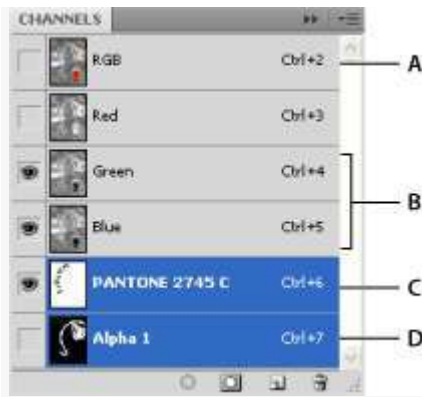
1. Do one of the followings:
  - In Windows, choose Edit > Preferences > Interface.
  - In Mac OS, choose Photoshop > Preferences > Interface.

Select Show Channels inColor and click OK.

---

#### Select and edit channels

You can select one or more channels in the Channels panel. The names of all selected, or *active*, channels are highlighted.



### *Selecting multiple channels*

**A.** Not visible or editable **B.** Visible but not selected for editing **C.** Selected for viewing and editing **D.** Selected for editing but not viewing

- To select a channel, click the channel name. Shift-click to select (or deselect) multiple channels.
- To edit a channel, select it and then use a painting or editing tool to paint in the image. You can paint on only one channel at a time. Paint with white to add the selected channel's color at 100% intensity. Paint with a value of gray to add the channel's color at a lower intensity. Paint with black to fully remove the channel's color.

---

### Rearrange and rename alpha and spot channels

You can move alpha or spot channels above the default color channels only if the image is in Multichannel mode (Image > Mode > Multichannel). For information about that mode's limitations,

To change the order of alpha or spot channels, drag the channel up or down in the Channels panel. When a line appears in the position you want, release the mouse button.


- To rename an alpha or spot channel, double-click the channel's name in the Channels panel, and enter a new name.

---

### Delete a channel

You may want to delete spot or alpha channels you no longer need before saving an image. Complex alpha channels can substantially increase the disk space required for an image.



1. In Photoshop, select the channel in the Channels panel and do one of the followings:
  - Alt-click (Windows) or Option-click (Mac OS) the Delete icon .
  - Drag the channel name in the panel to the Delete icon.
  - Choose Delete Channel from the Channels panel menu.
  - Click the Delete icon at the bottom of the panel, and then click Yes.

## LAYER

A layer is simply one image stacked on top of another. Imagine I have a piece of paper and I paint it red. Then I take a piece of clear cellophane and paint a yellow circle and lay it over the paper. Now I take another piece of cellophane and paint some blue type and lay that on top of the yellow circle. I now have a background (red) and 2 layers (yellow and blue.) Just like in the picture below.

A background with 2 layers.



This is how your image with would look on the screen or when printed.

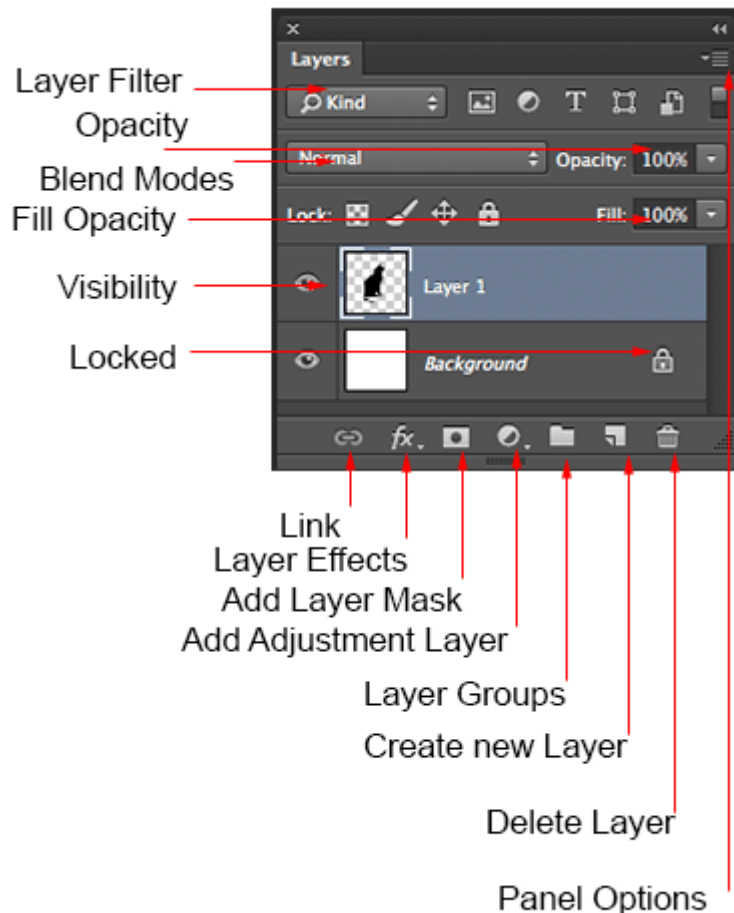
Broken apart so you can see how the layers work  
Each layer stacks on top of the previous one.



That is, it! The concept of layers is that simple. Photoshop uses the Layers Pallette to allow you to do this with your images. More than one layer is called a composition.

## LAYERS PANEL

Photoshop's layers Panel is a powerful tool that allows you do many special things to your layed compositions. Next, we will look at the Photoshop layers pallete.



Have you ever wondered what all the parts of a layer's panel do? Here is a screen grab of the layers Panel. I'll explain what all the parts are here.

**Layer Filter:** This enables you to hide layers based on different things. Makes it easier to find the layers that you want to work with.

**Opacity:** 0= transparent 100 = fully opaque. press number keys on keyboard to instantly set to multiples of 10 or adjust the slider for an exact amount of transparency on each layer.

**Blend Modes:** Change these to change the way that the selected layer blends with the layers underneath it. Great for compositing and special effects. (With the move tool selected, press *Shift+* or *Shift-* to cycle through blending modes.

**Fill opacity:** Adjusts the amount of opacity of the pixels only, but any layer styles are unaffected and remain 100% opaque.

**visibility:** If the eye is showing that layer is visible. Click on the eye and the layer will still be there but invisible until you click on the eye again.

**Locked:** The padlock means that something is locked in the layer. (Also click in the 4 icons in the “lock” next to fill opacity to make certain things editable or locked). Here are the different things that can be locked/unlocked.

**Lock all:** If the box is checked the layer is totally protected from any editing.

**Lock Position:** You can make any changes except for moving the image.

**Lock Image pixels:** You cannot draw on this layer if checked.

**Lock transparent:** You can paint on this layer but not where it is transparent.

Useful tools at the bottom of the panel

**Link:** Enabled you to link layers. These will all move together unless unlinked.

**Layer Effects (Styles):** Special effects applied to your image layer. Noted by the little f. Each effect will be listed. multiple effects may be used at once.

**Add Layer Mask:** This is the button to press to add a layer mask to the currently selected layer. Allows you to paint away parts of your layer without damaging your original image.

**Add Adjustment Layer:** The best way to apply image adjustments. There can change the color or tone of an image. All layers are affected underneath an adjustment layer (Unless clipped). This is a good option to using Image>Adjustments because adjustment layers are non-destructive and re-editable.

**Layer Groups:** A good organizational tool. This puts layers into a folder. You can choose multiple layers and press Cmd/Ctrl+G to put them in a group or create a group by clicking this icon. Layers can be dragged in or out of groups in the Layers panel.

**Create New Layer:** Press this icon to create a new layer. Drag an existing layer into this icon to create a duplicate of that layer,

**Delete Layer:** Drag a layer into this icon to remove it. Or select the layer and then press this icon to get the same result.

**Panel Options:** This will open a drop-down menu that provides a number of options, many that aren't listed anywhere else.

[Photoshop actions](#) enable you to record a repetitive process and save that information as an action which you can then use for other tasks down the road. Not only that, you can edit actions after the fact and customize them to suit your needs.

An *action* is a series of tasks that you play back on a single file or a batch of files—menu commands, panel options, tool actions, and so on. For example, you can create an action that changes the size of an image, applies an effect to the image, and then saves the file in the desired format.

Actions can include steps that let you perform tasks that cannot be recorded (for example, using a painting tool). Actions can also include modal controls that let you enter values in a dialog box while playing an action.

In Photoshop, actions are the basis for *droplets*, which are small applications that automatically process all files that are dragged onto their icon.

Photoshop and Illustrator come with predefined actions installed that help you perform common tasks. You can use these actions as is, customize them to meet your needs, or create new actions. Actions are stored in sets to help you organize them.

You can record, edit, customize, and batch-process actions, and you can manage groups of actions by working with action sets.

## **01. Open the Actions panel**

When you open the Actions panel in Photoshop CC this is what you see. Note all the actions which are included, which you can use immediately.

Before beginning to record actions, it is a good idea to close the Default Actions and create a New Set with your name.

Click on the icon at the top right of the Actions panel. When the pop-up appears, choose New Set and in the dialog box that appears immediately after that, type in your name and click on OK.

A new set appears with your name.

## **02. Record your action**

Recording an action is easy. Simply click on the icon at the top right of the Actions panel and in the popup, menu click on New Action.

In the New Action dialog box, type in the name of your action and click on Record.

All the steps you take will be recorded in the Actions panel.

What you will find is that creating an action is often a process of experimentation to get the affects you desire. To end the recording, press the stop button.

### **03. Play the action**

When playing actions, you have a few options. To access those, click on the icon at the top-right of the Actions panel and in the pop-up menu, click on Playback Options.

This brings up the Playback Options dialog box. You have three choices: Accelerated, Step by Step and Pause For.

Accelerate will play the action at normal speed. Be aware that when you do so, you might not be able to see what the action is doing to your file. If you want to see how the Action performs, choose the Step by Step option instead.

The third option, pause for \_\_\_ Seconds allows you to set a pause between each step of the action. This could be useful for debugging if you are having trouble with the execution of the action.

### **04. Managing and editing actions**

You can apply an action to other images after it has been recorded. Another way to apply actions is to go to File>Automate>Batch.

Note that at the top of the dialog box are three important drop downs. The Set heading refers to Action Sets that have been recorded. In this case, the default action set is available.

Immediately under that is the action heading. Here, you can choose which action you want to use. The third drop down is for the source of images, which could be a folder, import, opened files, or Bridge.

### **05. For best results**

Take some time to plan the steps of the actions before recording them and if necessary, write down all the steps on a piece of paper before you begin. While you can edit actions after the

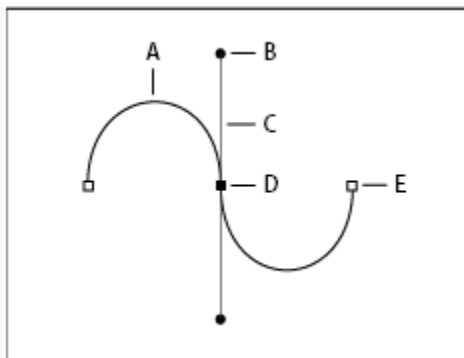
fact, it is a better use of your time to work out any potential problems before you begin recording.

It is also a good idea to keep a log of the actions you record with all the steps in case you need to make changes later.

---

## Path segments, components, and points

A path consists of one or more straight or curved segments. *Anchor points* mark the end points of the path segments. On curved segments, each selected anchor point displays one or two *direction lines*, ending in *direction points*. The positions of direction lines and points determine the size and shape of a curved segment. Moving these elements reshapes the curves in a path.

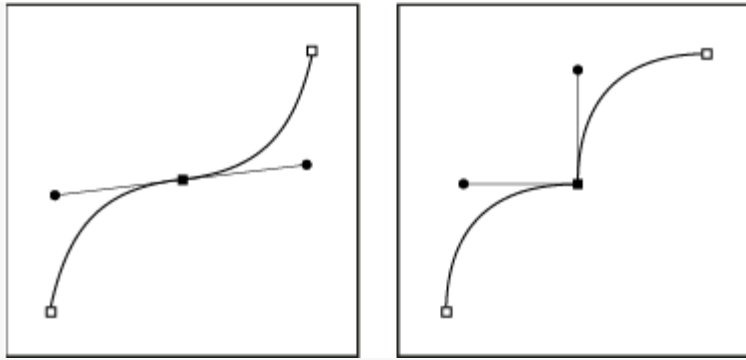


*A path*

**A.** Curved line segment **B.** Direction point **C.** Direction line **D.** Selected anchor point **E.** Unselected anchor point

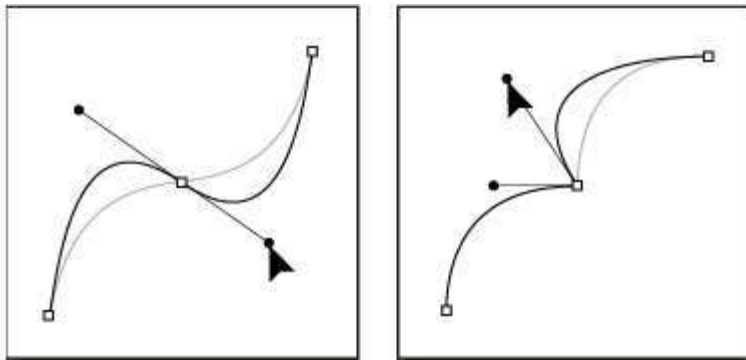
A path can be *closed*, with no beginning or end (for example, a circle), or *open*, with distinct *end points* (for example, a wavy line).

Smooth curves are connected by anchor points called *smooth points*. Sharply curved paths are connected by *corner points*.



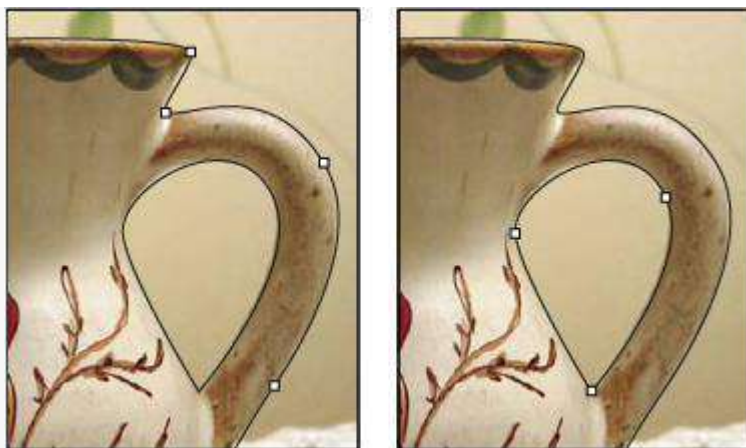
*Smooth point and corner point*

When you move a direction line on a smooth point, the curved segments on both sides of the point are adjusted simultaneously. By comparison, when you move a direction line on a corner point, only the curve on the same side of the point as the direction line is adjusted.



*Adjusting a smooth point and a corner point*

A path does not have to be one connected series of segments. It can contain more than one distinct and separate *path component*. Each shape in a shape layer is a path component, as described by the layer's clipping path.





## *Separate path components selected*

### **Select a path**

Selecting a path component or path segment displays all of the anchor points on the selected portion, including any direction lines and direction points if the selected segment is curved. Direction handles appear as filled circles, selected anchor points as filled squares, and unselected anchor points as hollow squares.

1. Do one of the following:

- To select a path component (including a shape in a shape layer), select the Path Selection tool , and click anywhere inside the path component. If a path consists of several path components, only the path component under the pointer is selected.
- To select a path segment, select the Direct Selection tool , and click one of the segment's anchor points, or drag a marquee over part of the segment.



*Drag a marquee to select segments.*

To select additional path components or segments, select the Path Selection tool or the Direct Selection tool, and then hold down Shift while selecting additional paths or segments.

#### **Note:**

When the Direct Selection tool is selected, you can select the entire path or path component by Alt-clicking (Windows) or Option-clicking (Mac OS) inside the path. To activate the



Direct Selection tool when most other tools are selected, position the pointer over an anchor point, and press Ctrl (Windows) or Command (Mac OS).

## Select multiple paths | Photoshop CC

You can select multiple paths on the same layer or across different layers.

1. In the **Paths** panel, do any of the following to make the paths visible:

- Shift-click to select contiguous paths.
- Ctrl-click (Windows) or Command-click (Mac OS) to select non-contiguous paths.

Select the **Path Selection** tool or the **Direct Selection** tool and do any of the following:

- Drag over the segments.
- Shift-click the paths.

To select additional path components or segments, select the **Path Selection** tool or the **Direct Selection** tool, and then hold down the Shift key while selecting additional paths or segments.

### Note:

You can choose to work with paths in the isolation mode. To isolate only the layer containing a path, with the path active, double-click using a selection tool. You can also isolate single or multiple layers by using the **Select/Isolate Layers** menu item or by setting **Layer Filtering** to **Selected**.

You can exit the isolation mode in several ways, such as:

- *Turning off **Layer Filtering***
- *Switching **Layer Filtering** to something other than **Selected***
- *Double-clicking away from a path using the path selection tools*

---

## Reorder paths

You can reorder saved paths that are not Shape, Type, or Vector Mask paths in the **Paths** panel.

1. In the **Paths** panel, drag the path to the position you want. In Photoshop CC, you can select and drag more than one path simultaneously.

---

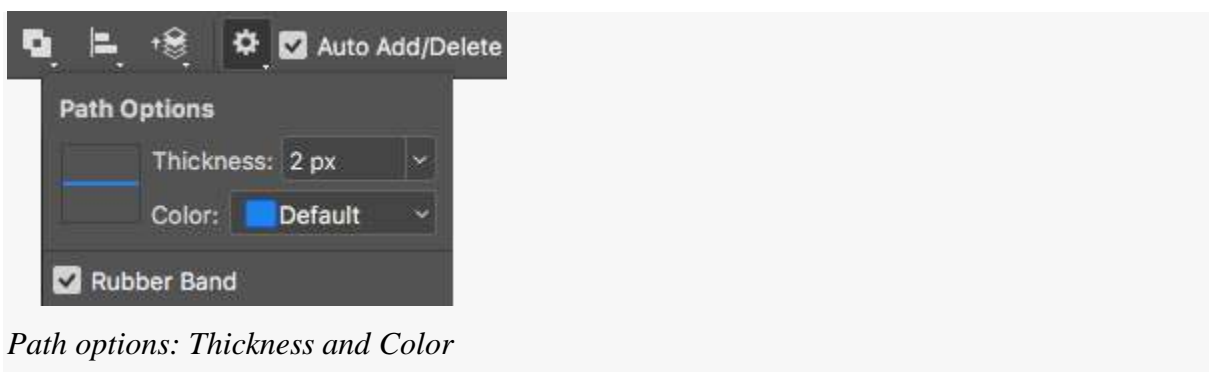
## Duplicate paths

1. In the **Paths** panel, select the path you want to duplicate. In Photoshop CC, you can select more than one path.
2. Do any of the following:
  - Alt-drag (Windows) or Option-drag the paths.
  - Choose **Duplicate Path** from the panel menu.

---

## Specify path options

You can define the color and thickness of path lines to suit your taste and for easier visibility. While creating a path—using the Pen tool, for example—click the gear icon (⚙️) in the Options bar. Now specify the color and thickness of path lines. Also, specify whether you want to preview path segments as you move the pointer between clicks (**Rubber Band** effect).



---


## Adjust path segments

You can edit a path segment at any time, but editing existing segments is slightly different from drawing them. Keep the following tips in mind when editing segments:


- If an anchor point connects two segments, moving that anchor point always changes both segments.

- When drawing with the Pen tool, you can temporarily activate the Direct Selection tool so that you can adjust segments you've already drawn; press Ctrl (Windows) or Command (Mac OS) while drawing.
- When you initially draw a smooth point with the Pen tool, dragging the direction point changes the length of the direction line on both sides of the point. However, when you edit an existing smooth point with the Direct Selection tool, you change the length of the direction line only on the side you're dragging.


### Move straight segments

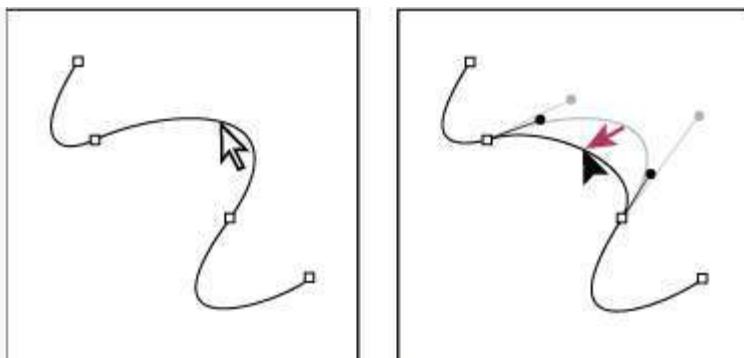
1. With the Direct Selection tool , select the segment you want to adjust.
2. Drag the segment to its new position.

### Adjust the length or angle of straight segments

1. With the Direct Selection tool , select an anchor point on the segment you want to adjust.
2. Drag the anchor point to the desired position. Shift-drag to constrain the adjustment to multiples of 45°.

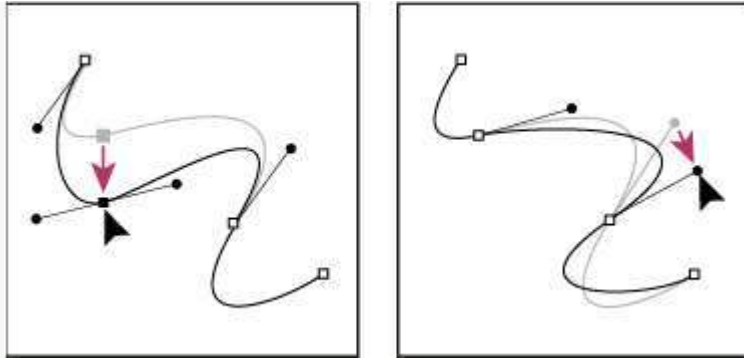
### Adjust the position or shape of curved segments

1. With the Direct Selection tool , select a curved segment, or an anchor point on either end of the curved segment. Direction lines appear, if any are present. (Some curved segments use just one direction line.)
2. Do any of the following:
  - To adjust the position of the segment, drag the segment. Shift-drag to constrain the adjustment to multiples of 45°.



*Click to select the curve segment. Then drag to adjust.*

- To adjust the shape of the segment on either side of a selected anchor point, drag the anchor point or the direction point. Shift-drag to constrain movement to multiples of 45°.



*Drag the anchor point or drag the direction point.*



**Note:**

Adjusting a path segment also adjusts the related segments, letting you intuitively transform path shapes. To only edit segments between the selected anchor points, similar to earlier Photoshop versions, select Constrain Path Dragging in the options bar.

**Note:**

You can also apply a transformation, such as scaling or rotating, to a segment or anchor point.

**Delete a segment**

1. (Optional) If you're creating an opening in a closed path, select the Add Anchor Point tool , and add two points where you want the cut to occur.
2. Select the Direct Selection tool , and select the segment you want to delete.
3. Press Backspace (Windows) or Delete (Mac OS) to delete the selected segment. Pressing Backspace or Delete again erases the rest of the path.

**Delete the direction line of an anchor point**


- Using the Convert Anchor Point tool, click the anchor point of the direction line.

The smooth point becomes a corner point.

### **Extend an open path**

1. Using the Pen tool, position the pointer over the endpoint of the open path you want to extend. The pointer changes when it's precisely positioned over the endpoint.
2. Click the endpoint.
3. Do one of the followings:
  - To create a corner point, position the Pen tool where you want to end the new segment, and click. If you are extending a path that ends at a smooth point, the new segment will be curved by the existing direction line.
  - To create a smooth point, position the Pen tool where you want to end the new curved segment, and drag.

### **Connect two open paths**

1. Using the Pen tool, position the pointer over the endpoint of the open path that you want to connect to another path. The pointer changes when it's precisely positioned over the endpoint.
2. Click the endpoint.
3. Do one of the following:
  - To connect the path to another open path, click an endpoint on the other path. When you precisely position the Pen tool over the other path's endpoint, a small merge symbol  appears next to the pointer.
  - To connect a new path to an existing path, draw the new path near the existing path, and then move the Pen tool to the existing path's (unselected) endpoint. Click that endpoint when you see the small merge symbol that appears next to the pointer.

### **Move or nudge anchor points or segments using the keyboard**




1. Select the anchor point or path segment.
2. Click or hold down any of the arrow keys on the keyboard to move 1 pixel at a time in the direction of the arrow.

Hold down the Shift key in addition to the arrow key to move 10 pixels at a time.

---

### **Add or delete anchor points**

Adding anchor points can give you more control over a path or it can extend an open path. However, try not to add more points than necessary. A path with fewer points is easier to edit, display, and print. You can reduce the complexity of a path by deleting unnecessary points.

The toolbox contains three tools for adding or deleting points: The Pen tool , the Add Anchor Point tool , and the Delete Anchor Point tool .

By default, the Pen tool changes to the Add Anchor Point tool as you position it over a selected path, or to the Delete Anchor Point tool as you position it over an anchor point. You must select Auto Add/Delete in the options bar to enable the Pen tool to automatically change to the Add Anchor Point or Delete Anchor Point tool.

You can select and edit multiple paths simultaneously. You can also reshape a path while adding anchor points by clicking and dragging as you add.

#### **Note:**

Don't use the Delete or Backspace keys or the Edit > Cut or Edit > Clear commands to delete anchor points. These keys and commands delete the point and line segments that connect to that point.

### **Add or delete anchor points**

1. Select the path you want to modify.
2. Select the Pen tool, the Add Anchor Point tool, or the Delete Anchor Point tool.
3. To add an anchor point, position the pointer over a path segment and click. To delete an anchor point, position the pointer over an anchor point and click.

### **Disable or temporarily override automatic Pen tool switching**

You can override automatic switching of the Pen tool to the Add Anchor Point tool or the Delete Anchor Point tool. This is useful when you want to start a new path on top of an existing path.

- In Photoshop, deselect Auto Add/Delete in the options bar.

---

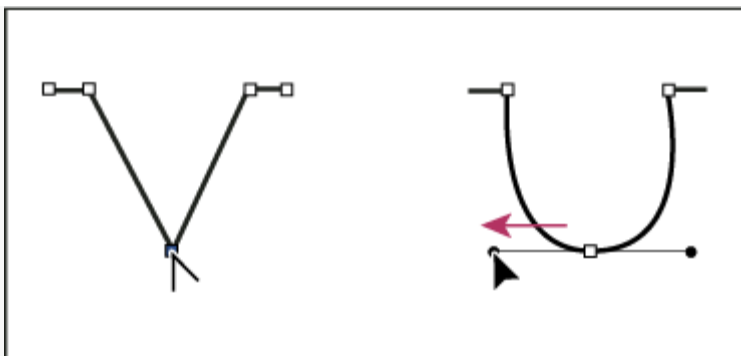
### Convert between smooth points and corner points

1. Select the path you want to modify.
2. Select the Convert Point tool or use the Pen tool and hold down Alt (Windows) or Option (Mac OS).

**Note:**

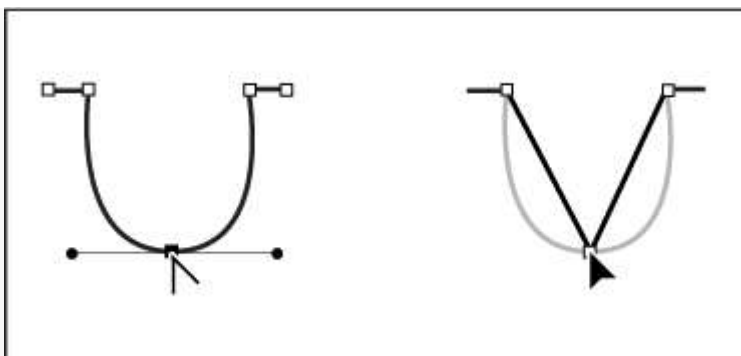
To activate the Convert Point tool while the Direct Selection tool is selected, position the pointer over an anchor point, and press Ctrl+Alt (Windows) or Command+Option (Mac OS).

3. Position the Convert Point tool over the anchor point you want to convert, and do one of the followings:
  - To convert a corner point to a smooth point, drag away from the corner point to make direction lines appear.



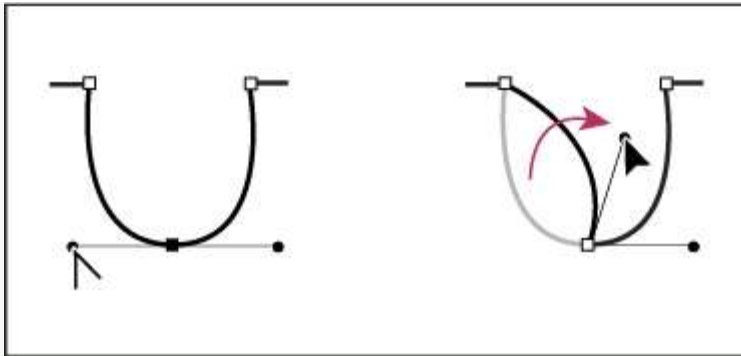
*Dragging a direction point out of a corner point to create a smooth point*

- To convert a smooth point to a corner point without direction lines, click the smooth point.



### *Clicking a smooth point to create a corner point*

- To convert a corner point without direction lines to a corner point with independent direction lines, first drag a direction point out of a corner point (making it a smooth point with direction lines). Release the mouse button only (don't release any keys you may have pressed to activate the Convert Anchor Point tool), and then drag either direction point.
- To convert a smooth point to a corner point with independent direction lines, drag either direction point.




### *Converting a smooth point to a corner point*

## **Adjust path components**

You can reposition a path component (including a shape in a shape layer) anywhere within an image. You can copy components within an image or between two Photoshop images. Using the Path Selection tool, you can merge overlapping components into a single component. All vector objects, whether they are described by a saved path, work path, or vector mask, can be moved, reshaped, copied, or deleted.

You can also use the Copy and Paste commands to duplicate vector objects between a Photoshop image and an image in another application, such as Adobe Illustrator.

## **Change the overlap mode for the selected path component**

1. Using the Path Selection tool , drag a marquee to select existing path areas.
2. Choose a shape area option from the Path Operations drop-down menu in the options bar:



## Combine Shapes

Adds the path area to overlapping path areas.

## Subtract From Shape Area

Removes the path area from overlapping path areas.

## Intersect Shape Areas

Restricts the area to the intersection of the selected path area and overlapping path areas.

## Exclude Overlapping Shape Areas


Excludes the overlap area.

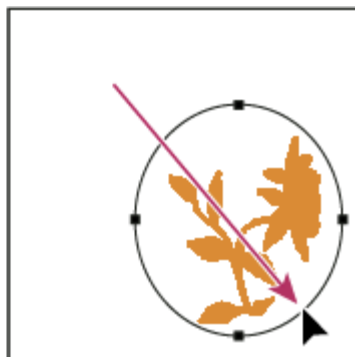
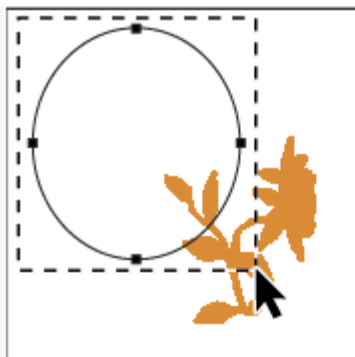
## Show or hide the selected path component

Do one of the following:

- Choose View > Show > Target Path.
- Choose View > Extras. This command also shows or hides a grid, guides, selection edges, annotations, and slices.

## Move a path or path component

1. Select the path name in the Paths panel, and use the Path Selection tool  to select the path in the image. To select multiple path components, Shift-click each additional path component to add it to the selection.
2. Drag the path to its new location. If you move any part of a path beyond the canvas boundaries, the hidden part of the path is still available.




*Dragging a path to a new location*


## Note:

If you drag a path so that the move pointer is over another open image, the path is copied to that image.

### **Reshape a path component**



1. Select the path name in the Paths panel, and use the Direct Selection tool  to select an anchor point in the path.
2. Drag the point or its handles to a new location.

### **Merge overlapping path components**

1. Select the path name in the Paths panel, and select the Path Selection tool .
2. To create a single component from all overlapping components, choose Merge Shape Components from the Path Operations drop-down menu in the options bar.


### **Copy a path component or path**

Do any of the following:


- To copy a path component as you move it, select the path name in the Paths panel, and click a path component with the Path Selection tool . Then Alt-drag (Windows) or Option-drag (Mac OS) the selected path.
- To copy a path without renaming it, drag the path name in the Paths panel to the New Path button  at the bottom of the panel.
- To copy and rename a path, Alt-drag (Windows) or Option-drag (Mac OS) the path in the Paths panel to the New Path button at the bottom of the panel. Or select the path to copy, and choose Duplicate Path from the Paths panel menu. Enter a new name for the path in the Duplicate Path dialog box, and click OK.
- To copy a path or path component into another path, select the path or path component you want to copy, and choose Edit > Copy. Then select the destination path, and choose Edit > Paste.

### **Copy path components between two Photoshop files**

1. Open both images.

2. In the source image, use the Path Selection tool  to select the entire path or the path components that you want to copy.
3. To copy the path component, do any of the following:
  - Drag the path component from the source image to the destination image. The path component is copied to the active path in the Paths panel.
  - In the source image, select the path name in the Paths panel and choose Edit > Copy to copy the path. In the destination image, choose Edit > Paste. You can also use this method to combine paths in the same image.
  - To paste the path component into the destination image, select the path component in the source image, and choose Edit > Copy. In the destination image, choose Edit > Paste.

### Delete a path component


1. Select the path name in the Paths panel, and click a path component with the Path Selection tool .
2. Press Backspace (Windows) or Delete (Mac OS) to delete the selected path component.

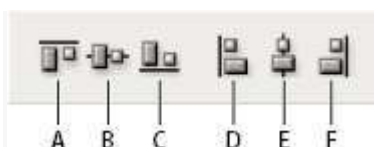
### Align and distribute path components

You can align and distribute path components that are described in a single path. For example, you can align the left edges of several shapes contained in a single layer or distribute several components in a work path along their horizontal centers.

#### Note:

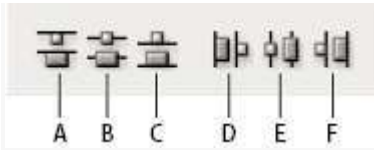
To align shapes that are on separate layers, use the Move tool.

- To align components, use the Path Selection tool  to select the components you want to align. Then choose an option from the Path Alignment drop-down menu in the options bar.



*Alignment options*

- To distribute components, select at least three components you want to distribute. Then choose an option from the Path Arrangement drop-down menu in the options bar.



### *Distribute options*

*Channels* are grayscale images that store different types of information:

- *Color information channels* are created automatically when you open a new image. The image's color mode determines the number of color channels created. For example, an RGB image has a channel for each color (red, green, and blue) plus a composite channel used for editing the image.
- *Alpha channels* store selections as grayscale images. You can add alpha channels to create and store masks, which let you manipulate or protect parts of an image.
- *Spot color channels* specify additional plates for printing with spot color inks.
- An image can have up to 56 channels. All new channels have the same dimensions and number of pixels as the original image.

The file size required for a channel depends on the pixel information in the channel. Certain file formats, including TIFF and Photoshop formats, compress channel information and can save space. The size of an uncompressed file, including alpha channels and layers, appears as the right-most value in the status bar at the bottom of the window when you choose Document Sizes from the pop-up menu.

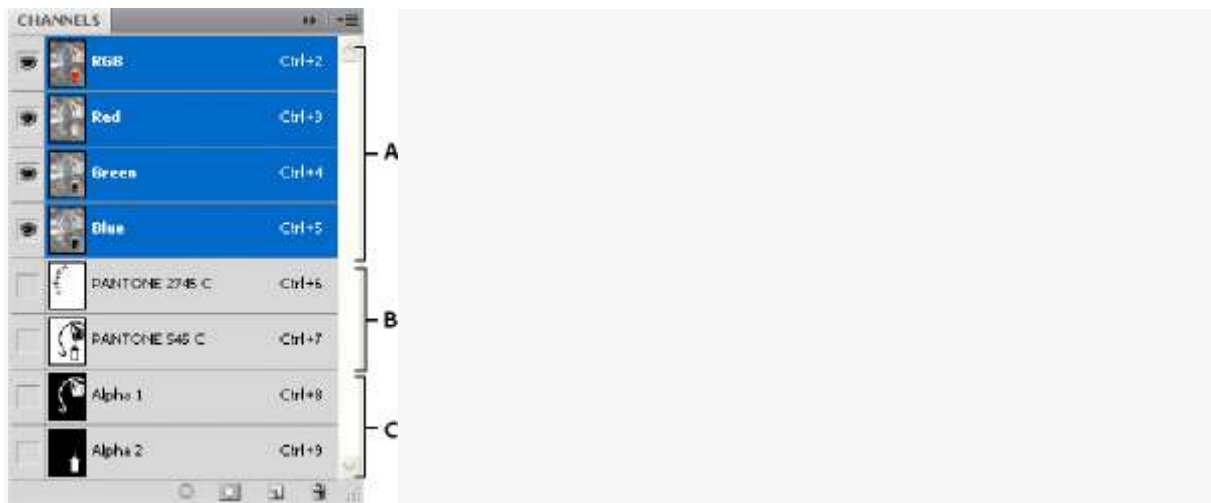
### **Note:**

As long as you save a file in a format supporting the image's color mode, the color channels are preserved. Alpha channels are preserved only when you save a file in Photoshop, PDF, TIFF, PSB, or raw formats. DCS 2.0 format preserves only spot channels. Saving in other formats may cause channel information to be discarded.

---

## **Channels panel overview**

The Channels panel lists all channels in the image—composite channel first (for RGB, CMYK, and Lab images). A thumbnail of the channel’s contents appears to the left of the channel name; the thumbnail is automatically updated as you edit the channel.



*Channel types*

**A.** Color channels **B.** Spot channels **C.** Alpha channels

## Display the Channels panel

1. Choose Windows > Channels.

## Resize or hide channel thumbnails

1. Choose Panel Options from the Channels panel menu. Click a thumbnail size or click None to turn off the display of thumbnails.

Viewing thumbnails is a convenient way of tracking channel contents; however, turning off the display of thumbnails can improve performance.

---

## Show or hide a channel

You can use the Channels panel to view any combination of channels in the document window. For example, you can view an alpha channel and the composite channel together to see how changes made in the alpha channel relate to the entire image.


1. Click in the eye column next to the channel to show or hide that channel. (Click the composite channel to view all default color channels. The composite channel is displayed whenever all the color channels are visible.)

**Note:**

To show or hide multiple channels, drag through the eye column in the Channels panel.

### Show color channels in color

Individual channels are displayed in grayscale. In RGB, CMYK, or Lab images, you can view the individual channels in color. (In Lab images, only the *a* and *b* channels appear in color.) If more than one channel is active, the channels always appear in color.

You can change the default to show the individual color channels in color. When a channel is visible in the image, an eye icon  appears to its left in the panel.

1. Do one of the followings:
  - In Windows, choose Edit > Preferences > Interface.
  - In Mac OS, choose Photoshop > Preferences > Interface.

Select Show Channels in Color and click OK.

### Select and edit channels

You can select one or more channels in the Channels panel. The names of all selected, or *active*, channels are highlighted.



### *Selecting multiple channels*

**A.** Not visible or editable **B.** Visible but not selected for editing **C.** Selected for viewing and editing **D.** Selected for editing but not viewing

- To select a channel, click the channel name. Shift-click to select (or deselect) multiple channels.
- To edit a channel, select it and then use a painting or editing tool to paint in the image. You can paint on only one channel at a time. Paint with white to add the selected channel's color at 100% intensity. Paint with a value of gray to add the channel's color at a lower intensity. Paint with black to fully remove the channel's color.

---

### **Rearrange and rename alpha and spot channels**

You can move alpha or spot channels above the default color channels only if the image is in Multichannel mode (Image > Mode > Multichannel). For information about that mode's limitations.

To change the order of alpha or spot channels, drag the channel up or down in the Channels panel. When a line appears in the position you want, release the mouse button.

#### **Note:**


Spot colors are overprinted in the order of their appearance from top to bottom in the Channels panel.

- To rename an alpha or spot channel, double-click the channel's name in the Channels panel, and enter a new name.

---

### **Delete a channel**

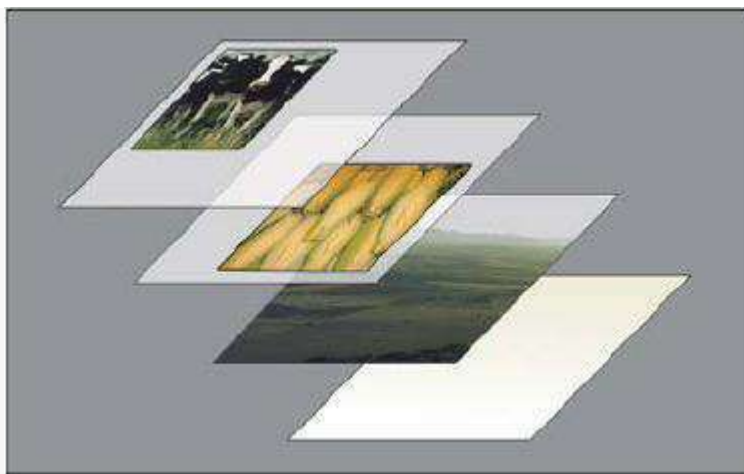
You may want to delete spot or alpha channels you no longer need before saving an image. Complex alpha channels can substantially increase the disk space required for an image.

1. In Photoshop, select the channel in the Channels panel and do one of the followings:
  - Alt-click (Windows) or Option-click (Mac OS) the Delete icon .

- Drag the channel name in the panel to the Delete icon.
- Choose Delete Channel from the Channels panel menu.
- Click the Delete icon at the bottom of the panel, and then click Yes.

### **About Photoshop layers**

Photoshop layers are like sheets of stacked acetate. You can see through transparent areas of a layer to the layers below. You move a layer to position the content on the layer, like sliding a sheet of acetate in a stack. You can also change the opacity of a layer to make content partially transparent.



*Transparent areas on a layer let you see layers below.*

You use layers to perform tasks such as compositing multiple images, adding text to an image, or adding vector graphic shapes. You can apply a layer style to add a special effect such as a drop shadow or a glow.

### **Introduction to layers**

#### **Organizing Photoshop layers**

A new image has a single layer. The number of additional layers, layer effects, and layer sets you can add to an image is limited only by your computer's memory.

You work with layers in the Layers panel. *Layer groups* help you organize and manage layers. You can use groups to arrange your layers in a logical order and to reduce clutter in



the Layers panel. You can nest groups within other groups. You can also use groups to apply attributes and masks to multiple layers simultaneously.

### **Photoshop layers for non-destructive editing**

Sometimes layers don't contain any apparent content. For example, an *adjustment* layer holds color or tonal adjustments that affect the layers below it. Rather than edit image pixels directly, you can edit an adjustment layer and leave the underlying pixels unchanged.

A special type of layer, called a *Smart Object*, contains one or more layers of content. You can transform (scale, skew, or reshape) a Smart Object without directly editing image pixels. Or, you can edit the Smart Object as a separate image even after placing it in a Photoshop image. Smart Objects can also contain smart filter effects, which allow you to apply filters non-destructively to images so that you can later tweak or remove the filter effect.

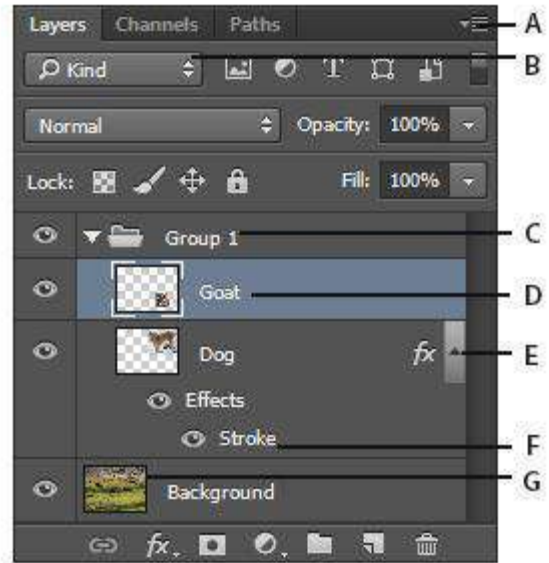
### **Video layers**

You can use video layers to add video to an image. After importing a video clip into an image as a video layer, you can mask the layer, transform it, apply layer effects, paint on individual frames, or rasterize an individual frame and convert it to a standard layer. Use the Timeline panel to play the video within the image or to access individual frames.

---

### **Photoshop Layers panel overview**

The Layers panel in Photoshop lists all layers, layer groups, and layer effects in an image. You can use the Layers panel to show and hide layers, create new layers, and work with groups of layers. You can access additional commands and options in the Layers panel menu.



*Photoshop Layers panel*

**A.** Layers panel menu **B.** Filter **C.** Layer Group **D.** Layer **E.** Expand/Collapse Layer effects **F.** Layer effect **G.** Layer thumbnail

### Display the Photoshop Layers panel

1. Choose Window > Layers.

### Choose a command from the Photoshop Layers panel menu

1. Click the triangle in the upper-right corner of the panel.

### Change the size of Photoshop layer thumbnails

1. Choose Panel Options from the Layers panel menu and select a thumbnail size.

### Change thumbnail contents

1. Choose Panel Options from the Layers panel menu and select Entire Document to display the contents of the entire document. Select Layer Bounds to restrict the thumbnail to the object's pixels on the layer.

### Note:

Turn off thumbnails to improve performance and save monitor space.

### Expand and collapse groups

1. Click the triangle to the left of a group folder.

### Filter Photoshop layers

At the top of the Layers panel, the filtering options help you find key layers in complex documents quickly. You can display a subset of layers based on name, kind, effect, mode, attribute, or color label.



*Filter layers options in the Layers panel*

1. Choose a filter type from the pop-up menu.
2. Select or enter the filter criteria.
3. Click the toggle switch to switch layer filtering on or off.

---

## **Convert background and Photoshop layers**

When you create a new image with a white background or a colored background, the bottommost image in the Layers panel is called *Background*. An image can have only one background layer. You cannot change the stacking order of a background layer, its blending mode, or its opacity. However, you can convert a background into a regular layer, and then change any of these attributes.

When you create a new image with transparent content, the image does not have a background layer. The bottommost layer is not constrained like the background layer; you can move it anywhere in the Layers panel and change its opacity and blending mode.

### **Convert a background into a Photoshop layer**

1. Double-click Background in the Layers panel or choose Layer > New > Layer from Background.
2. Set layer options.
3. Click OK.

### **Convert a Photoshop layer into a background**

1. Select a Photoshop layer in the Layers panel.

2. Choose Layer > New > Background from Layer.

Any transparent pixels in the layer are converted to the background color, and the layer drops to the bottom of the layer stack.

**Note:**


You cannot create a background by giving a regular layer the name, Background—you must use the Background from Layer command.

### **Turn the background layer into a regular layer**


### **Duplicate Photoshop layers**

You can duplicate layers within an image or into another or a new image.

### **Duplicate a Photoshop layer or group within an image**

1. Select a layer or group in the Layers panel.
2. Do one of the followings:
  - Drag the layer or group to the Create a New Layer button .
  - Choose Duplicate Layer or Duplicate Group from the Layers menu or the Layers panel menu. Enter a name for the layer or group and click OK.

### **Duplicate a Photoshop layer or group in another image**

1. Open the source and destination images.
2. From the Layers panel of the source image, select one or more layers or a layer group.
3. Do one of the following:
  - Drag the layer or group from the Layers panel to the destination image.
  - Select the Move tool , and drag from the source image to the destination image. The duplicate layer or group appears above the active layer in the Layers panel of the destination image. Shift-drag to move the image content to the same location it occupied in the source image (if the source and destination images have the same pixel dimensions) or

to the center of the document window (if the source and destination images have different pixel dimensions).

- Choose Duplicate Layer or Duplicate Group from the Layers menu or the Layers panel menu. Choose the destination document from the Document pop-up menu and click OK.
- Choose Select > All to select all the pixels on the layer and choose Edit > Copy. Then choose Edit > Paste in the destination image. (This method copies only pixels, excluding layer properties such as blending mode.)

### **Create a new document from a Photoshop layer or group**

1. Select a layer or group from the Layers panel.
  2. Choose Duplicate Layer or Duplicate Group from the Layers menu or the Layers panel menu.
  3. Choose New from the Document pop-up menu and click OK.
- 

### **Sample from all visible Photoshop layers**

The default behavior of the Mixer Brush, Magic Wand, Smudge, Blur, Sharpen, Paint Bucket, Clone Stamp, and Healing Brush tools is to sample color only from pixels on the active layer. This means you can smudge or sample in a single layer.

1. To smudge or sample pixels from all visible layers with these tools, select Sample All Layers from the options bar.
- 

### **Change transparency preferences**

1. In Windows, choose Edit > Preferences > Transparency & Gamut; in Mac OS, choose Photoshop > Preferences > Transparency & Gamut.
2. Choose a size and color for the transparency checkerboard or choose None for Grid Size to hide the transparency checkerboard.
3. Click OK.

## Photoshop actions:

The first thing you need to do is to drag-and-drop a Lens Distortions filter into your photo project... Once you've done that, simply apply the Control action to the filter, and you're all set.

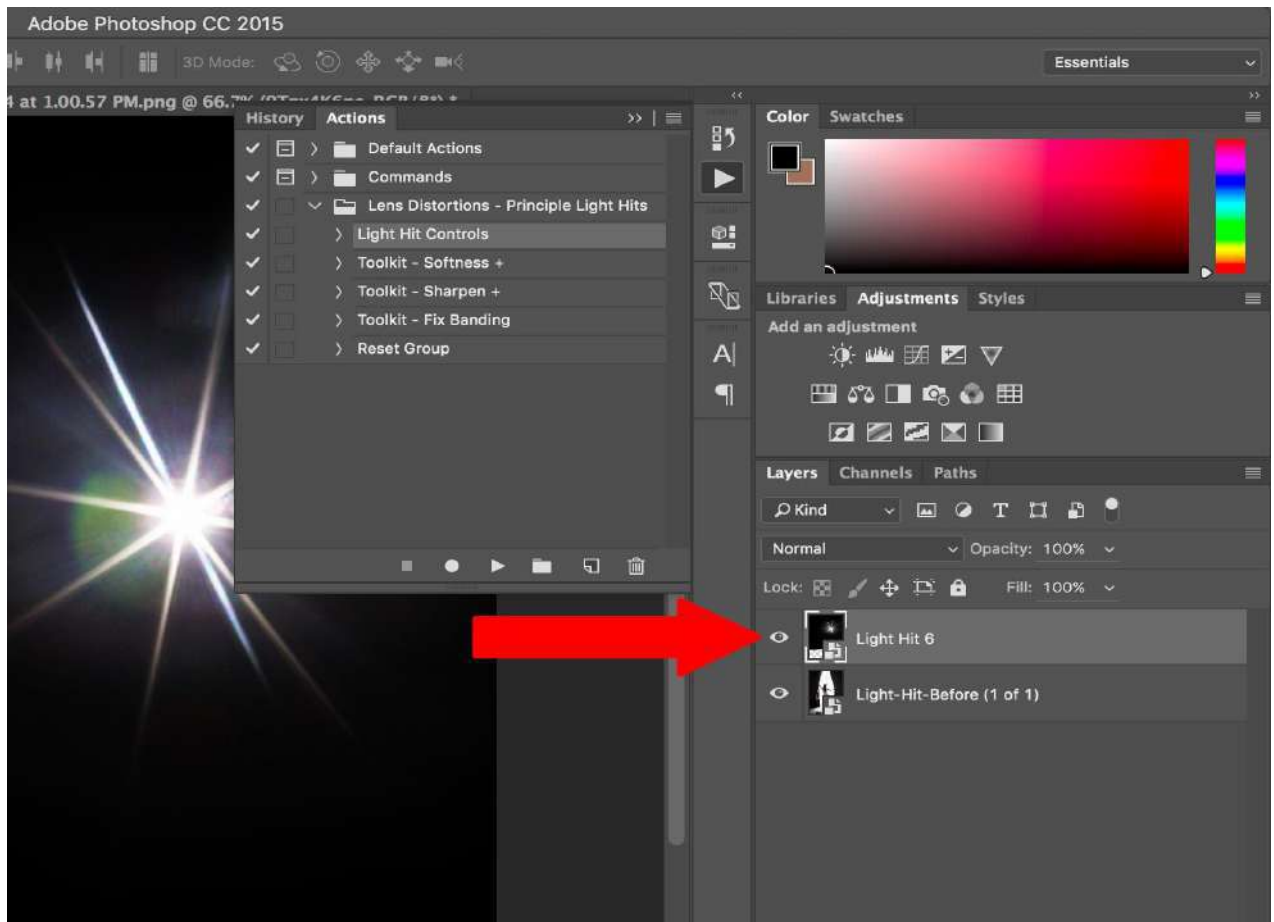
There are 5 different controls in each of our Photoshop actions:

- Controls (i.e. “Light Hit Controls”)
- Toolkit – Softness +
- Toolkit – Sharpen +
- Toolkit – Fix Banding
- Reset Group



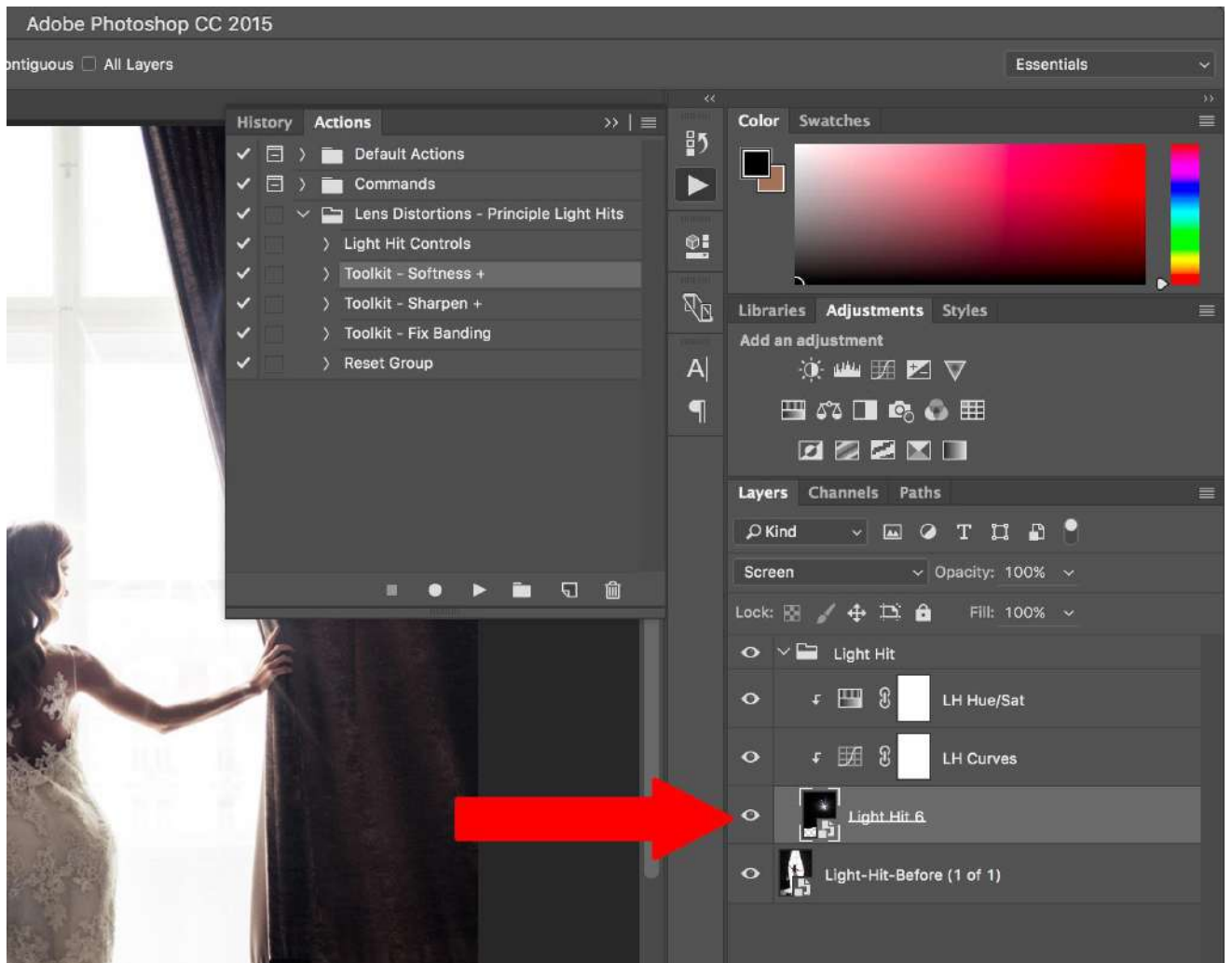
## Controls

This is the primary action you'll use and applies all the default Lens Distortion settings to your filter. To use it, select the Lens Distortions filter you want to apply it to, then click the little “play” button in the actions panel.



### **Toolkit – Softness +**

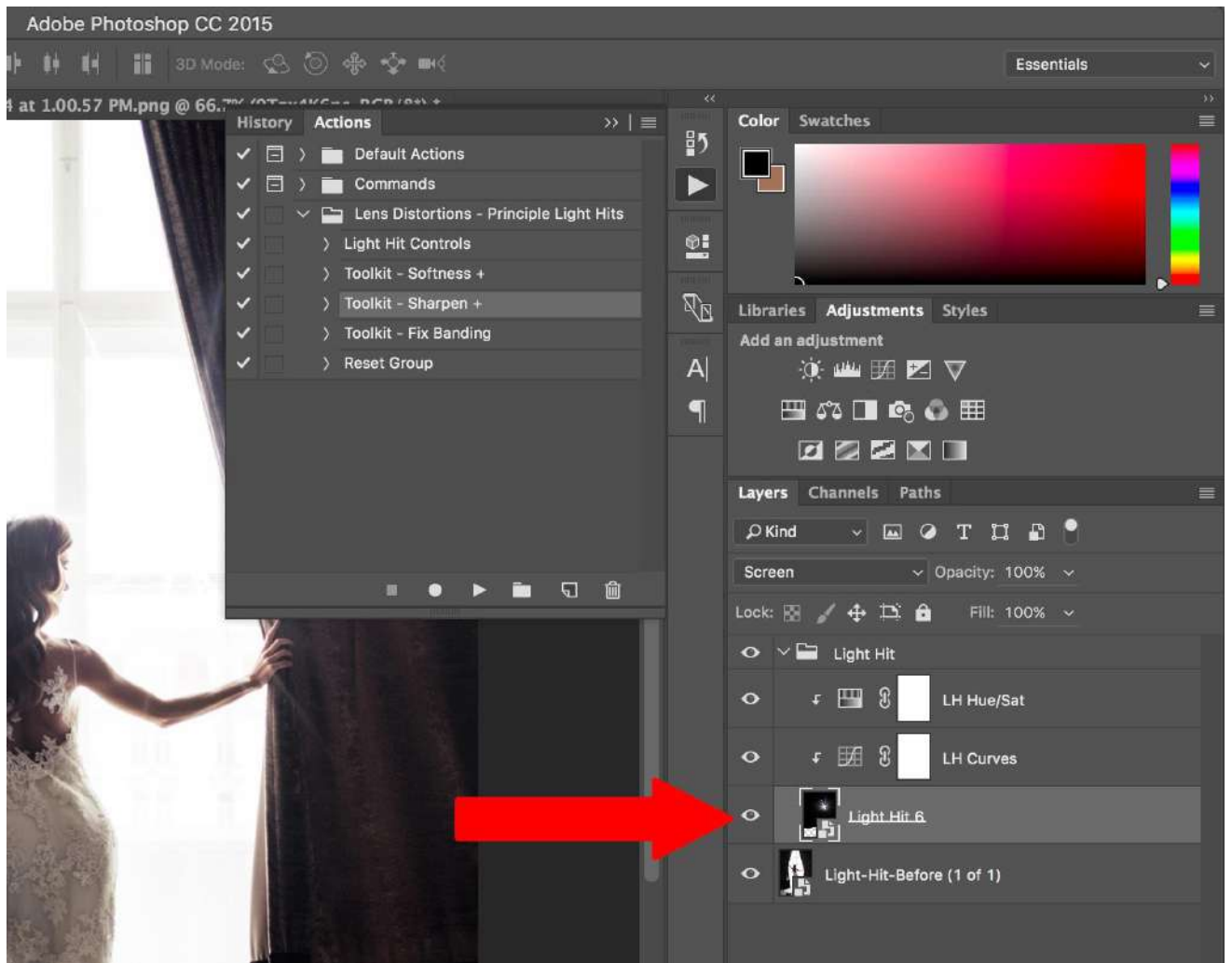
Once you've applied the Controls action above, you can use this action to add some Softness to the filter. Open the main filter folder and apply this effect directly to the filter.



### **Toolkit – Sharpen +**

Once you've applied the Controls action above, you can use this action to add some Sharpness to the filter. Open the main filter folder and apply this effect directly to the filter.



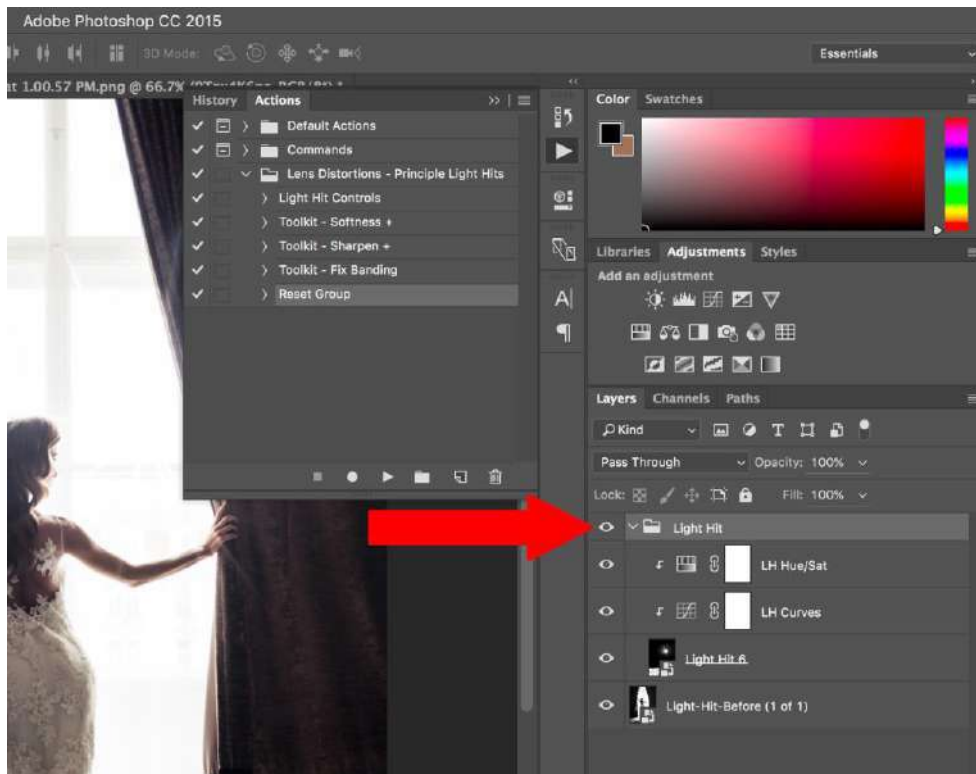


### **Toolkit – Fix Banding**

If you are running into any banding, chances are Photoshop defaulted your project to 8 Bit. Apply this action to update your project to 16 Bit.

### **Reset Group**

If you need to start over, apply this action to the main filter folder to reset all its effects.



**You may have already used a filter or two in Photoshop (perhaps as a step in an earlier chapter). In this chapter, filters are the star players.** Depending on which filters you apply and which settings you choose, the results can range from a subtle change to a total morph. You can make an image look (almost) as if it's hand painted, silk-screened, or sketched; apply distortion; add a pattern, texture, or noise; create a mosaic or a patchwork of tiles—the creative possibilities are infinite. Once you start using the Filter Gallery, you'll see ... time will fly by.



*A This is the original image.*



*B We applied the Charcoal filter.*

### **Applying filters**

You can apply filters to a whole layer or just to a selection on a layer. Most of the Photoshop filters are applied either via the Filter Gallery or via an individual dialog. A small handful of them, such as Clouds and Blur, are applied in one step simply by choosing the filter name from a submenu on the Filter menu. If you apply a filter to a Smart Object, it becomes an editable, removable Smart Filter.



If you try to select a filter and discover that it's not available, the likely cause is that it's incompatible with the current document color mode or bit depth. All the Photoshop filters are available for RGB files, most of the filters are available for Grayscale files, fewer are available for CMYK Color, Lab Color, and 16-bits-per-channel files, still fewer are available for 32-bits-per-channel files, and none are available for Bitmap and Indexed Color files.

Most of the Photoshop filters are housed conveniently under one roof in the Filter Gallery dialog. There you can preview dozens of filters and filter settings, show and hide each filter effect that you've already previewed, and change the sequence in which Photoshop applies them to your document.

### **To use the Filter Gallery**

1. Click an image layer; or for more flexibility, click a duplicate image layer or a Smart Object (see "To apply a Smart Filter" on page 344).
2. *Optional:* To limit the filter to a specific area of the image, create a selection.
3. The Foreground and/or Background colors are used by many filters and you must choose those colors now, before opening the Filter Gallery.
4. Choose Filter > **Filter Gallery**. The resizable gallery opens.




5. To change the zoom level for the preview, click the Zoom Out button  or Zoom In button  in the lower-left corner of the dialog, or choose a preset zoom level from the menu. (If the preview is magnified, you can drag it in the window.)
6. Do either of the following:


In the middle pane of the dialog, click an arrowhead to expand any of the six filter categories, then click a filter thumbnail.

Choose a filter name from the menu below the Cancel button.


7. On the right side of the dialog, choose settings for the filter.
8. To edit the list of effects (bottom right portion of the dialog), do any of these optional steps:

To apply an additional filter effect, click the **New Effect Layer** button, , click a filter thumbnail in any category, then choose settings. The effect may take a moment or two to process.

To **replace** one filter effect with another, click a filter effect name on the scroll list (don't click the New Effect Layer button), then choose a replacement filter and settings.

To **hide** a filter effect, click the visibility icon  next to the effect name (click again to redisplay).

To change the **stacking** position of a filter effect to produce a different result in the image, drag the effect name upward or downward on the list.

To remove a filter effect from the list, click it, then click the **Delete Effect Layer** button. 

9. Click OK.

- To hide or show the previews in the Filter Gallery for all but one filter effect, Alt-click/Option-click the visibility icon for that effect.
- To remove a non-Smart Filter, click a prior document state or snapshot on the History panel.
- In Edit/Photoshop > Preferences > Plug-Ins, uncheck Show All Filter Gallery Groups and Names to list, on the submenus on the Filter menu, only filters that are not in the Filter Gallery, or check this option to list all Photoshop filters on the submenus, including those that are available in the Filter Gallery (the gallery opens when you choose a filter name).

### **Filters that Use the Foreground and Background Colors**

The filters listed below use the current Foreground and/or Background colors. Some filters, such as Charcoal, Graphic Pen, and Photocopy (in the Sketch category), look good in the default Photoshop colors of black and white, whereas others look better in color.

Artistic > Colored Pencil (Background color), Neon Glow (Foreground and Background colors)

- Distort > Diffuse Glow (Background color)
- Pixelate > Point (Background color)
- Render > Clouds, Difference Clouds, Fibers (Foreground and Background colors)
- Sketch > Bas Relief, Chalk & Charcoal, Charcoal, Conté Crayon, Graphic Pen, Halftone Pattern, Note Paper, Photocopy, Plaster, Reticulation, Stamp, Torn Edges (Foreground and Background colors)
- Stylize > Tiles (Foreground or Background color)
- Texture > Stained Glass (Foreground color)

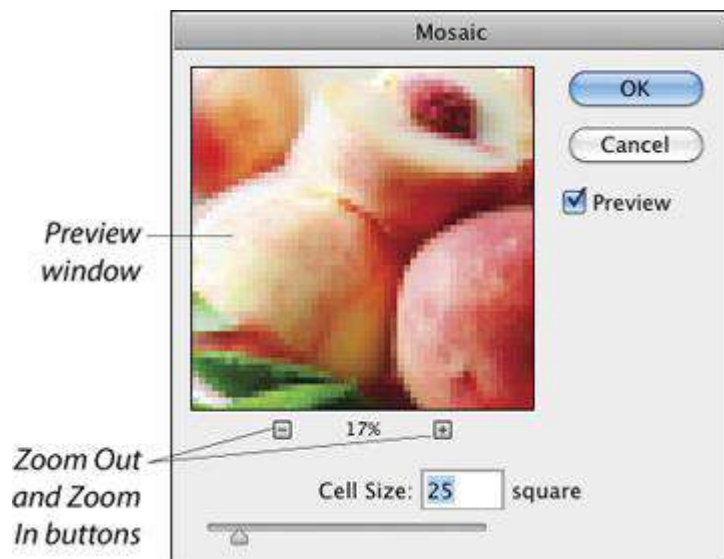
## Reapplying the Last Filter Quickly

- To reapply the last-used filter(s) using the same settings, choose Filter > [last filter name or Filter Gallery] (Ctrl-F/Cmd-F).
- To reopen either the last-used filter dialog or the Filter Gallery showing the last-used settings, press Ctrl-Alt-F/Cmd-Option-F.

## Preview Window

Some Photoshop filters are applied via an individual dialog (not via the Filter Gallery). Of those individual dialogs, some have a preview window and some do not.

- For individual filter dialogs that contain a preview window, you can click the + button to zoom in or the – button to zoom out (we usually do the latter). Most of the individual dialogs also have a Preview check box.
- In some filter dialogs (such as Blur > Gaussian Blur and Motion Blur), if you click in the document window (square pointer), that area of the image will appear in the preview window. You can drag the image inside the preview window.
- To compare the image with and without the current filter effect, click and hold on the preview, then release.



# UNIT – 4: MULTIMEDIA AUTHORING TOOLS

**Multimedia Authoring:** These are the tools which provide the capability for creating a complete multimedia presentation, including interactive user control, are called authoring tools/programs.

Some of the examples are:

- 1 Macromedia Flash.
- 2 Macromedia Director
- 3 Author ware
- 4 Quest.

One can also say that multimedia authoring is the creation of multimedia productions, sometimes called **Movies** or **Presentations**. As we are interested in computer science, we are mostly interested in **interactive applications**.

## **Introduction**

- Multimedia authoring tools provide the framework for organizing and editing the elements of a multimedia project.
- Authoring software provides an integrated environment for combining the content and functions of a project.
- It enables the developer to create, edit, and import data.

Authoring tools capability

Authoring tools should possess the following capabilities:

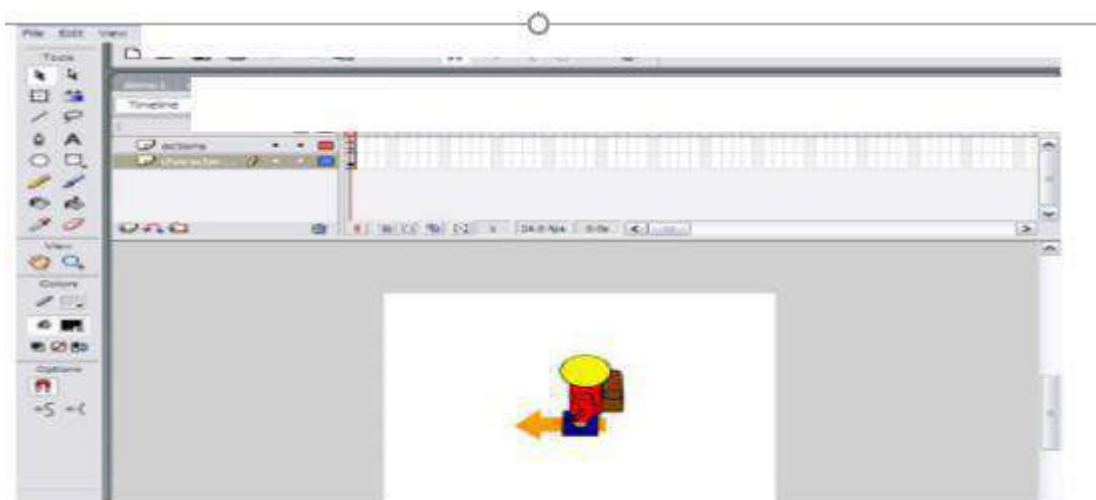
- 1 Interactivity
- 2 Playback
- 3 Editing
- 4 Programming / Scripting

- 5 Cross Platform
- 6 Internet Playability
- 7 Delivery/Distribution
- 8 Project organization

### **Types of authoring tools**

- Card. and page-based tools.
- Icon-based, event-driven tools.
- Time-based tools.
- Card- and page-based authoring systems
  - a) Card. and page-based authoring systems provide a simple and easily understood metaphor for organizing multimedia elements.
  - b) It contains media objects such as buttons, text fields, and graphic objects.
  - c) It provides a facility for linking objects to pages or cards.

Card based authoring system example:



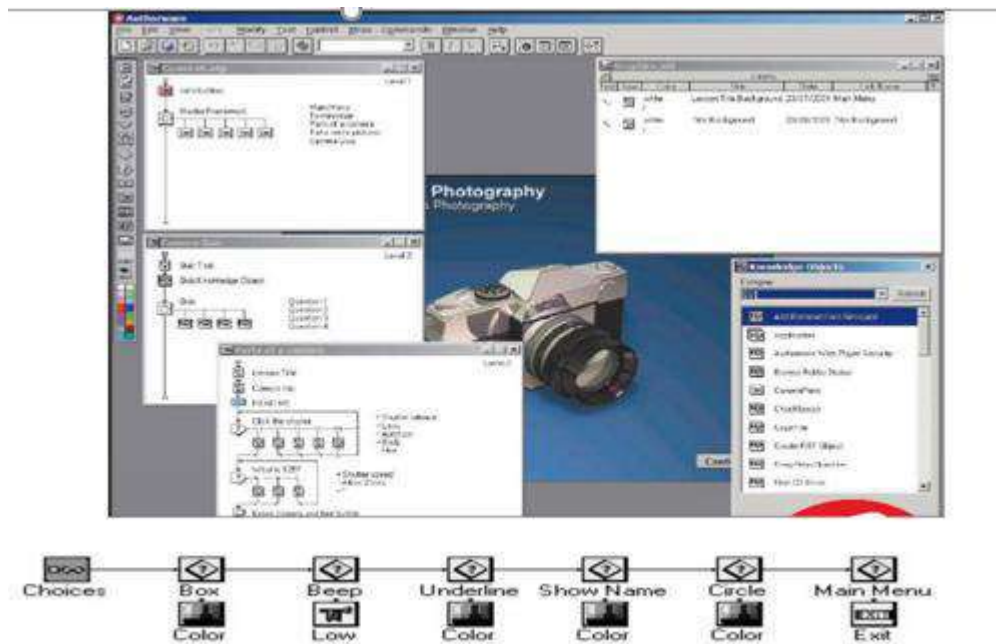
### **Example of authoring tools**

- HyperCard (Mac)
  - ToolBook (Mac / Windows)
- Icon-based, event-driven tools.
- Icon-based, event-driven tools provide a visual programming approach to organize and present multimedia.



- Multimedia elements and interaction cues are organized as objects in a flowchart.
  - Flowchart can be built by dragging appropriate icons from a library, and then adding the content.
    - examples of authoring tools
- Authorware(Mac/Windows)

IconAuthor (Windows) Icon based authoring system example:



### Time-based authoring tools

- Time-based tools are best suited for messages with a beginning and an end.
- Some time-based tools facilitate navigation and interactive control.
- Macromedia's Director and Flash are time-based development environments.

## ActionScript

ActionScript is an object-oriented programming (OOP) language that is designed specifically for Web site animation. Originally released with Macromedia Flash 4 and enhanced for Flash 5, ActionScript is a sophisticated version of the script language introduced in Flash 3. ActionScript makes it possible for developers to create onscreen environments (such as games, tutorials, and e-commerce applications) that can respond to user input through the keyboard or mouse. ActionScript is an *event-based* language: just as is the case in real life, actions are triggered by events.

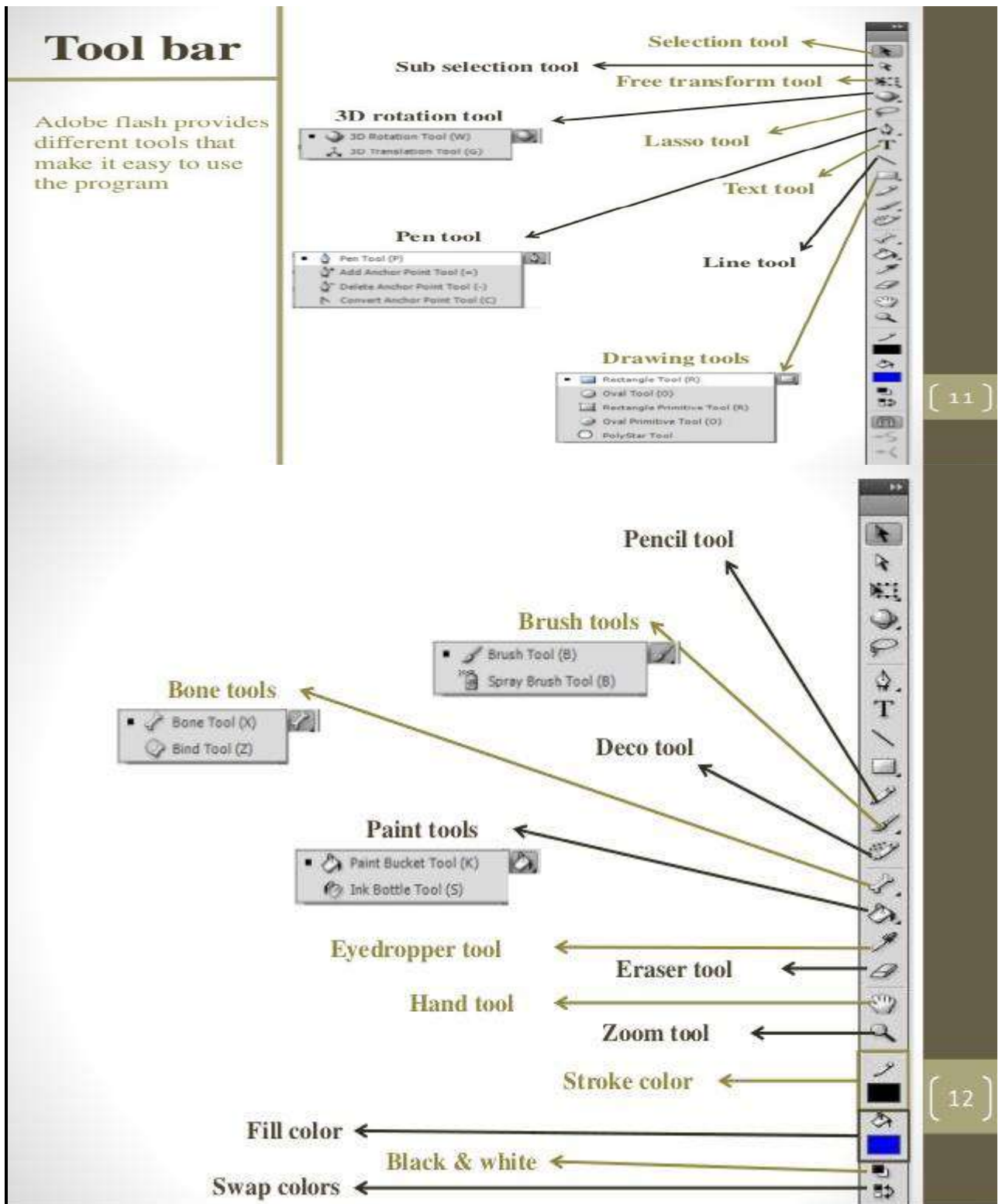
### Components of an effective API management strategy

We're using more APIs than ever before, and you need one central location to monitor their security, application connections, and traffic. Open this guide to see what makes an effective API management strategy.

- Corporate E-mail Address:
- I agree to TechTarget's [Terms of Use](#), [Privacy Policy](#), and the transfer of my information to the United States for processing to provide me with relevant information as described in our Privacy Policy.
- I agree to my information being processed by TechTarget and its [Partners](#) to contact me via phone, email, or other means regarding information relevant to my professional interests. I may unsubscribe at any time.
- 

ActionScript was modeled on ECMA (European Computer Manufacturers Association)-262, an international standard for JavaScript. In the Flash 5 version, new ActionScript syntax, conventions, and features were introduced that make it similar to JavaScript, which in turn makes the language automatically familiar to most Web developers. Flash 5 also includes a

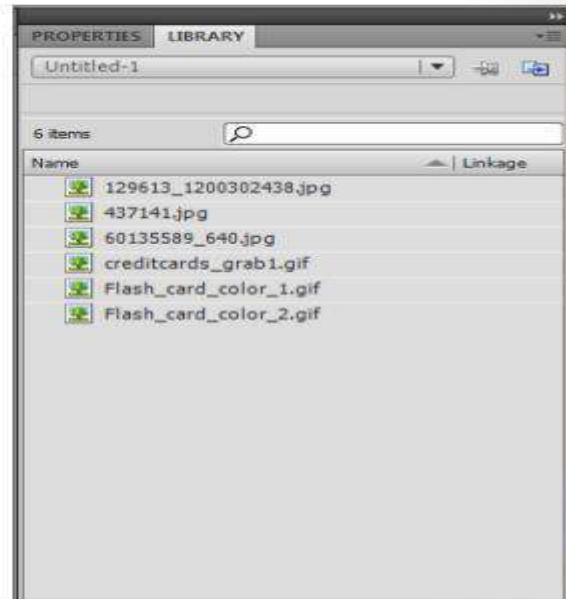
new ActionScript editing environment that automates editing tasks and reduces development time.



## Library:

The store of all the materials you use, either its an image, sound or video.

In next lesson you will learn how to import images into the library.



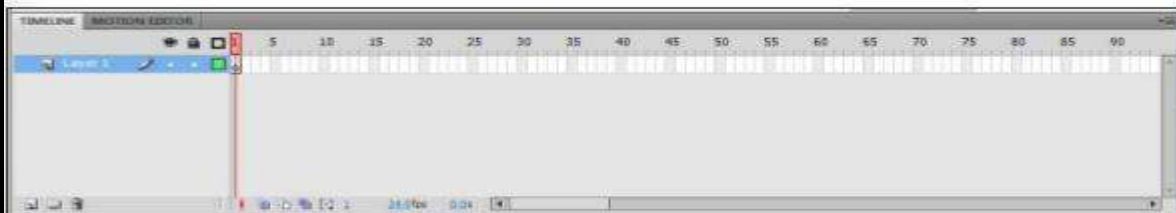
[ 13 ]

## Timeline:

We use the timeline to organize and control the flash contents (images, videos and sounds)

It contains two components:

- Layers
- frames



[ 14 ]

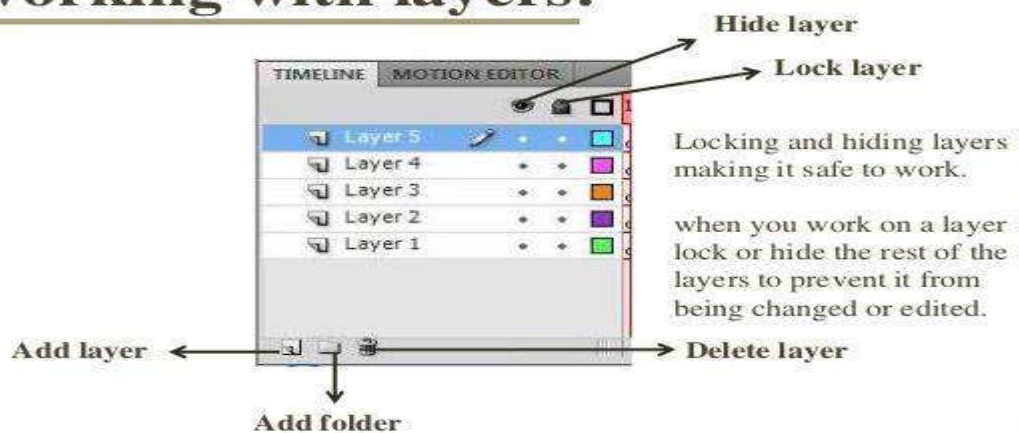
## Layers:

The flash system depends on the layers and each layer could have one or more objects.



[ 15 ]

## Working with layers:



We use folders to organize and gather the layers in case we have many layers

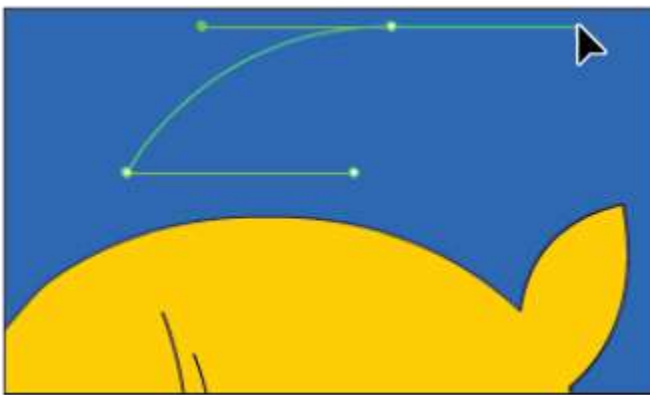
[ 16 ]

### **Adobe Flash Tutorial: Using the Pen Tool in Flash**

For precision illustration tasks, you will most likely want to use the Pen tool. The Pen tool allows for point-to-point drawing, and precise control over curves and lines in between. You can even add or remove points to fine-tune your work. If you've used the Pen tool in Illustrator CC, you'll already be familiar with the Pen tool and its related tools.

You'll use the Pen tool to create fins for your new fish in the following steps.

1. Select the Pen tool (🖋️) from the Tools panel. In the Properties Panel or Tools panel, set your stroke color to black (#000000).
2. In the space above your oval, click and release the mouse pointer on the Stage to create a new point. Move your pointer to the left of the point you just created and click and release again to create a second point. This point is joined to the first by a new path (line).
3. Position your cursor above and to the right of your last point. Click and hold down your mouse button, and then drag to the right. This forms a curve between your new point and the last one. Once you've gotten the curve just right, release the mouse button.



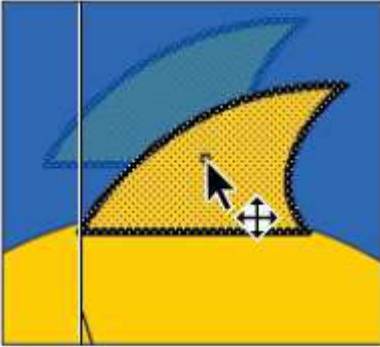
*Creating precision lines and curves using the Pen tool.*

You can create curves from any new point by holding down the mouse button and dragging in the direction you want to form the curve. (Be sure not to release the mouse button first!)



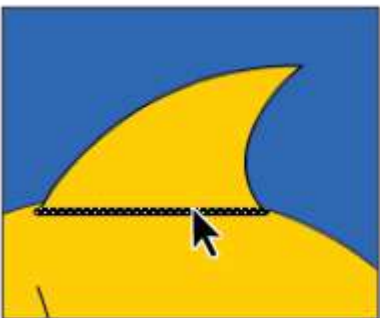
4. Next, you'll close up the shape. The next time you create a point, however, the Pen tool will attempt to draw a curve in the same direction as the last. To reset the last point drawn so that you can control the curve, click on the last point you created.
5. Move your pointer over the first point you created, and you should see a small loop appear below the pen cursor. Click and hold down your mouse button; drag to the right to form the final curve and release the mouse to complete the shape.
6. As with other path-based tools, shapes created with the Pen tool do not automatically fill. To fill the new shape, choose the Paint Bucket tool (🪣) from the Tools panel. In the Tools panel, make sure the Fill color is still set to the orange color labeled #FFCC00.
7. Click once inside your new shape to fill it with the currently active fill color.

- Now you'll move the fin into place and connect it with the rest of the body. Choose the Selection tool (⌘) and double-click the fill of the fin to select the entire shape. Drag it into place at the top of the oval, slightly overlapping it. Click the Stage to deselect the shape; when you deselect the shape, the two become merged.



*Move your new fin into place above the fish body.*

- The fin should now be merged with the oval. Use the Selection tool and click once to select the portion of the stroke that overlaps onto the oval. Only that portion should become selected. Press Backspace (Windows) or Delete (Mac OS) to clear away the selected stroke.



*Intersecting strokes in mergeable artwork become segmented and can be individually selected and removed.*

By default, strokes that overlap between two merged shapes become segmented, and individual portions can be selected and removed.



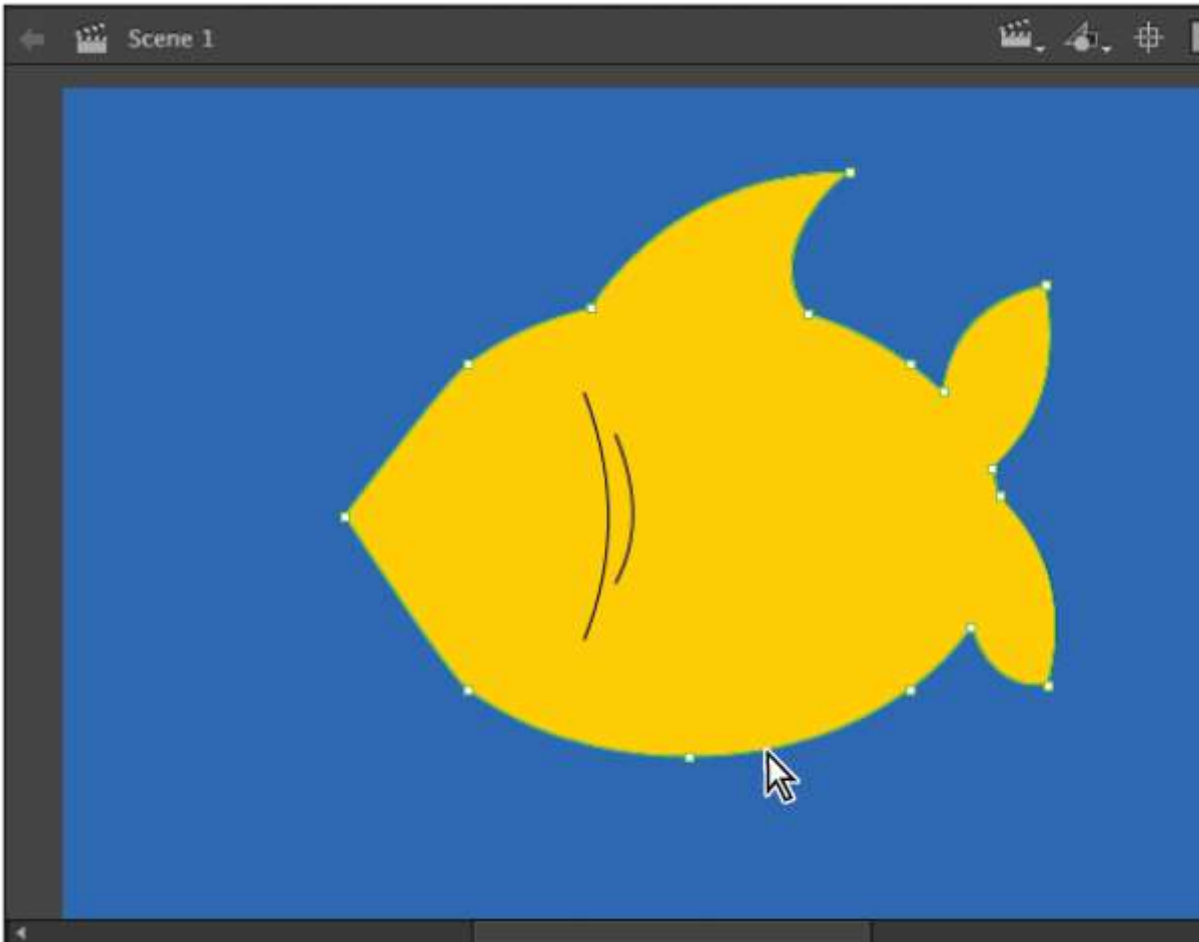
- Choose File > Save to save your file.

### **Using the Add and Delete Anchor Point tools**

You can add or remove points along existing paths with the Add and Delete Anchor Point tools. These tools are found under the Pen tool and enable you to further fine-tune your

illustrations. You'll add a bottom fin to your fish by manipulating the existing oval shape that forms its body.

1. Choose the Sub selection tool (⌘) from the Tools panel. Click on the edge of the oval; this reveals the points and paths that form this shape. From here, you can manipulate, add, or remove points along this path.

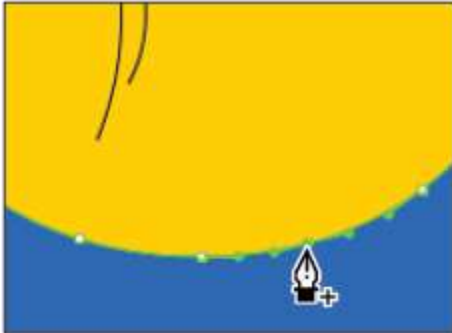


*Use the sub selection tool to activate the points and paths that compose a shape.*

2. Click and hold down your mouse pointer on the Pen tool (⌘)—this reveals the Add, Delete, and Convert Anchor Point tools. Choose the Add Anchor Point tool (⌘+). Note: You can also use the = and - keys to toggle between the Add and Delete anchor point tools.
3. At the bottom center of the oval, you'll notice a single anchor point. Using the Add Anchor Point tool, click once to the left and once to the right of that point to add two new anchor points.

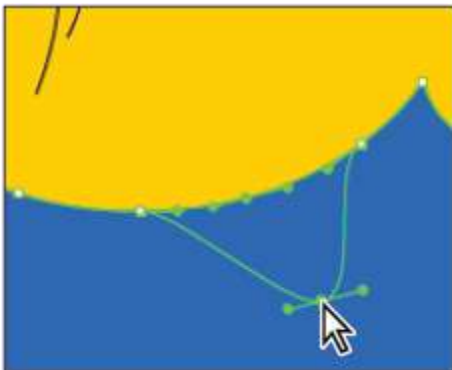


If you add the anchor point(s) in the wrong place, or add too many, choose the Delete Anchor Point tool (🔥) and click on any point to remove it.



*Use the Add Anchor Point tool to add two additional points surrounding the bottom point.*

4. Choose the sub selection tool and, if necessary, click on the outline of your oval to reactivate the points and paths. Click the point at the very bottom of the oval to activate it—the point now appears solid instead of hollow.
5. Click and drag the point down and to the right, which extends that portion of the oval into a fin-like shape.



*With more points in place, you can easily pull out and extend a fin from the existing shape.*

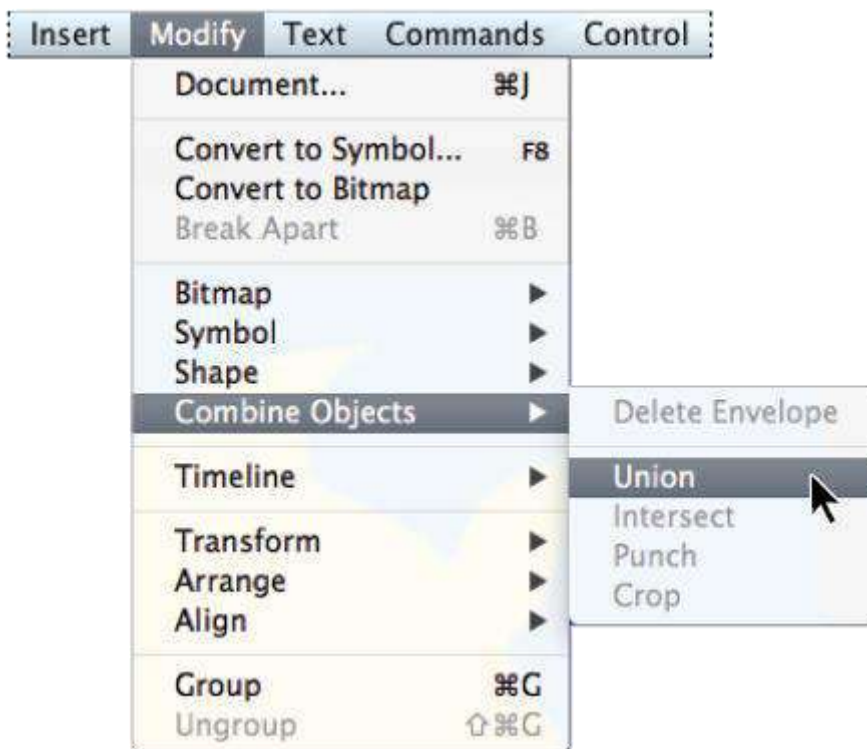
6. Choose File > Save to save your file and leave the file open.

### **Using the Combine Objects menu**

If you need to create more complex combinations of shapes, you can use the Combine Objects menu, found at Modify > Combine Objects. This menu enables you to create punches, crops, or intersections between overlapping shapes, and even lets you convert mergeable artwork into Drawing Objects.

Before you can perform any Combine Objects menu commands on a piece of artwork, it first must be converted to a Drawing Object. To do this, you'll use the Union command to convert your fish from mergeable artwork to a Drawing Object.

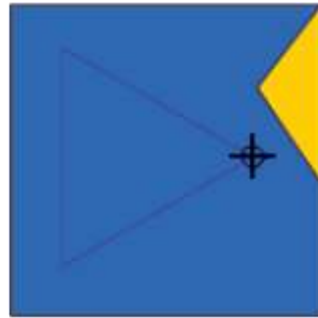
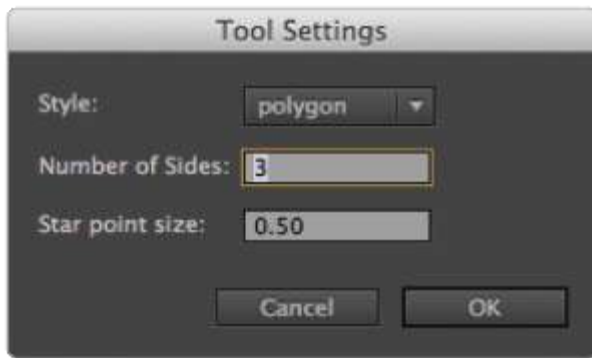
1. Select the entire fish by choosing Edit > Select All. You can also use the Selection tool (⬅) to draw a selection area around the artwork if you prefer.
2. Choose Modify > Combine Objects > Union. This command converts the selected artwork to a Drawing Object, and a bounding box appears around your fish and its parts. Choose Edit > Deselect All.



*Convert mergeable artwork to Drawing Objects using Modify > Combine Objects > Union.*

3. Select the Polystar tool (⬡), and enable Object Drawing mode by selecting the button at the bottom of the Tools panel. From the Properties Panel, press the Options button. This opens the Tool Settings dialog box for the Polystar tool.
4. In the Tool Settings dialog box, type 3 for the number of sides. Leave the Star point size at its defaults and press OK to exit the dialog box. On the Stage, while holding your Shift key (to constrain the angle), click and drag to draw a right-pointing triangle.

If the new triangle appears unfilled, select any fill color from the Tools panel, and use the Paint Bucket tool to fill it.

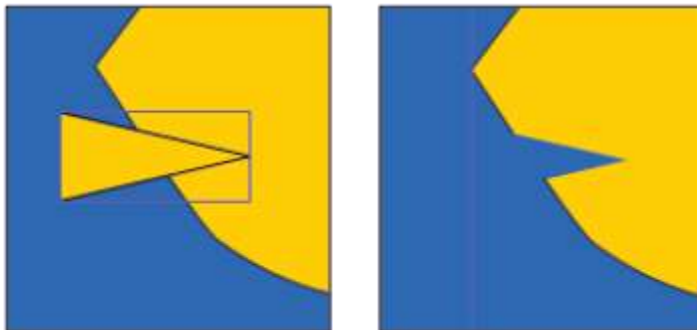


Use the Polystar tool to set and draw a triangle shape that you'll punch from the fish below.

5. With the shape still selected, choose the Free Transform tool (⌘+T) from the Tools panel. A bounding box with handles appears—grab the top middle handle and drag it downward to scale the shape down vertically.

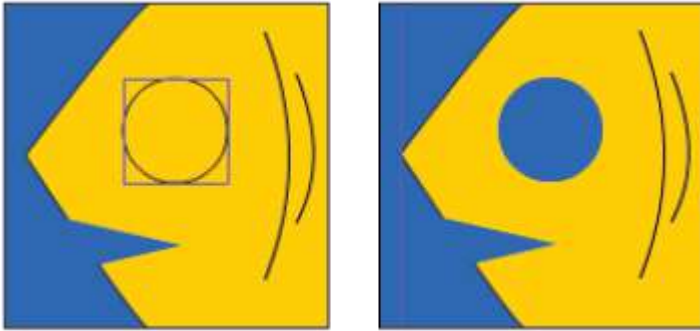
Choose the Selection tool and move the shape so that it overlaps the fish on the left where a mouth should be.

6. Choose Edit > Select All so that the new shape and your fish both appear selected. Choose Modify > Combine Objects > Punch. The new shape is knocked out from your fish, leaving behind a mouth-like opening.



Use Modify > Combine Objects > Punch to subtract one shape from another.

7. Select the Oval tool from the Tools panel. Make sure you have a fill color selected (any color will do). With the Shift key held down, click and drag to draw a small, perfect circle. To match the figure shown in this example, use your Property Inspector to set the circle to a width and height of 50. Switch to your Selection tool and position the circle on top of your fish above the mouth you created.
8. Choose Edit > Select All. With both the circle and fish selected, choose Modify > Combine Objects > Punch. This punches the circle into the body of the fish, making space for an eye.

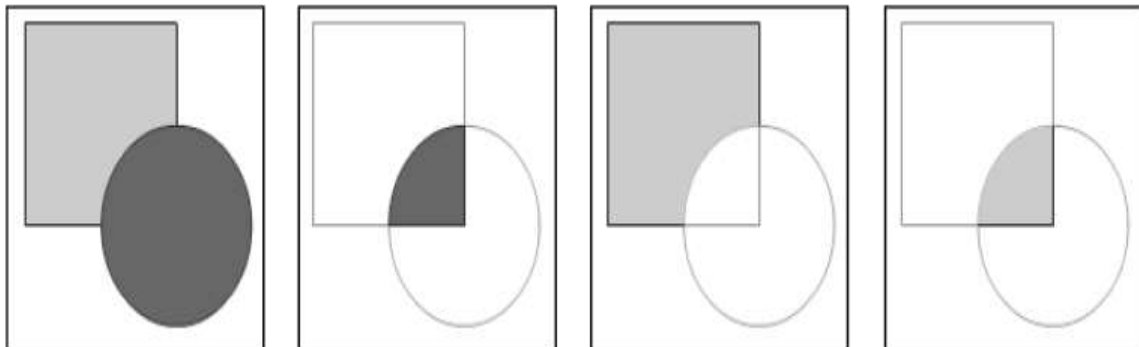


Use the *Punch* command to create a space for your fish's eye.

9. Choose File > Save to save your file.

### **The Combine Objects menu**

There are several commands available at Modify Combine Objects, not all of which you may use right away. Here's an overview of what each menu command does so that you can decide for yourself when and whether to use them.



*From left to right: Union, Intersect, Punch, Crop.*

*Union: Converts mergeable shapes into Drawing Objects. You can group several shapes into a single Drawing Object. In addition, shapes that are part of an Intersect, Punch, or Crop operation must all be Drawing Objects.*

*Intersect: Leaves behind only the overlapping area of two shapes.*

*Punch: Knocks out the top shape from the bottom shape.*

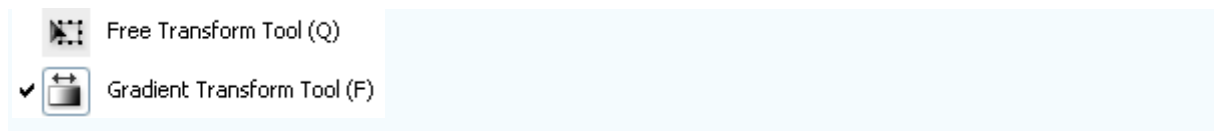
*Crop: Crops the bottom shape to conform to the top shape.*

In the tool section in the toolbar, we have different type of tool to make graphic and modification in that. All the tools are fully described below.

**Selection Tool:** Selection tool helps to select and modify the object shape. We can move any object which we have imported and drawn. We can delete the any part of any formatted

picture but remember the picture should be break (To make break picture, press Ctrl + B key).

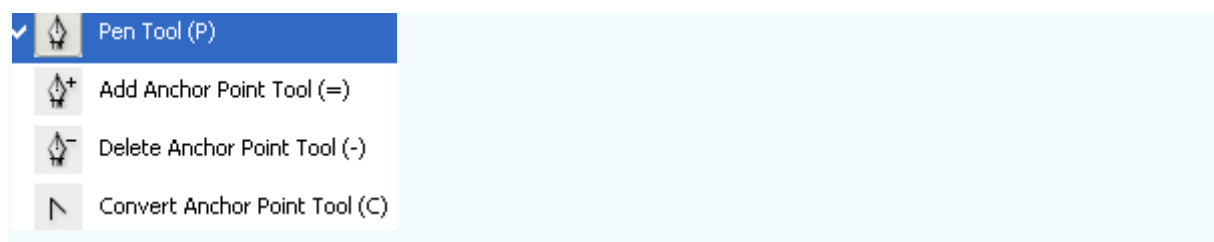
Subselection tool: When we need modification to make different shape of the drawn graphic in the document, we need a tool that is Subselection tool. It highlights the anchor point of the graphic by click on that to arrange point and get different shape as required.



Free Transform tool: Free Transform tool is very important tool, this tool helps us to scale, reflect of the picture and we can rotate the object from the particular point. we can also skew of object as per our requirement.

Gradient Transform Tool: With the help of this tool you can scale and rotate of the Gradient. When you fill the Gradient on object. Select the Gradient Transform tool or press F. As you click on Gradient Transform tool a bounding box will appear on the objects. then you can work this bounding box Ex. Moving the gradient, move the focal point. (focal point applicable only Radial gradient.), increase and decrease of gradient, rotate the gradient as required.

Lasso Tool: Lasso tool is a free hand selection tool. you can use this tool to select the object as you required and delete. when you select lasso option one magic wand tool and magic wand setting also appear down of the tool pallet, you can change the setting of this tool.



Pen Tool: Pen tool used to draw graphics by the anchor points and create line path in the document.

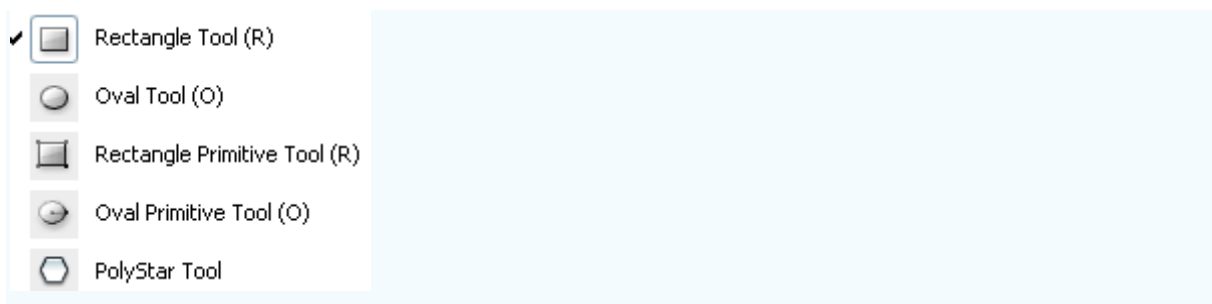
Add Anchor Point: We can add the more anchor point on the path and any graphic drawn by any tool by clicking on the path.

Delete Anchor Point: We can delete the anchor point on the same way by Clicking on the

Convert Anchor Point: We can modify the path with the help of this tool.

Text Tool: Select the text tool to type text on the document, click on the document with selected text tool a text area will appeared on the document, write the text on the box. You can scale the text area by scaling and more options will appear on the properties panel.

Line Tool: With the help of line tool we can also make the curve path and draw straight line. To make curve graphic we should draw a straight line then make curve by the help of Subselection tool.



**Rectangle Tool:** This section has different type of shape tool to draw shape on the document like Rectangle, Oval, Rectangle Primitive, Oval primitive and polyStar shapes. You can also modify the tool property to get different type of shapes.

**Pencil Tool:** This is a freehand tool that's used to drawing. when you click on the pencil tool more option will appear bottom of the tool pallet. you can create those shapes also.

**Brush Tool:** It's also a freehand tool and used to crate drawing and lines in the document. The different mode of the tool are also available in the flash to get different brush effect, select brush tool some more option will appear on the bottom of the toolbar make setting and apply on.

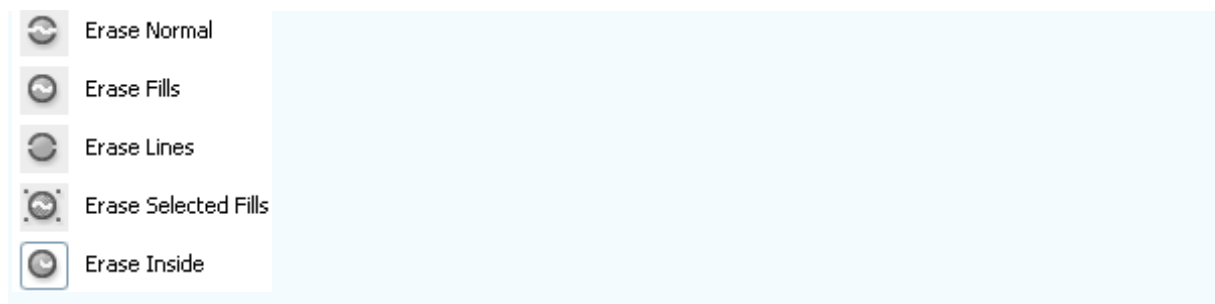


**Ink Bottle:** This tool is used to change the color of stroke color and other options.

**Paint Bucket:** The paint bucket helps us to fill selected color in the selected area and graphic, choose any color from the color palette and fill by the paint bucket tool.

**Eye Dropper Tool:** Eye Dropper tool helps to pick any color from the picture and graphic, As you want to fill a color that is not in the color palette, open colored picture and pick color by eye dropper tool and fill anywhere in the document as required.

**Eraser Tool:** This is an eraser tool we can erase the object and unwanted area in the document by five different ways, the list is given below.



**The view section in the toolbox, we have two tools those are used to move the document and zoom picture, the tools are described below:**

**Hand Tool:** With the help of hand tool, in the zoom mode of the document it is used to move document on any side or just hold down the space bar and move file with the help of the mouse.

**Zoom tool:** With the help of zoom tool, we can see the the object and picture in the zoom mode and modify on the right place as required and come back to actual size press double click on the zoom tool.

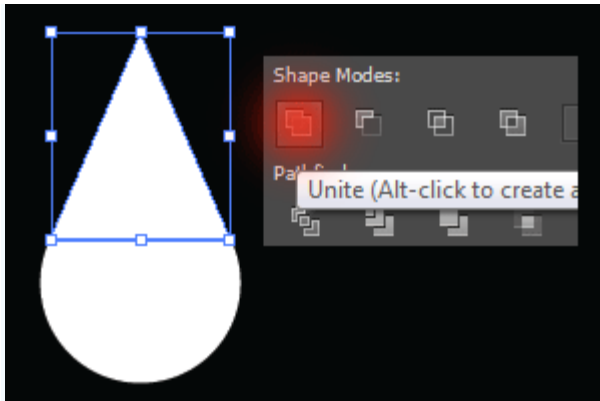
The color section in the toolbox, we have two tools those are used to fill stroke and foreground. The tools are described below.

**Stroke color:** When we draw any graphic by the help of any tool, the stroke automatically appears with the graphic with same graphic color. If we want to draw graphic with different stroke color, select stroke color before drawing.

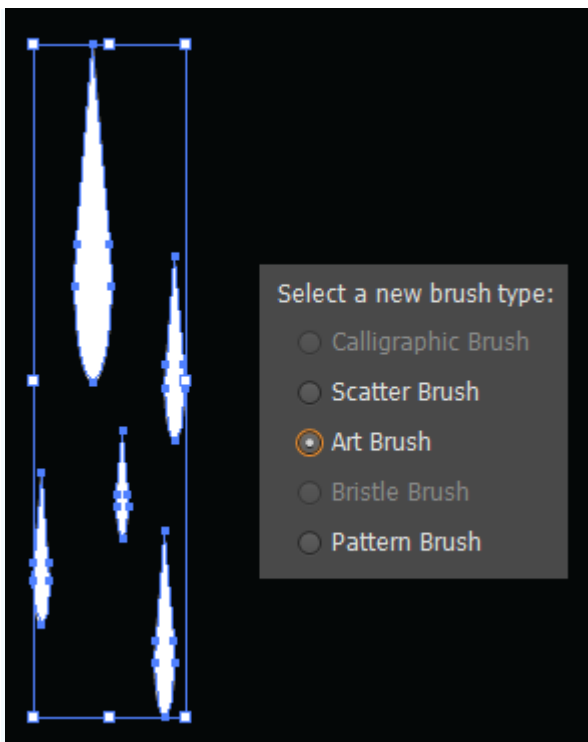
**Fill Color:** First choose foreground color as you want to fill in the graphic then draw graphic on the document, it can also change after the drawing but the tool is given in the toolbar so make first color setting.

In this tutorial we are going to teach you how to create a firework in the night by using illustrator, It will be drawn by using brush, pen tool etc. it can be use in the celebration greeting cards and offer banner background.

Take a new document with appropriate size as you need and draw a circle and tringle shape with white fill color, Open pathfinder panel by click Ctrl + Shift + F9. Select both shape and merge them by click on the "Unite" button as showing by red spot.

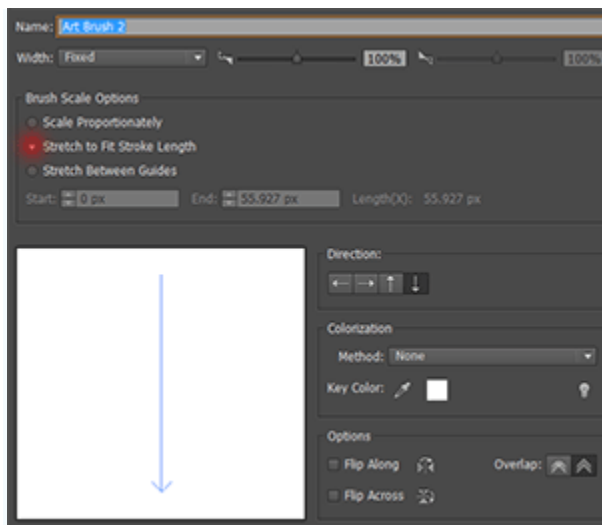


Still selected drop shape and make some more copies with different size and place them on different position. Select all the drop shape and shrink them horizontally. Drag and drop in the brush panel and choose "Art Brush".





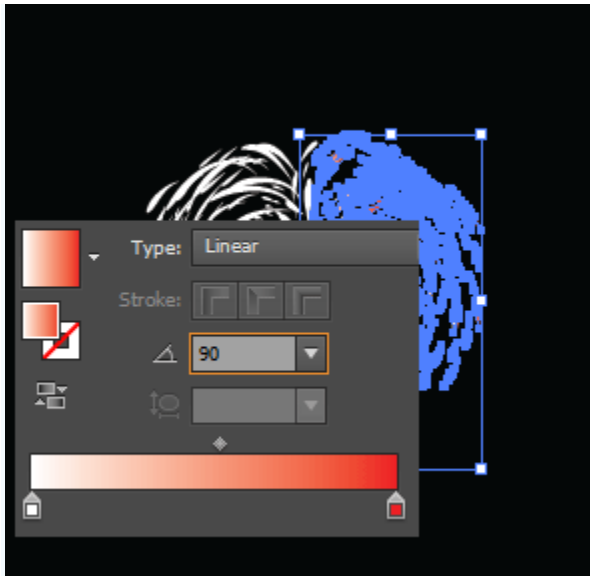
After choose "Art Brush" a window come and chose "Sketch to fit stroke length" as red spot and make other settings.



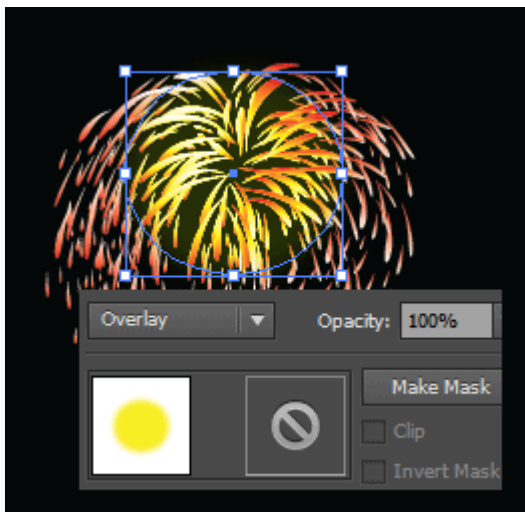
There is no need of drops shape, you can remove them. Select Pen tool (P key) and draw some lines, select all the lines and apply brush by click on the brush as we created earlier.



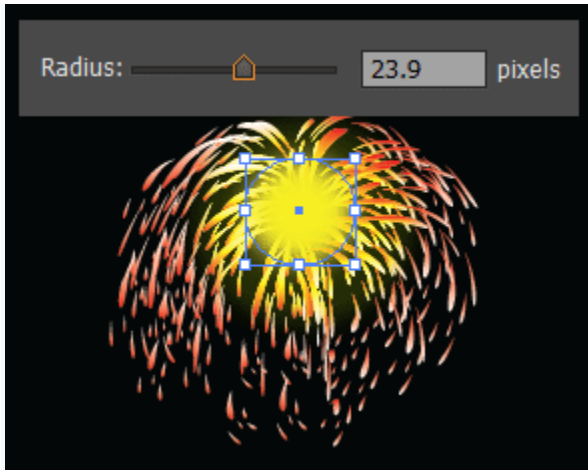
Select half portion of the fireworks and apply gradient color, open gradient panel and choose different color on the slider then change the angle. Apply same on the rest half portion but angle should be change "-90°".



Select Ellipse tool (L key) and draw a circle shape with fill "yellow" color, go to Effect menu > Blur > Gaussian Blur and put "radius" "20px". go to Window menu > Transparency and change blend mode "Normal" to "Overlay".



Still selected circle, make duplicate and decrease the size. Change the blend mode "Overlay" to "Normal" and apply gaussian blur "23.9px". Draw a little circle in the middle of yellow spot with "White" color and apply same blur effect.



This is the final result.



### Using the Drawing Tools

There are a few key concepts to grasp about how Flash creates artwork. Shapes drawn in Flash consist of strokes, fills, or both. Strokes are outlines, and fills are interiors. Even in shapes containing both, strokes and fills are independent of each other. That allows you to set their attributes separately, and even to move them independently to create unusual shapes.

Overlapping shapes drawn on the same layer interact. One shape drawn on top of another replaces any portions of the original shape that it obscures. Shapes of the same color merge where they touch, while shapes of different colors remain distinct although overlapped portions are replaced. Lines drawn by the Pencil, Line, Brush, Oval, or Rectangle tools are broken into segments where they intersect other shapes and bisect shapes beneath them.

You can take advantage of these unusual behaviors to create interesting negative shapes. To prevent shapes from interacting, group them by selecting them and choosing Modify, Group, or place them on separate layers.

When you draw, Flash sets the stroke color, fill color, or both according to the colors that have been selected in the Stroke and Fill boxes in the Toolbox, in the Property Inspector as you select a tool, or in the Color Mixer panel. Colors set in any of these locations are reflected in the others. Any shape drawn can have a stroke, but only shapes with interiors have fills.

### **The Line Tool**

The Line tool is the most basic of the drawing tools. It draws straight lines or, given what you know about how Flash creates artwork, strokes. Simply select the Line tool; then click and drag to create straight lines. Shift+dragging draws lines that are vertical, horizontal, or diagonal.

You can, of course, use the Line tool to draw closed shapes such as squares or rectangles that can have fills applied to them. However, you must manually add a fill, using the Paint Bucket tool, after you have closed a shape.

### **The Pencil Tool**

The Pencil tool enables you to draw freeform lines and shapes. Select the Pencil tool and then select a pencil mode in the Options section of the Toolbox to straighten, smooth, or maintain rough shapes (using the Ink setting) as you draw. Click and drag to sketch lines that mirror your mouse movements. Shift+drag to draw horizontal or vertical lines. As with the Line tool, you can use the Pencil tool to draw closed shapes but must use the Paint Bucket tool to manually add fills to them.

### **The Oval and Rectangle Tools**

The Oval and Rectangle tools allow you to draw simple shapes quite easily. Click and drag with either tool to draw ovals and rectangles. Shift+drag to draw perfect circles or squares.

You have the option to draw rectangles with rounded corners. It's an imprecise task, but if you have the patience, you can produce good results. When using the Rectangle tool, select the Round Rectangle Radius modifier at the bottom of the Toolbox. Enter a number in the

corner radius field. Zero results in straight corners. Finding the best radius for a rounded corner is a matter of trial and error.

## **The Pen Tool**

The Pen tool is the most powerful drawing tool. If you haven't used an illustration program before, the Pen tool will take some getting used to, particularly the way it draws curves. All that power comes with a learning curve, but you'll be rewarded with precisely drawn curves and irregular shapes that cannot be achieved with any other tool.

The Pen tool draws by establishing anchor points that it then connects. Select the Pen tool and move your mouse pointer onto the Stage.

## **Drawing Line Segments**

To draw line segments, click to place anchor points as you draw. The first anchor point appears as a small hollow dot, which changes to a blue square as you draw additional anchors. Shift+clicking draws vertical, horizontal, or diagonal (45 degree) lines. You must end the anchoring process, indicating to Flash that you've completed a path, whether you draw an open or closed shape. To end an open path, either double-click the final anchor point, Cmd-click (Mac) or Ctrl+click (Windows) away from the path or click the Pen tool in the Toolbox. To end a closed path, hold the Pen over the first anchor point.

## **Drawing Curved Segments**

The true power of the Pen tool lies in its capability to draw mathematically precise curved segments. These curves are known as *Bezier curves*. To calculate arcs precisely, curves are defined by four elements: two anchor points and two control handles. To draw curves with the Pen tool, click and drag anchor points. Drag in the direction you want your curve to be drawn. The first time you click and drag, you'll see that your anchor point has a control handle. Click to create a second anchor point and drag in the opposite direction—away from the curve—to draw an arc.

The length and angle of the control handles determine the shape of the curve. It's best to complete a path before you adjust your curves. It can take quite a while to become proficient with Bezier curves, and only the most experienced computer artists can create them precisely as they draw. It's much more efficient, and less aggravating if you are less experienced, to quickly draw an approximation of your desired shape and then adjust the curves. To close a curved path, click on the initial anchor point and drag away from the curve.

## **Adjusting Anchor Points**

The fastest way to draw with the Pen tool is to complete a path and then adjust the anchor points. Anchor points in curved paths have control handles, and the anchor points of line segments are corner points. After you draw a shape, you can add or delete anchor and corner points, convert anchor points to corner points and vice versa, or move any existing points.

To adjust anchor points, you must first select them. The placement of anchor points may not be obvious in curved and nonuniform paths. Use the Subselection tool and click on a path to reveal your anchor points.

If you click directly on an anchor, you both select that point and reveal all others in the path. If you click on a path but not directly on an anchor point, you simply reveal the anchor points in the path and must then click directly on one to select it. Shift+click additional anchor points to add them to the selection. You can then click directly on an anchor to reveal any control handles. Click and drag anchor points to move them. You can also select an anchor point and use the arrow keys to nudge it.

As you edit shapes, you may want to change a corner point to a curve point to be able to add curves, or vice versa to straighten a shape. To convert a corner point to a curve point, Option-drag (Mac) or Alt+drag (Windows) to create a control handle. Drag the new control handle in the direction of the curve you want to draw. To convert a curve point to a corner point, select your path with the Subselection tool to reveal your anchor points. Then select the Pen tool and click a curve point to convert it.

You may also find that you need to either add detail to or simplify shapes as you edit them. Adding points allows you to refine a shape by adding detail, either additional curves or corners. To add an anchor point, click on an existing path with the Pen tool. To simplify a shape, you can delete anchor points. The simplest way to delete an anchor point is to click on it with the Subselection tool and press Delete. Or, choose the Pen tool and click once to delete a corner point or twice to delete a curve point.

## **Adjusting Segments**

Segments of paths can also be adjusted, independently of their anchor points. You may find that you need to refine a small portion of a curve. To adjust the size and angles of curves,

click and drag their control handles. Where two curved segments intersect, two control handles extend away from a common anchor point. Each handle controls a curve on either side of a common anchor point. Dragging one end of a double-control handle changes both curves on either side of the anchor point, which stays in place.

To adjust the arc of the curved segment between anchor points only, Option-drag (Mac) or Alt+drag (Windows) the one end of a double-control handle.

If a curve segment intersects a straight segment, you'll see a control handle just on the curve side of the anchor. Click and drag this handle with the Subselection tool to change the arc or click and drag the anchor point to move the curve.

You can also use the Arrow tool to move segments. As the Arrow tool is positioned over a path, a small corner or curve appears to the right of the arrow, indicating which kind of segment can be selected. When you select a curved segment, a small curve appears to the right of the arrow.

A right angle appears when you select a straight segment. Click and drag with the Arrow tool to move segments, leaving the anchor points in place.

## **The Brush Tool**

The Brush tool is the lone Flash painting tool, and the only tool that allows you to mimic brush strokes. You really need a pressure-sensitive tablet, one that allows you to use a stylus instead of a mouse, to get the most realistic brush strokes. Trying to paint with a mouse is like trying to sign a check with a bar of soap. Tablets are sensitive to changes in pressure as you paint and adjust the weight of your strokes accordingly.

Choose brush modifiers in the Options section of the Toolbox to adjust the size and shape of your brush. You can also choose among Brush modes that allow you to paint normally, fills only, behind objects, inside objects, or in selections. Combine brush strokes with more precisely drawn shapes to create interesting textures.

The position, size, and shape of the timeline can be manipulated to better suit your workflow, much like any other Flash window or panel. On a dual monitor system, the timeline can be exiled to the second monitor, together with all the panels leaving the stage clear and unencumbered for wild creativity.

- Move the timeline by dragging it by the Timeline Title Bar, which is the bar at the top that says timeline. If the timeline is docked, click anywhere in the gray area above the layer stack to undock the timeline and reposition it.
- If undocked, resize the timeline by dragging on the lower right corner (PC), or the size box (Mac), which is also in the right corner. If docked, drag the bar at the bottom of the timeline that separates the layers from the application window, either up or down.
- To resize the name and icon controls (either to accommodate longer names or to apportion more of the timeline to frames), click and drag the bar that separates the name and icon controls from the frames area.

**Layer**

**specifics**

Knowing how to work with layers makes the Flash creation process flow much more smoothly. By default, new layers are stacked on top of the currently active layer. To rearrange layers, click in the blank area (between the layer name and the layer toggle icons), and drag the Layer Bar to the desired position in the layer stack and release.

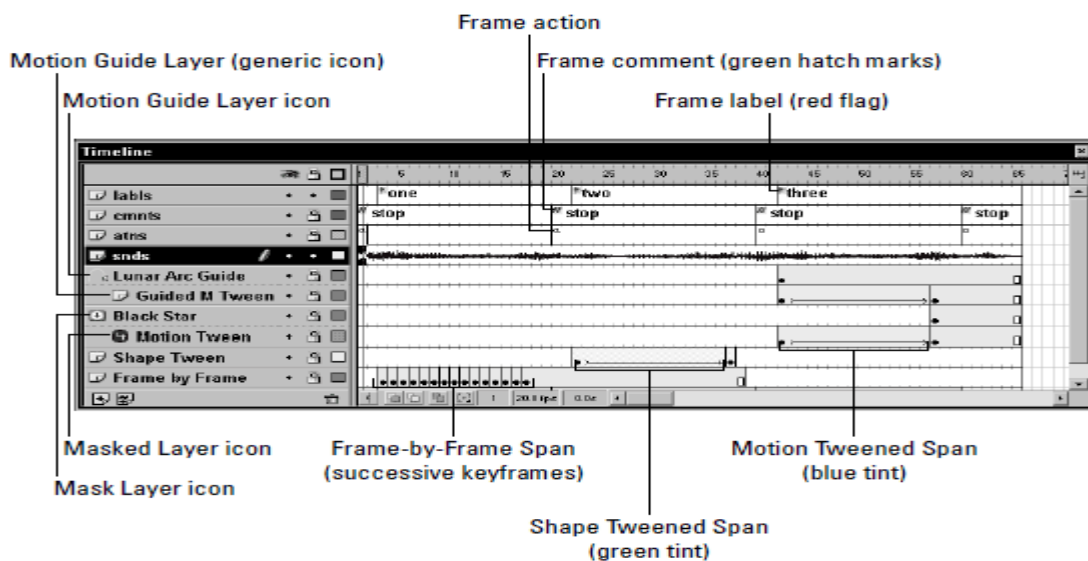
For enhanced functionality and control, as well as to enable reliable interactivity and ActionScripting, it's a good habit to give your layers meaningful names. Simply double-click the layer's name on the Layer Bar and enter a meaningful name.

**Timeline**

**specifics**

The new Flash 5 timeline still offers you many clues about what's going on with your animation, as shown below.

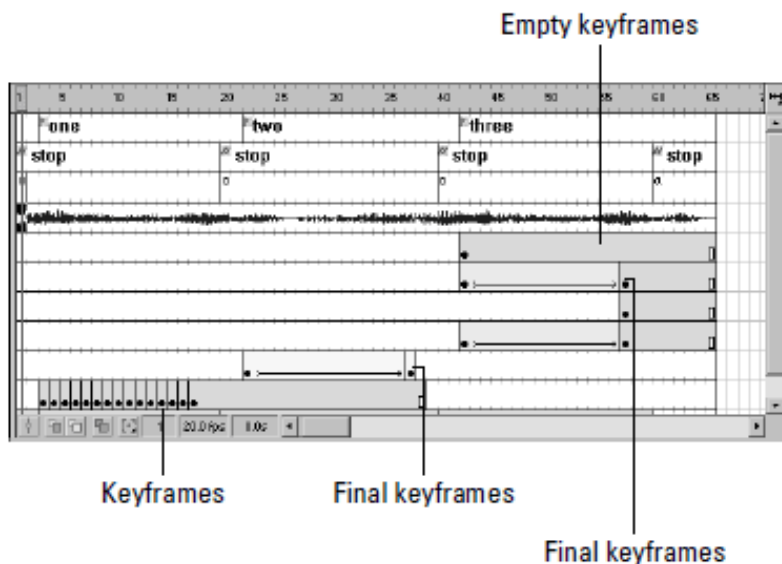
*Flash 5 Style Layer specifics*





- **Keyframe:** A keyframe is any frame in which the contents of the frame may differ from the contents of either the previous or subsequent frames. Solid circles designate keyframes with content.
- **Keyframe spans:** Keyframe spans newly designated in Flash 5 are the sections from one keyframe up to (but not including) the next keyframe, which are separated from each other by vertical lines. Thus, as shown, the span between frames 3 and 6 in the buttons layer is a keyframe span. Note that these spans can now be dragged intact, as a whole to a different location. This functionality is shown in the selected span between frames 8 and 13 in the buttons layer.
- **Final keyframe:** The final frame of a keyframe span with content is marked with an empty rectangle (that is, frame 6 of the buttons layer), and a vertical line to the right of the rectangle.

*Here's the same timeline that was shown above*



- **Intermediate frame(s):** The intermediate frames of a nonempty keyframe span are gray.
- **Empty span(s):** Empty spans are white (for example, the visible portion of the sweep mc layer).
- **Frame-by-Frame Animation:** Frame-by-Frame Animation is animation composed entirely of keyframes. In a Frame-by-Frame Animation, the contents of each individual frame differ from both the previous and subsequent frames.
- **Tweened Animation:** Tweened Animation is an animation where the movement or change is interpolated, or tweened, over a range of frames that extend between two

keyframes. An arrow stretching across a colored keyframe span designates a Tween, of which there are two varieties:

- **Motion Tweens:** Motion Tweens are indicated by a blue tint.
- **Shape Tweens:** Shape Tweens are indicated by a green tint.
- **Motion Guide Layer:** A Motion Guide Layer is used to guide an animated item along a path, which can be drawn with either the Pencil or the Line Tool.
- **Mask Layer:** A Mask Layer is a layer that is used to selectively obscure the layers beneath it.
- **Label:** Labels are used to give layers meaningful names, rather than using frame numbers. The advantage of this is that named layers can be moved without breaking ActionScript calls assigned to them. Upon export, Labels are included as part of the .SWF, so it makes sense to keep them short. Use the Frame Panel to add a Label to a selected frame. Press Enter/Return after typing a frame label or comment to ensure that the label takes.
- **Comment:** Comments are special Labels, preceded by a double-slash “//” Comments do not export, so you can be verbose (within the confines of the timeline) without adding to the .SWF. Use the Frame Panel to add a Comment, which is merely a label preceded by “//,” to a selected frame.
- **Waveform:** This squiggly blue line in the snds layer is the waveform of a placed sound.
- **Frame Actions:** The small a’s in frames 1, 20, 40, and 60 of the atns layer designate the presence of frame actions.

## General

## preferences

The General Tab of the Flash Preferences dialog, which is accessed from the Main Menu by choosing Edit⇨Preferences, has two sections specifically related to the timeline and its behavior in Flash 5. These are Timeline Options and Highlight Color.

## Timeline

## options

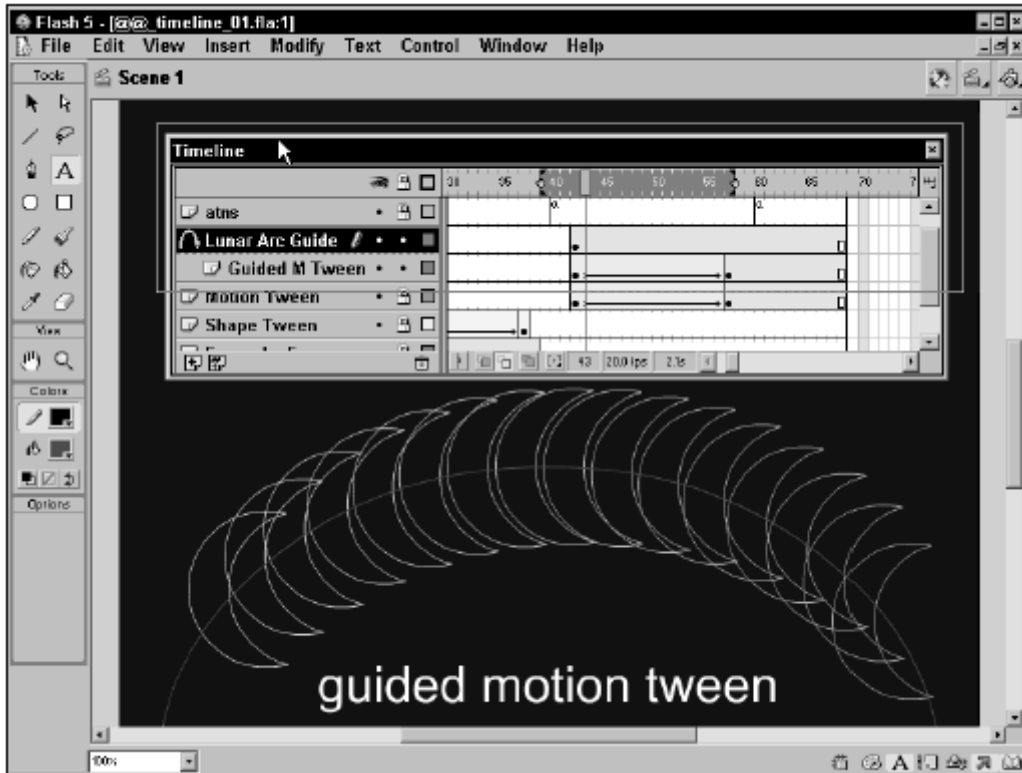
The Disable Timeline Docking option prevents the timeline from attaching to the application window after it’s been deployed as a floating panel.

On both the Mac and PC, to undock the timeline and deploy it as a floating palette as shown in Figure below, click the gray area to the left of the eyeball icon and then, with the mouse still depressed, drag the palette away from the application window.

To prevent the timeline from docking, press the Control key while dragging. To permanently disable timeline docking, use Edit⇨Preferences and, under Timeline Options, check the Disable Timeline Docking check box. The timeline can be dragged away from its docked

position by clicking the Timeline Header and dragging the timeline away from the edge of the Flash application.

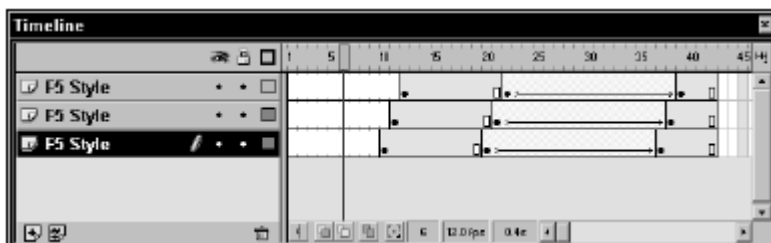
*The timeline deployed as a floating palette*



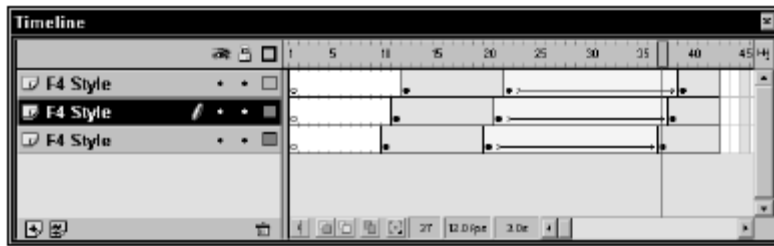
The next two options in the Preferences dialog let you revert to the Flash 4 timeline style:

- **Flash 4 Selection Style:** Flash 5 introduced a new methodology for selecting frames in the timeline. This option toggles that functionality back to Flash 4 Selection Style.
- **Flash 4 Frame Drawing:** Flash 5 also introduced a new methodology for drawing frames in the timeline. This option toggles that functionality back to the Flash 4 style. The difference between the timelines in Flash 5 and Flash 4 shows below

*Flash 5 frame drawing*



*Flash 4 frame drawing*



## Highlight color

The Highlight Color options in the Preferences dialog control which colors are used for selected objects:

- **Highlight Color:** This preference controls the highlight color for selected groups, symbols, or text excluding shapes.
- **Use this color:** Check this option to choose a Highlight Color for selections from the Swatches pop-up.
- **Use layer color:** Check this option to use the layer color as the Highlight Color for selections. This option enables you to distinguish selected items according to their associated layer color (which you set in the Layer Properties dialog).

## Layer Properties

Layer Properties dialog is most readily accessed by Right/Ctrl+clicking any Layer Bar and then choosing Properties from the layer contextual menu. It can also be invoked by choosing Modify⇔Layer.

## The layers contextual menu

As shown below, the layers contextual menu affords convenient access to a number of layer-specific operations, many of which are duplicated elsewhere.

- **Show All:** Shows all layers. If some layers have had their visibility turned off, this makes them all visible.
- **Lock Others:** Unlocks the active layer and locks all other layers.
- **Hide Others:** Makes the currently active layer visible, if it is not visible, and hides all others.
- **Insert Layer:** Inserts a new layer above the currently active layer.
- **Delete Layer:** Deletes the active layer.
- **Properties:** Invokes the Layer Properties dialog for the currently active layer.
- **Guide:** Transforms the current layer into a Guide Layer.
- **Add Motion Guide:** Inserts a new Motion Guide Layer directly above the current layer and automatically links the current layer to the Guided Layer.

- **Mask:** Transforms the current layer into a Mask Layer.
- **Show Masking:** Use this command on either the Mask or the Masked Layer to activate the masking effect essentially, this command locks both layers simultaneously, which enables the masking effect.

### The Layer Properties dialog

The Layer Properties dialog is used to control and edit the properties of the active layer and to facilitate routine layer operations.

- **Name:** Use this option to change the name of the layer.
- **Show:** With this option checked, the layer is visible; otherwise, it's hidden.
- **Lock:** This option enables you to lock or unlock the layer.
- **Type:** These options are used to set the type of layer:
  - **Normal:** This is the default, used for drawing and animation.
  - **Guide:** Guide Layers have two purposes. They can be used either as Motion Guides or as drawing guides. Guide Layers aren't exported, so they aren't visible and they don't add to the exported file size. Guided Layers are linked to a Guide Layer.

*This composite screen shot shows the layer contextual menu and ensuing Layer Properties dialog.*

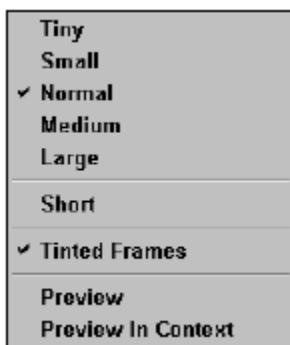


- **Mask:** A Mask Layer is used in conjunction with a Masked Layer to create special effects. The Masked Layer is hidden except beneath filled areas of the Mask Layer that it's linked to.
- **Outline Color:** Use this to choose the color of the layer's outlines.
- **View layer as outlines:** When this is checked, all items on the layer appear as outlines, according to the color chosen in the previous option. Viewing as outlines speeds the display while editing because all items are shown as thin outlines. As discussed in the previous section regarding General Preferences, this option can be used in conjunction with the Highlight Color options of the General tab of Edit ⇔ Preferences, to either give each layer a unique color, or to employ a global color for all outlines.
- **Layer Height:** Use this to increase the height of the *individual* layer. This means that you can have most layers displayed tiny, and yet have others display with more visible content. This is useful if you use the Preview or Preview in Context Timeline options on the Frame View options pop-up. It's also useful when viewing the waveforms of sound files.

**Frame View options**

As shown below, the Frame View options pop-up is used to customize the size, color, and style of frames displayed within the timeline. These features can prove very helpful when you are working with cartoon animation, and want to see each frame previewed. Or, if you are working on an extremely long project with a huge timeline, it can be helpful to tweak the size of the individual frames, so that you can see more of the timeline at a single glance.

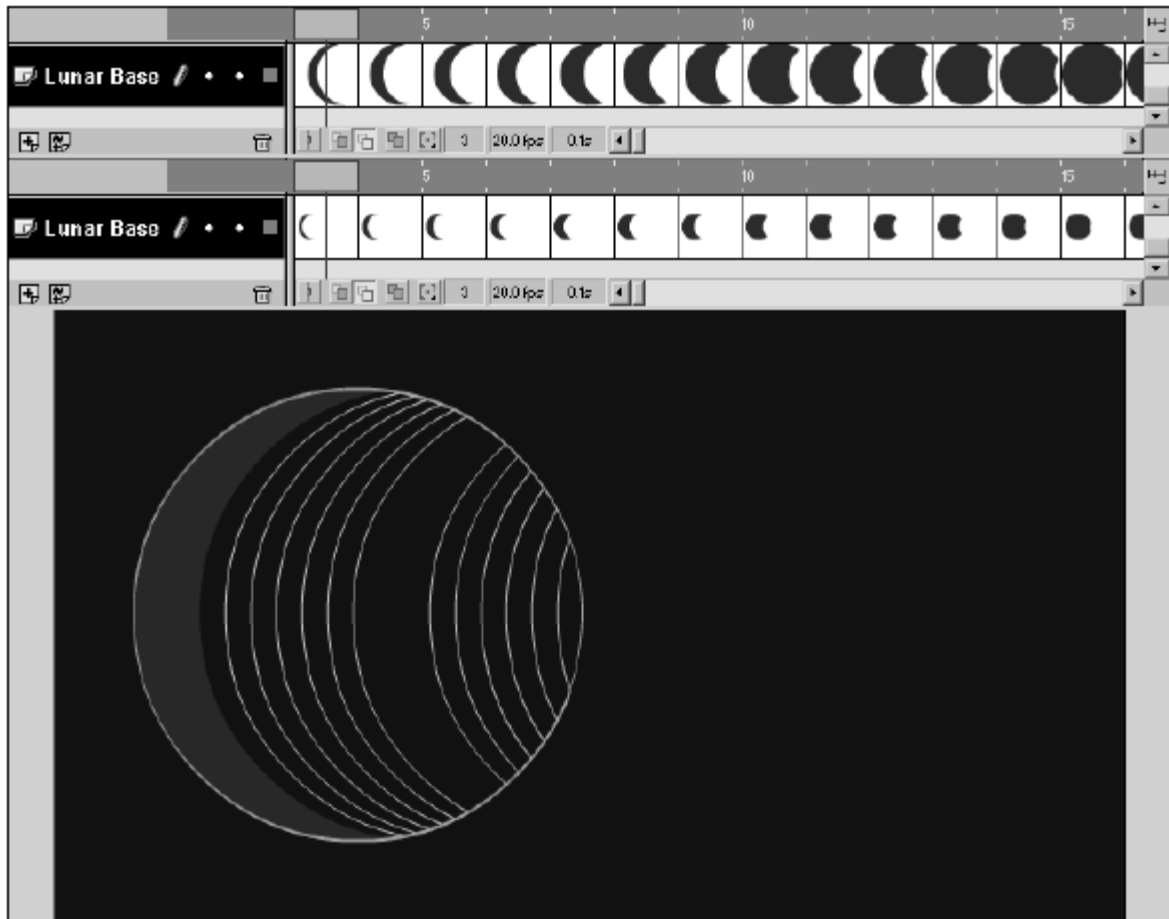
*The Frame View options pop-up is used to customize the size, color, and style of frames displayed within the timeline.*



When used in conjunction with the Layer Height option of the Layer Properties dialog, you can customize your timeline in myriad ways to better suit your particular project. Your options include:

- **Tiny, Small, Normal, Medium, Large:** These options afford a range of sizes for the width of individual frames. When working on extremely long animations, narrower frames facilitate some operations.
- **Short:** This option makes the frames shorter in height, permitting more Layer Bars to be visible in the same amount of space. When working with many layers, short layers help squelch the tedium of scrolling through layers of layers.
- **Tinted Frames:** This option toggles tinted frames on or off. With Tinted Frames on, the tints are as follows:
  - **White:** Empty or unused frames (for any layer). This is the default.  
The white color of empty or unused frames is unaffected regardless of whether Tinted Frames is on or off.
  - **Gray:** There are two kinds of gray frames: (a) The grayed-out gray frames in the default (empty) timeline are a quick visual reference that indicates every fifth frame, like the tick marks on a ruler. These tick frames appear regardless of whether Tinted Frames are enabled. (b) The solid gray color, which appears when Tinted Frames are enabled, indicates that a frame is either filled or otherwise used. Frame usage means that the frame has something in it, which may be either visible or invisible as, for example, an item with an alpha of 0 percent, or a hidden symbol.
  - **Blue:** Indicates a Motion Tween span.
  - **Green:** Indicates a Shape Tween span.
  - **A red arrow:** Indicates a Motion Tween span, when Tinted Frames are off.
  - **A green arrow:** Indicates a Shape Tween span, when Tinted Frames are off.
- **Preview:** As shown at the top of Figure below, the preview option displays tiny thumbnails that maximize the element in each frame. Thus, the scale of elements is not consistent from frame to frame. In this Frame-by-frame animation, the phases of the moon increase over a span of 15 frames.
- **Preview in Context:** As shown at the bottom of below one, when previewed in context, the same animation is seen with accurate scale from frame to frame (because elements are not maximized for each frame).

*In this composite screen shot, the Frames are shown with Preview option (top) and Frames shown with Preview in Context option (middle) for the same animation (bottom).*



## Scene and Symbol Bar

Nested between the Menu Bar and the top of the timeline is the Scene and Symbol Bar shown below. The Scene Name button, at the far left, indicates the name of the current scene. When in Symbol Editing Mode, click this button to return to the current scene. To the right is the Edit Scene button, and at the far right is the Edit Symbols button. Click either button to evoke a menu of scenes or symbols that are editable within the current movie.

### *The Scene and Symbol Bar*



Scenes are used to organize a Flash project into logical, manageable parts. By default, on export Flash plays back all of the scenes within the movie in the order in which they are listed in the Scene Panel.

To navigate to other scenes from within the Movie Editor:

- Click the Edit Scene button at the far right of the Scene and Symbol Bar, and then click the desired scene.



- Navigate to a specific scene from the View Menu with the View ⇄ Go To command. Use the Scene Panel, to manage your scenes. The Scene Panel may be accessed with either of these commands: Modify ⇄ Scene or Window ⇄ Panels ⇄ Scene.

### ***The Scene Panel***



When your movie is published to .SWF, the scenes play in the order in which they are listed in either the Scene Panel or the Scene pop-up.

- To delete a scene, either use the Scene Panel's delete button or, from the Insert menu, use the Insert ⇄ Remove Scene command.
- To add a scene, either use the Scene Panel's add button or, from the Insert menu, use Insert ⇄ Scene.
- Use the duplicate button on the Scene Panel to duplicate a scene.
- To rename a scene, simply double-click the scene name and type the new name.
- To rearrange scene order, simply click and drag a scene to alter its position with in the Scene Panel. You can use actions to force the movie to access scenes outside the default linear order

### **Color types overview**

In Flash there are three kinds of colors

- Normal colors (solid)
- Gradients (linear and radial)
- Bitmaps

Both RGB and HSB model is supported for colors.

### **Tools overview**

#### **Color related tools**

Flash CS6 has several color tools and controls

### **In the tools panel**

- Paint bucket and ink buckets
- Stroke color and fill color (for most tools). Select colors before you choose a tool to draw

### **In the properties panel**

- Stroke color and fill color

### **Color panel**

- Color selection

### **Swatches**

- Preset colors

### **How to use the color selection popups**

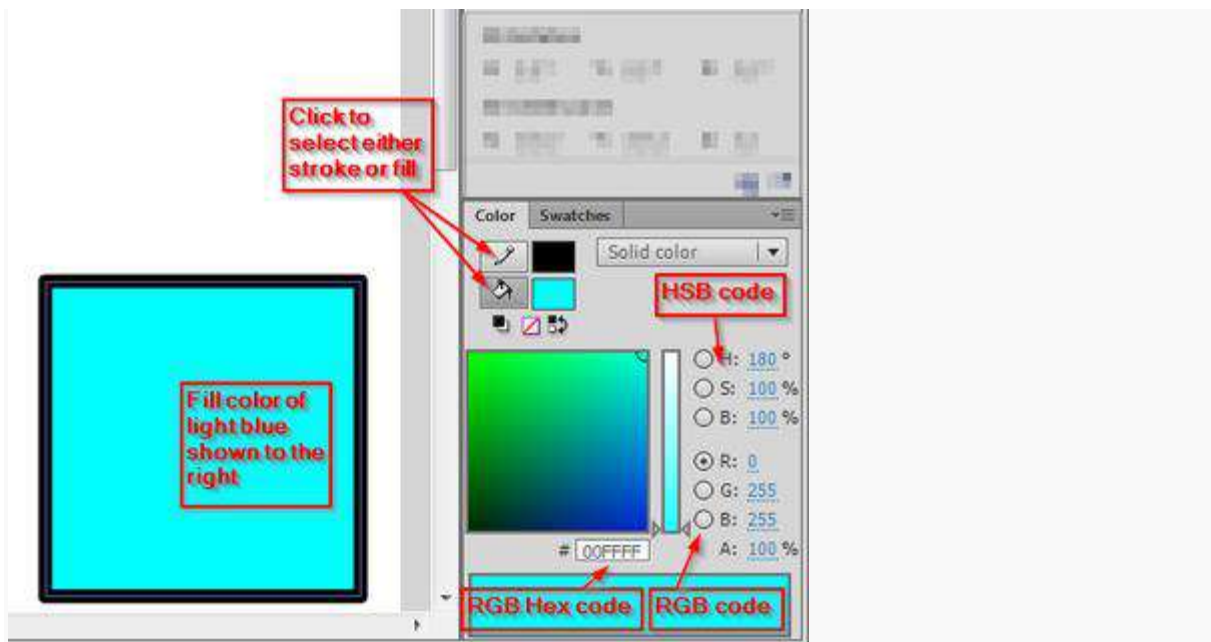
When you select (or change) fill or stroke color, a color popup swatches pane will pop up. You then can select a color with the eye-dropper tool or alternatively from any color in the Flash workspace.

You also can change alpha channel or type a 6-digit hexadecimal RGB Code (see color panel explanation below)

### **How to use the color and the swatches panels**

We recommend to have the color panel docked on top right, else get it with menu *Window-Color* (or *SHIFT-F9*).

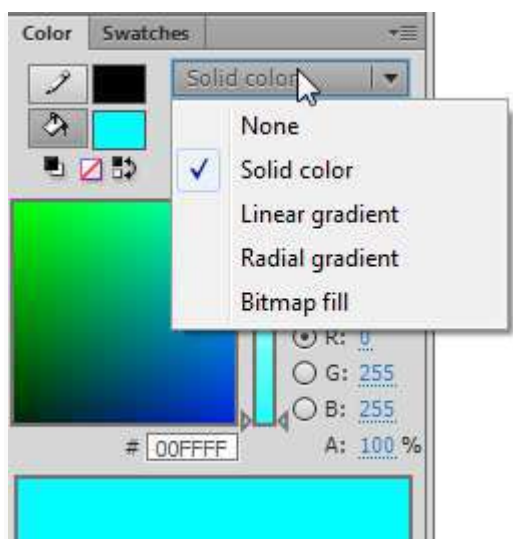
In order to work with the color or the swatches panel, select an object on the stage.



The Flash CS6 Color panel

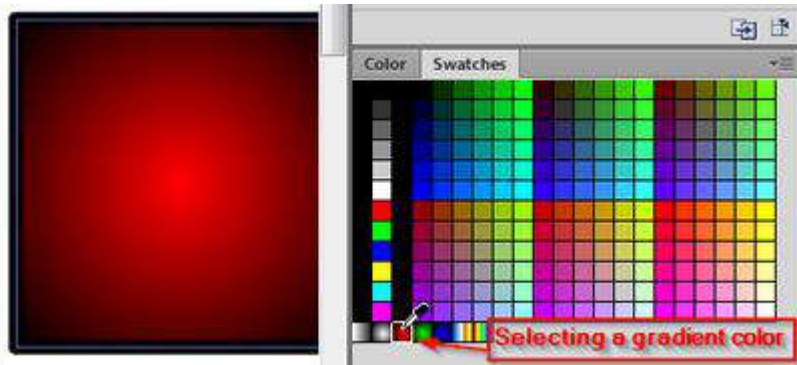
In the color panel you then can:

- Select either stroke or fill for adjustments
- Select various colors (depending on color type)
- Change the alpha channel (i.e. transparency)
- Select the color type (Solid, linear or radial gradient, bitmap fill). See below for more explanations



The Flash CS6 Color panel - select color type

The swatches panel includes a series of standard colors. These are same ones you get with the Fill controls in the Tools and Parameters panel.



The Flash CS6 Color swatches panel

Let's now introduce various color types in more detail.

## Solid colors

---

Solid colors can be defined in various ways (and there is a whole science behind it). Let's just recall a few principles. For more information, please see the Wikipedia links in the colorarticle.

Let's define a few terms first:

### Hue

- means "color"

### Saturation

- means amount of a color you apply, i.e. the intensity.

### Brightness

- How much light you apply. A lot of light makes the color washed out and very little light makes it very dark.

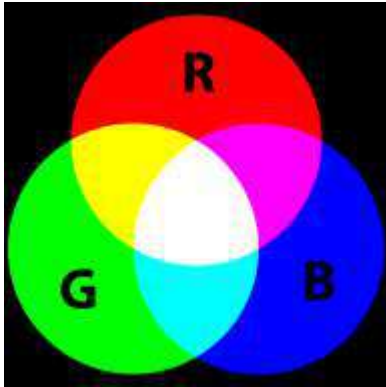
### Transparency

- How much you can see through
- See alpha channel below

### RGB colors

RGB colors are the most popular ones used in computing applications. A color is defined by the **amount** of **R**ed - **G**reen - **B**lue. By default, the CS6 color panel is in RGB mode.

RGB is the way computer monitors work. E.g. to get a nice yellow you need 100% Red + 100% Green + 0% Blue. RGB is a so-called **additive** color mixing model. "Projection of primary color lights on a screen shows secondary colors where two overlaps; the combination of all three of red, green, and blue in appropriate intensities makes white". Now if you project each of these primary colors with different intensity, overlapping colors will change.



This model is not how colors work when you mix real paint. Then you'd rather work with a red-yellow-blue model. Color printers yet work with another model, i.e. magenta, cyan and yellow (or more).

RGB colors can be encoded in various ways. For Internet formats such as HTML, CSS or Flash, most often a *hex triplet* is used, i.e. a hexadecimal 6-digit number. With 2 hexadecimal digits you can represent numbers in the range of 0 to 255.

With ordinary numbers you would represent a full red like this:

(255,0,0) - meaning full red, no green, no blue

The corresponding hex triplet is `FF 00 00`:

`#FF0000`

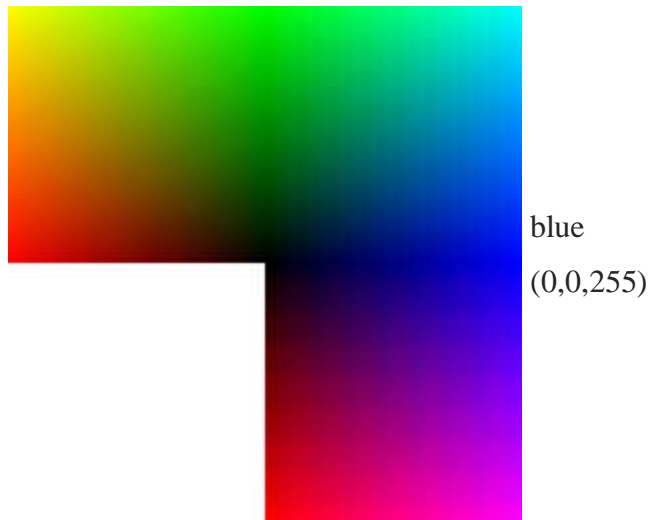
In terms of percentage of colors, you get:

(100%, 0%, 0%)

It represents "TrueColor", i.e. RGB values in 24 bits per pixel (bpp). In TrueColor, colors can be defined using three integers between 0 and 255, each representing red, green and blue intensities. For example, the following image shows the three "fully saturated" faces of the RGB cube, unfolded into a plane:

- (0, 0, 0) yellow      green      cyan

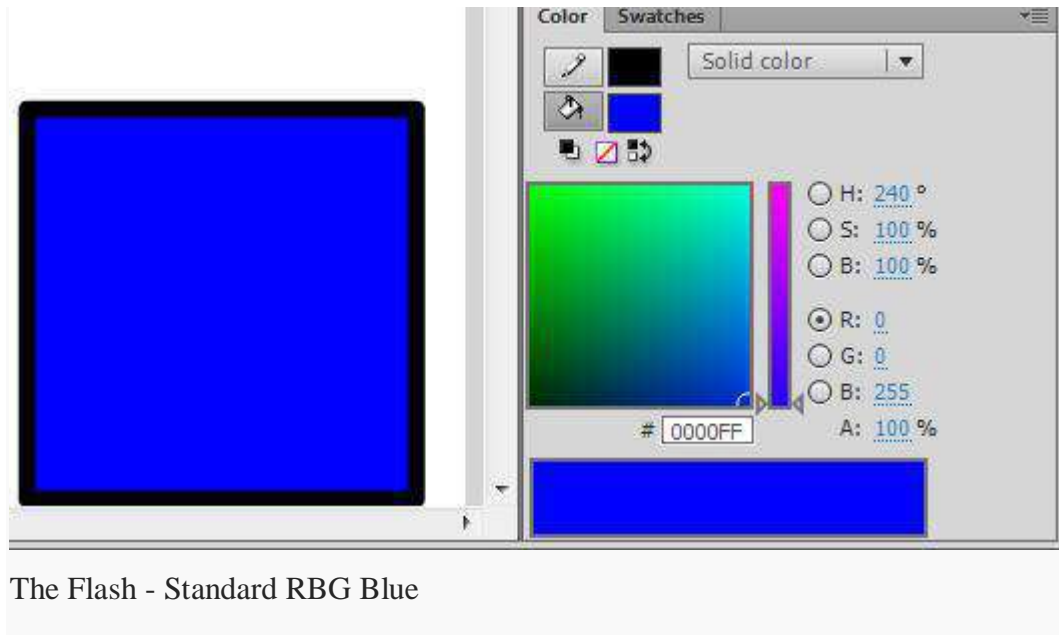
- is black (255,255,0) (0,255,0) (0,255,255)
- (255, 255, 255)  
is white
- (255, 0, 0)  
is red red (255,0,0)
- (0, 255, 0)  
is green
- (0, 0, 255)  
is blue blue (0,0,255)
- (255, 255, 0) is yellow
- (0, 255, 255) is cyan red (255,0,0) magenta (255,0,255)
- (255, 0, 255)  
is magenta



### Using the Flash color panel with solid RGB colors

- It's probably a good idea to starting picking out a standard color from the swatches panel that will display so-called "web-safe" colors. You also could create your own palette.
- Once you start from a "standard" color, you then can either adjust RGB values or color/brightness/saturation with the slider (see the HSB model below) or select another color from clicking into the **Color Picker**. The color range you will see depend on selections made to the right.

Below is a standard blue (the brightness/saturation slider remains in the middle)

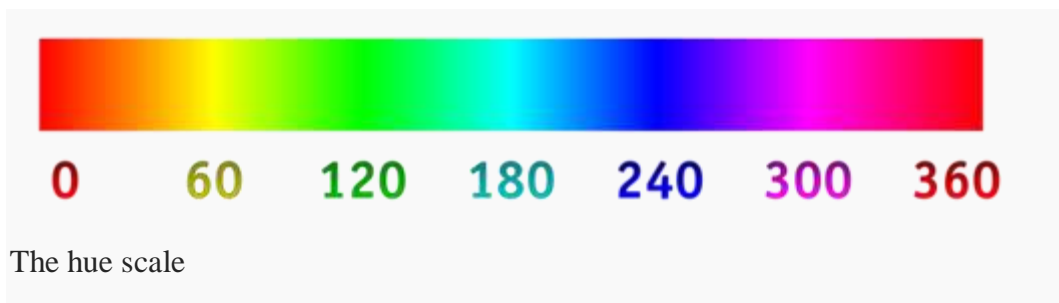


### The HSB/HSV model

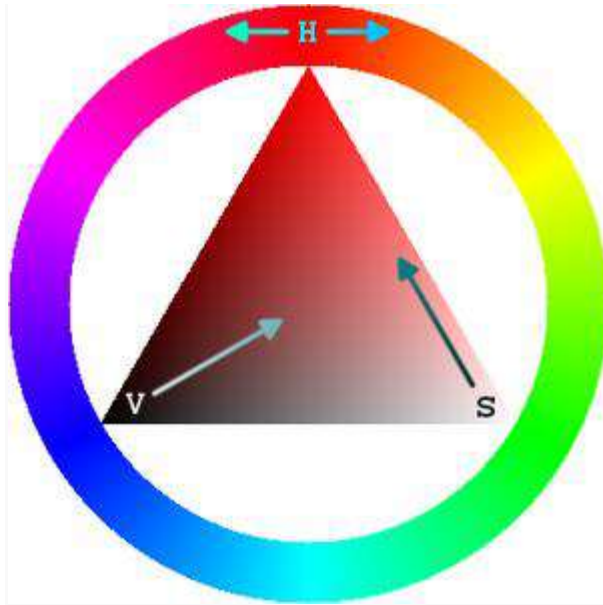
The HSB (Hue, Saturation, Brightness) also known as HSV (Hue, Saturation, Value) defines a color in terms of three components:

1. **Hue**, the color: Represented as a position in the 360 degrees of a color circle.
2. **Saturation**, the intensity or "purity" of the color: Ranges from 0-100%. 0 means no color (white), 100 means intense color.
3. **Value or Brightness** of the color: Ranges from 0-100%. 0 is always black. Depending on the saturation, 100 may be white or a more or less saturated color.

The Hue scale from 0 to 360 degrees is the following:



In many graphics tools (not in Flash) you get a HSV color wheel that looks like this:



The hue scales

On the outside you can select a color (**H**), then on the inside you can select **V** and **S**.

For more information about HSV.

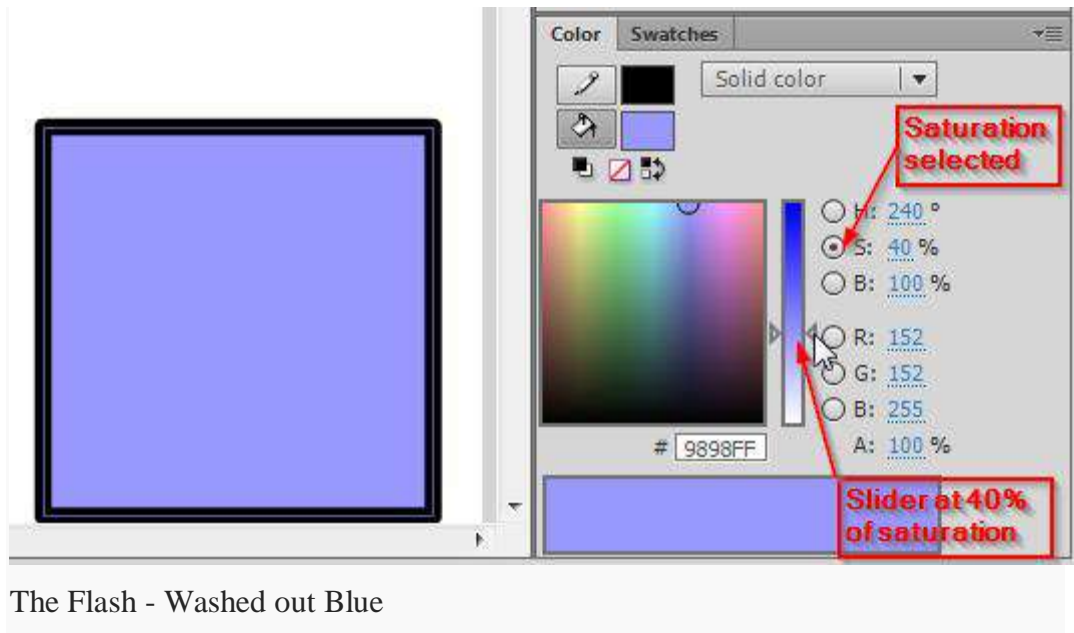
In Flash, with the slider to the right you can either

- change color
- adjust both Saturation
- adjust Brightness.

Clicking into the color picker will change all three.

Below is a washed out blue with less saturation. We selected the saturation button selected and move the slider to 40%.





### Tint and Shade

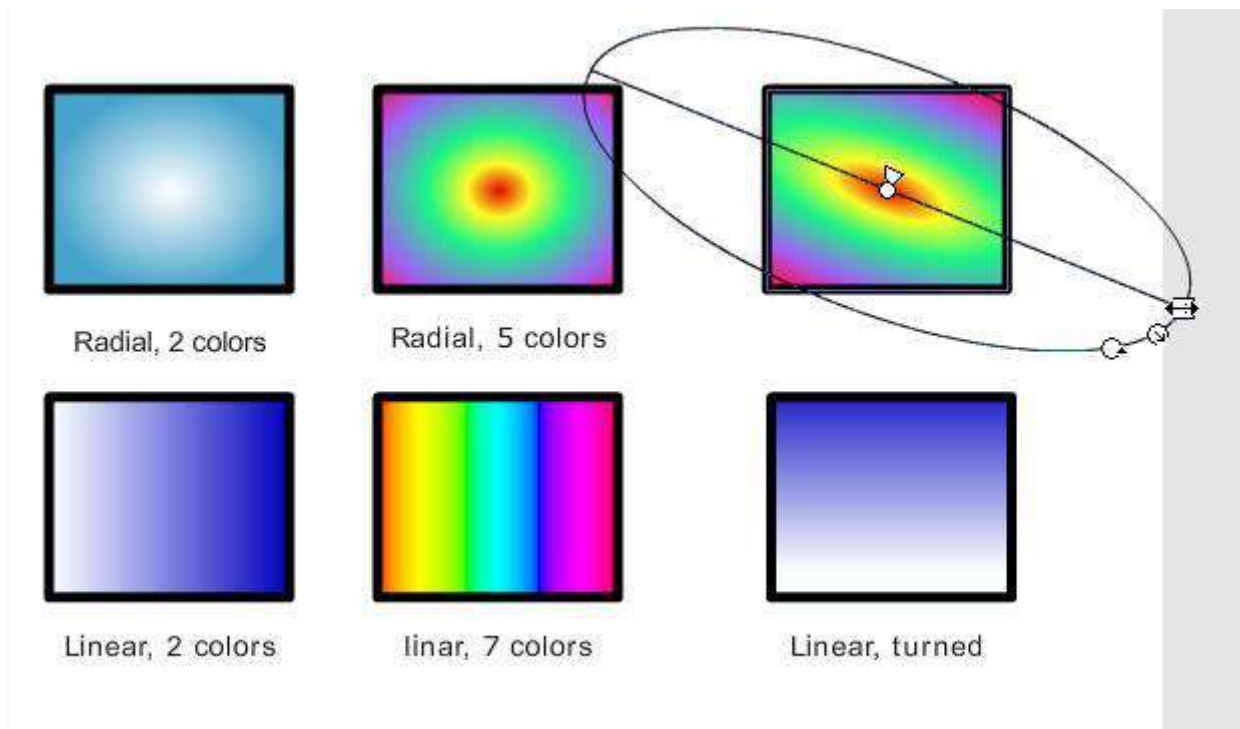
“In color theory, a tint is the mixture of a color with white, and a shade is the mixture of a color with black. Mixing with white increases value or lightness, while mixing with black reduces chroma. Mixing with any neutral color, including black and white, reduces chroma or colorfulness. The intensity does not change.”

In Flash, **tint** is a color that you can add to a symbol in motion twinning. Alternatively (but not at the same time) you can modify its brightness. In addition, you can change its alpha value (make it more or less transparent)

### Flash Color Gradients

Flash supports there are 2 kinds of color gradients (see the picture below)

- Linear: the color changing in one direction
- Radial: the color changing from the center to the outside



### Linear and radial gradients and Gradient Transform

Color gradients work with **color bands**. You can define 2 or more colors and Flash will fill in intermediate colors between them. The result then depends:

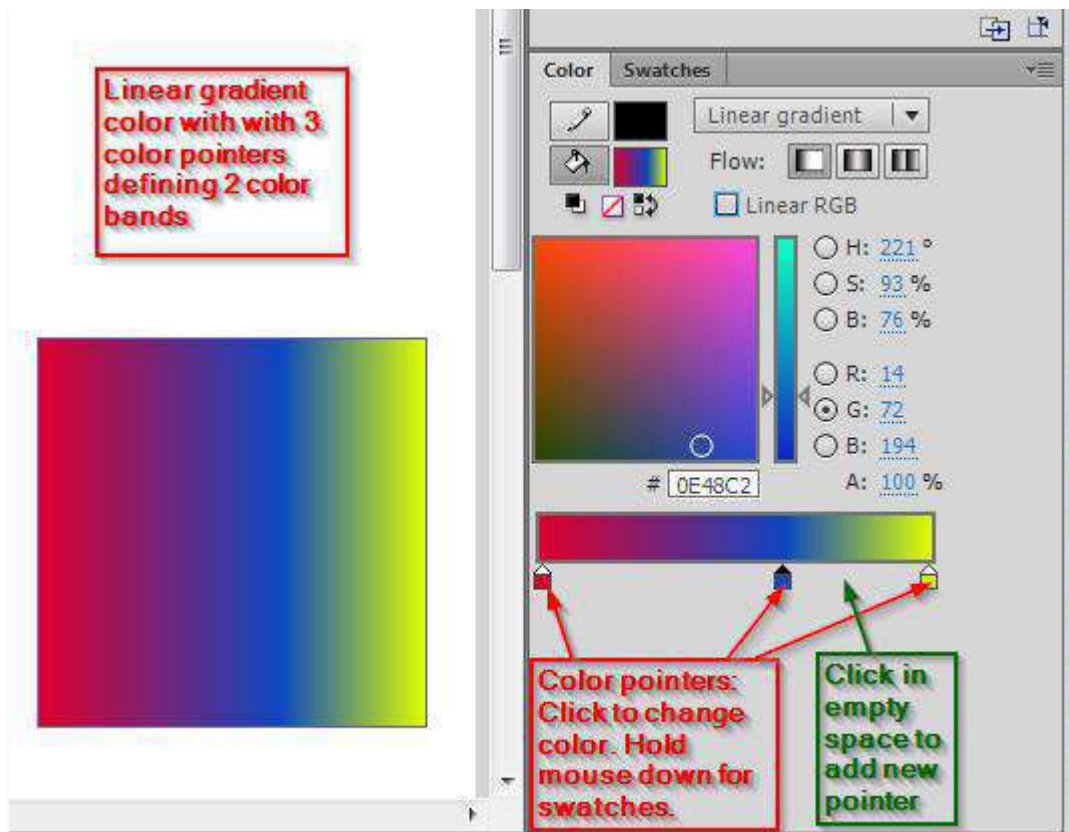
- on the choice of colors
- on the width of the color band (from one color to the next one)

You can change these by defining and dragging color pointers in the Color panel.

### Defining color bands

There are some built-in gradients (linear and radial) that you may use as is, however you most likely want to define your own. In order to achieve that, you need the color panel and then manipulate the controls in the preview window.

If you select either "linear" or "radial" **Type** you will see the **gradient preview window** at the bottom of the color panel:



Color panel - Gradients - Color points

The little "arrow squares" you now can move from left-to-right are called **color pointers** and they delimit **color bands**.

Here is a list of common operations:

**(a) Adjust color bands**

- To make a color band smaller or larger, move various *color pointers* left or right

**(b) Add new color bands**

- Click into the area of the color pointers. This will add a new color pointer.

**(c) Change the color of a color pointer**

- Click on a color pointer. The selector pointer should have a black arrow (instead of a white). Now select a color in the panel above.

**(d) Remove a color pointer**

- Drag it down and off (below the gradient preview window)

### (e) Copy a color from a pointer

- Hold down the mouse for a while will work like the Eyedropper tool

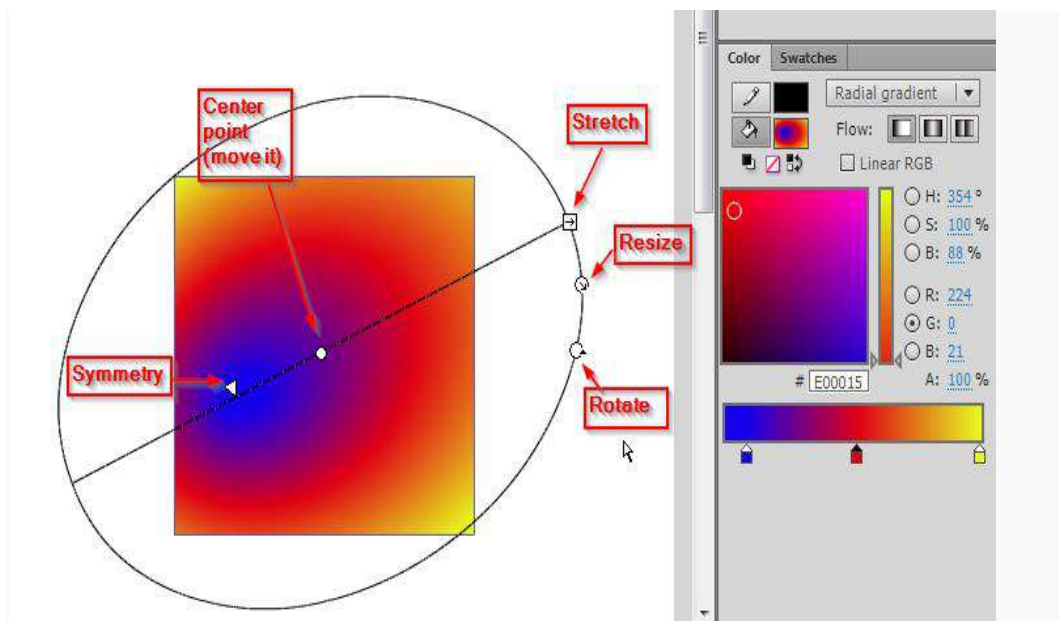
## 4.2 Transforming gradients

With the gradient transform tool (hidden underneath the Free Transform tool) you can do five things:

1. rotate gradients (both linear and radial).
2. stretch out the gradient
3. stretch the radial gradient in only one direction (make an oval)
4. Make the "rings" of a radial gradient asymmetric
5. Move the center of the gradient color

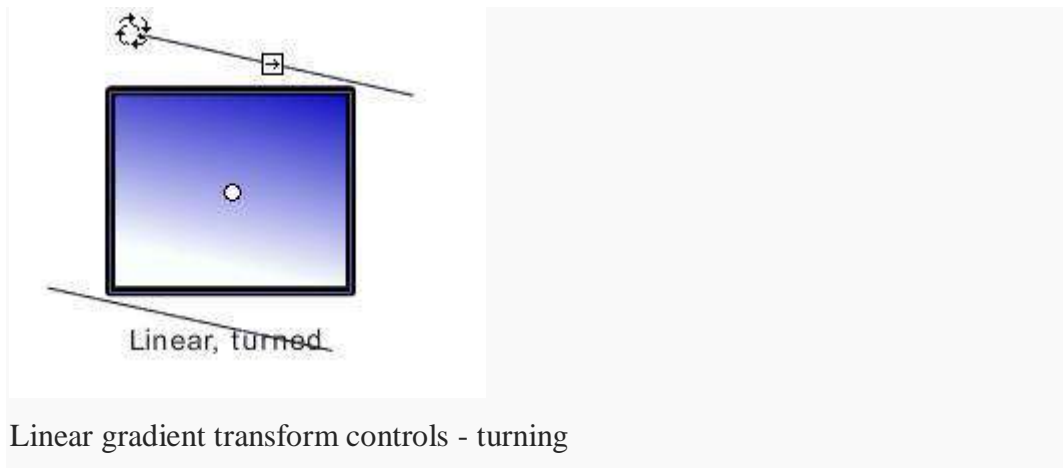
### Procedure

- Select the tool (hold down the mouse over the Free Transform tool in the tools panel) and select the **Gradient Transform Tool** (🔧)
- After selecting an object, you will see five handles with which you can: stretch in one direction, resize, turn, make the rings ellipsoid, and finally move the whole gradient. See the screen capture below, which shows the handles for a radial gradient transform:



Radial Gradient Transform controls

Stretching or rotating a linear gradient works in a similar way:



## The alpha channel

In computer graphics, alpha compositing is the process of combining an image with a background to create the appearance of partial transparency

In more simple terms, you can set the alpha to some percentage:

- 100% can't see through
- 80% bad see trough
- 50% in between
- 30% good see through
- 10% good see through, but very little color
- 0% no color left



Hint: With the alpha channel you can create other effects than see-through "windows". E.g. you can overlay textures with color or the other way around.

## Drawing with bitmaps

---

You can use any bitmap (e.g. a picture of yourself) as color. To do so, you firstly must import the bitmap file (e.g. a \*.jpg file) and then adjust the size with the gradient transform tool

### Importing a bitmap

There are two solutions:

- You can just paste a bitmap graphic into the library from the clipboard. For example, if you see a nice (and copyright free) texture on the Internet with the Firefox navigator, do the following: (1) View image, (2) Copy Image, (3) CTRL-V into Flash
- Save the image on your computer then click on the *Import* button in the colors panel.

### Using a bitmap


- You can use a bitmap graphics either as stroke or as fill color.

### Adjusting "grain size"

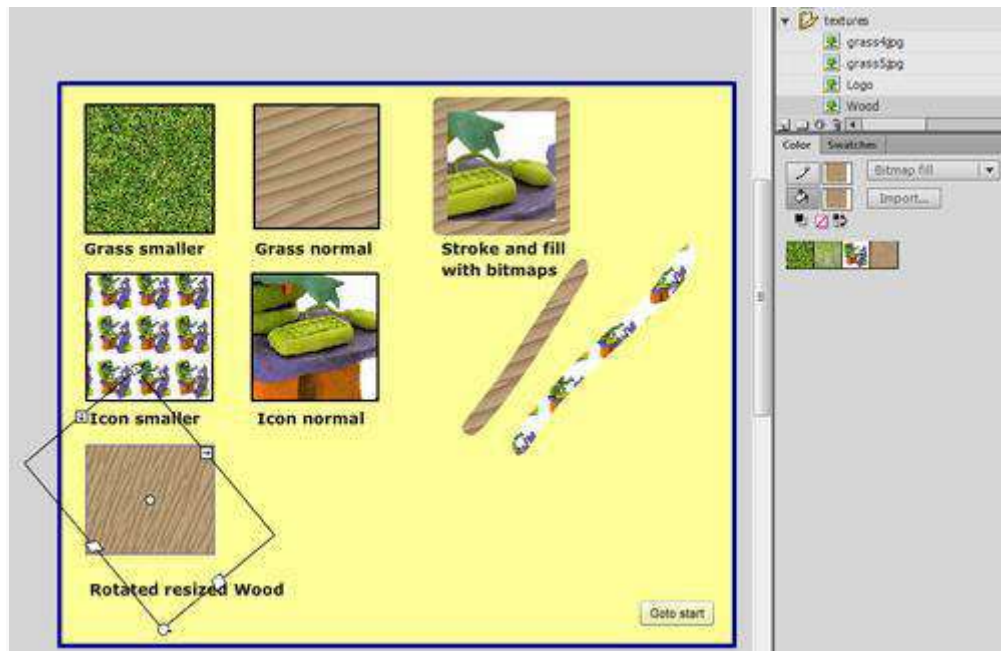
With the **gradient transform tool**, you can adjust how a bitmap will be applied.

You can change:

- Size, i.e. whether the bitmap is applied as is, or reduced or magnified in x, y direction or both
- Rotation
- Skew (a kind of distortion)

Select the Gradient Transform tool () underneath the Free Transform tool, and then

- Click on the fill or stroke
- Play with the handles (if the bitmap is big, you may have to search for these handle way out of the stage 1)



Gradient Transform tool on bitmaps

## Filters for symbol instances

You can apply various color changes to all symbol instances (movie clips, buttons and graphics). To do so, play with the *Color* and *Blend* controls in the properties panel.

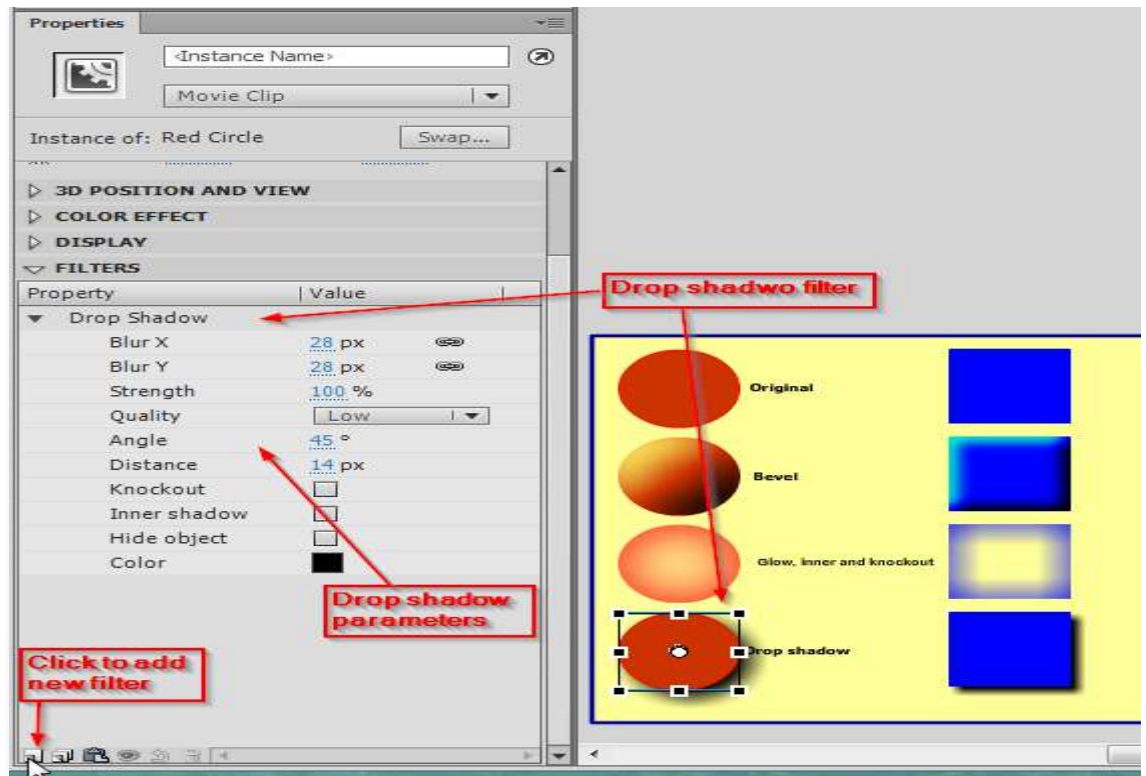
You can add filters (e.g. a gradient glow a bevel or a drop shadow) to movie clip and button symbol instances. Use the Filters panel to do so, (click on the tab in the properties panel), else use menu **Windows->Properties->Filters**)

To add filters, simply click on the + sign and then play with the parameters. Using different sorts of "Quality" also has an important effect on the rendering, high quality may slow down certain computers.

- Blur X and Blur Y define the size of the affected area
- Strength the force of the filter (more or less)
- Shadow and Highlight, the dark/light colors of the filter effect
- Angle and Distance, direction of the filter effect
- Knockout and Type, whether it applies to the inside and whether the original drawing is knocked away.

This is a nice feature that beginners often overlook. So, if you need cool looking 3D effects on graphics explore these filters. You also can apply several filters to

the same object. In the screen capture below, we show an attempt to create a floating 3D button from a simple red circle.



Flash CS6 filters you can apply to movie clip and button instances

**Understand the basics of frame-by-frame animation.** This is considered the "traditional" method of animation, in that each frame has the same picture but slightly altered. When the frames are played together, the images appear to move. This is the same basic technique used by traditional hand-animators and is more time-consuming than tweening.

- By default, Flash will produce animations at 24 frames per second (FPS). That means one second of animation will have 24 frames, but not every frame has to be different. You can adjust this if you'd like, and many Flash animations use 12 FPS, but 24 will produce much "smoother"-looking animation.

**Install Flash Professional.** There are a variety of Flash animation programs available, but the most powerful is Adobe's Flash Professional CC. You can install the trial for free, or you can use another product if subscribing to Adobe's Creative Cloud doesn't appeal to you. The rest of this article will refer to Flash Professional or any other editing program as "Flash".

**Create your assets.** Because frame-by-frame animation requires multiple images with slight differences, you will need to create all of these assets by hand. You can create them



all before you begin or make them as you go. You can use the Flash program to draw directly in your project, or you can draw your assets in your favorite image creation program.

- If you want your images to scale without sacrificing quality, you will want to create them as vectors instead of raster's. Vector images will redraw themselves whenever they are scaled, which means there won't be any pixilation or aliasing. Raster images are the traditional images you're probably used to (photos, Paint images, etc.). These do not scale well and can look quite ugly if you want to make them larger.

**Create your first frame.** When you first start Flash, you will have a blank layer and an empty Timeline. As you add frames, your timeline will be populated automatically. You can interact with layers much in the same way that you would in Photoshop.

- Before adding your image, create a basic background for your movie. Rename Layer 1 to "Background" and then lock the layer. Create a second layer and name it whatever you'd like. This will be the layer that you create your animation on.
- Add your drawing to the canvas in the first frame. You can either import your drawing from your computer, or you can use Flash's drawing tools to create a drawing directly in the program.
- Your first frame will be a *keyframe*. Keyframes are frames that have an image and form the backbone of your animation. You will be creating a new keyframe each time the picture changes.
- Keyframes are denoted by a black dot in the timeline.
- You don't need a new image in each frame. In general, having a keyframe every four-five frame will produce a good animation.

**Turn your drawing into a symbol.** By turning your drawing into a symbol, you can easily add it multiple times in a frame. This is especially useful if you need to quickly create multiple objects, such as a school of fish.

- Select your entire drawing. Right-click on the selection and select "Convert to Symbol". This will add the image to your Library where you can quickly access it in the future.
- Delete the drawing. Don't worry! You will be adding it back to the scene by simply dragging the object from your Library. You can add multiple instances of the same drawing to your scene with ease!

**Add some blank frames.** Once you have your first frame ready, you can add your blank frames that will exist between the first keyframe and the second keyframe. Press **F5** four or five times to add blank frames after your first keyframe.

**Create your second keyframe.** Once you've added a few blank frames, you're ready to create your second keyframe. There are essentially two different ways you can do this: you can copy your existing keyframe and make small adjustments, or you can create a blank keyframe and insert a new image. If you are using art created in another program, you'll want the second method. If you created your art using Flash's design tools, use the first method.

- To create a keyframe using the contents of the previous keyframe, press **F6**. To create a blank keyframe, right-click on the last frame in your timeline and select "Insert Blank Keyframe". Anything in your scene will be removed.
- Once you've created your second keyframe, you will need to make adjustments to the image to give the effect of motion. If you're using Flash's design tools, you can use the Transform tool to select aspects of your drawing and move them slightly, such as the arm of a stick-person.
- If you're inserting new art for each keyframe, you will want to ensure that it is placed in the same location or in the next logical location on the screen. This will ensure that the art doesn't jump around between frames.

**Repeat the process.** Now that you've created two keyframes, it's time to iterate. You will be repeating essentially the same process until you are finished with your animation. Add a few blank frames between each keyframe and ensure that your movements look fluid.<sup>[1]</sup>

- Make small, incremental changes. Your animation will look much smoother if you make very small changes to the keyframe. For example, if you want a stick person waving an arm, your second keyframe should not be the other end of the wave. Instead, use a few keyframes to transition from the beginning of the wave to the end. This will produce a much smoother animation.

### **Making a Point-to-Point Animation (Tweening)**

**Understand the basics of *tweening*.** Flash contains a function called tweening, which allows you to essentially set start- and end-points for your object. Flash will then move and transform the object based on your settings between these two points,

creating the illusion of animation. You will not need to create images for every keyframe like you would with frame-by-frame animation.

- Tweening is especially useful for creating "morphing" effects, where one object becomes another one over the course of the animation.
- Frame-by-frame animation and tweening can be used in conjunction in the same movie.
- You can only have one object undergoing a motion tween at a time. This means that if you want multiple objects to animate at once, they will all need to be on separate layers.

**Create your first object.** Unlike frame-by-frame animation, you will not need to create multiple objects to animate using the tween function. Instead, you will be creating one object, and then changing its properties during the tweening process.

- Before adding your image, create a basic background for your movie. Rename Layer 1 to "Background" and then lock the layer. Create a second layer and name it whatever you'd like. This will be the layer that you create your animation on.
- It is highly-recommended that you use Flash's built-in design tools or import your image from a vector-drawing program. Vectors can scale easily without distortion, while traditional raster images will not scale and transform well.

**Convert your object to a symbol.** In order to tween your object, you will need to convert it into a symbol. This is the format for objects that Flash uses in order to manipulate them. If you try to tween an object that has not been made into a symbol, you will be prompted to do so first.

- Right-click on your object and select "Convert to Symbol" The object will be added to your Library, which makes it easy to clone objects.

**Create your first motion tween.** A motion tween moves the object from one location to another Right-click the symbol in your scene and select "Create Motion Tween". 24 frames will be added to your timeline, as this is the default length of a tween. Remember, by default Flash will animate at 24 frames per second, meaning that this motion tween will take one second to complete.

- When you create the motion tween, you are automatically moved to the last frame of the tween.

**Create the trajectory.** Once you have created the tween, you can move the object to the location that you want it to end up at. Flash will display the trajectory line, which is dotted to show the location of the object for each frame of the tween.

- When you first create the trajectory, it will be a straight line from the starting point to the ending point.

**Extend your background frames.** If you ran your animation right now, your object would move along the trajectory, but your background would disappear after one frame. To fix this, you will need to extend your background across all the frames of the animation.

- Select your background layer in the timeline. Click the last frame on your timeline, which should also be the frame your motion tween ends on. Press **F5** to insert frames up to this point, which will keep your background displayed during the entirety of the tween.

**Add keyframes.** Adding keyframes to your trajectory will allow you to transform your object during the motion tween. You can only make changes to an object if it is keyframed. To add keyframes to your trajectory, first select the frame in the timeline that you want to turn into a keyframe. Then, click and drag the object into the position you want it to be at during that frame. Your trajectory will adjust automatically, and a keyframe will be added to the timeline. Keyframes are denoted by diamond icons in the timeline.

**Adjust your tween trajectory.** To change the path of your object, you can click and drag each of the frame position markers on the path to a new location. Avoid putting too much variation in the path, or the object will move in an erratic fashion (unless this is your goal!).

**Transform your object.** Once you have your keyframes and trajectory established, you can transform your object so that it changes as it moves through the motion tween trajectory. You can change the shape, color, rotation, size, and any other property of the object.

- Select the frame that you want the transformation to occur in.
- Open the object's Properties panel. You can press **Ctrl+F3** if the panel is not currently visible.
- Change any values in the Properties window to affect the object. For example, you can change the hue or color, add filters, or change the scale.
- You can also use the Free Transform tool to freely change the shape however you'd like.

**Make finishing touches to your tween.** Test your motion tween by pressing **Ctrl+⌘ Enter**. Ensure that the transformations look good and that the animation is moving at the correct speed. If your animation is moving too fast, you can either lower the FPS of the scene or increase the tween span.

- By default, the FPS is 24, so try lowering it to 12. To do so, click outside the scene and then change the FPS in the Properties panel. Changing it to 12 will double the length of your animation but may make it more "jerky".
- To change your tween span, select the layer that contains the tween and use the slider to pull it out. If you want the tween to be twice as long, extend it to 48 frames. Make sure to insert blank frames in your background layer so that the background doesn't disappear halfway through the animation. To extend the background, select the layer, click the last frame of the animation in the timeline, and then press **F5**

## Adding Sound and Music

**Record or find your sound effects and music.** You can add sound effects to actions that occur in your animation to help make it stand out and give it some personality. Music makes your animation more immersive and can make a good animation into an incredible one. Flash supports a variety of file formats, including AAC, MP3, WAV, and AU. Choose the one that gives you the best quality for the lowest file size.

- MP3 will typically give you perfectly acceptable sound quality while keeping the file size to a minimum. Avoid WAV files if possible, as these are often quite large.

**Import the sound files to your Library.** Before you can add sounds to your project, they need to be added to Flash's Library. This will allow you to quickly add them to your project later. Click File → Import → Import to Library. Browse for the sound file on your computer. Make sure that the sound file has an easy-to-remember name, so you can quickly find it from the menu later.

**Create a new layer for each sound file.** This is not strictly necessary, as you can add sound to existing layers, but putting each file on its own layer will give you much more control over fading in and out, when to start and stop, and it is easier to move sounds around.

**Create a keyframe where the sound will start.** On the sound's layer, select the frame in the animation that you want the sound to start at. Press **F7** to insert a blank keyframe. For example, if you want to include a music file that plays for the duration of the animation, select the first frame on the music file's layer. If you are adding dialog for a character, select the frame where the character begins talking.

**Add the sound of music file.** In the Properties frame, you will see a Sound section. You may need to expand it to see the options. Click the "Name" menu and select the file you want from your Library.

**Configure the sound file.** Once you've selected a file, you can configure how it will play. What you select will be based on what you need the sound to accomplish in the animation. You can change the Effect, the Sync, and the Repeat settings of each sound using the menu underneath the Name menu in the Properties frame.

- Effect - This let's add effects to the sound, such as fading in and out or adding echo. You can select from preset settings in the drop-down menu, or you can click the pencil icon next to the menu to create your own custom settings.
- Sync - This determines how the song is played in the animation. *Event* will play the sound until it is finished. If the same sound is triggered again before the first one ends, the the original will keep playing until it finishes. *Start* works like Event but stops if the sound plays again. *Stop* stops the sound in that frame. If you want to use this in conjunction with other sound properties, create a new keyframe where you want the sound to stop and use this option. *Stream* will attempt to match the sound that is playing with the number of frames on other layers. This is best used for dialogue.
- Repeat - This setting allows you to set how long the sound repeats. You can set it to play only once, or have it loop as many times as you want. If your animation is a looping animation, you should set your music to loop infinitely.

**Finish your project.** When you are finally finished with your project, save it as an SWF file. This is the format used to play the movie. You can play it in virtually any web browser or use a dedicated Flash player to watch it. There are also a variety of sites that you can upload it to for others to see, such as Newgrounds, Albino Blacksheep and Kongregate.

**Take your future projects further.** This guide addresses the basics of creating an animation, but there is so much more you can learn and do. Add a few buttons and branching paths, and you've got a choose-your-own-adventure game. You can take a crash course in ActionScript and gain much more control over the finer details of your animation. Keep experimenting, and you'll soon find yourself learning all kinds of tricks and implementations.

## Layers Introduction

---

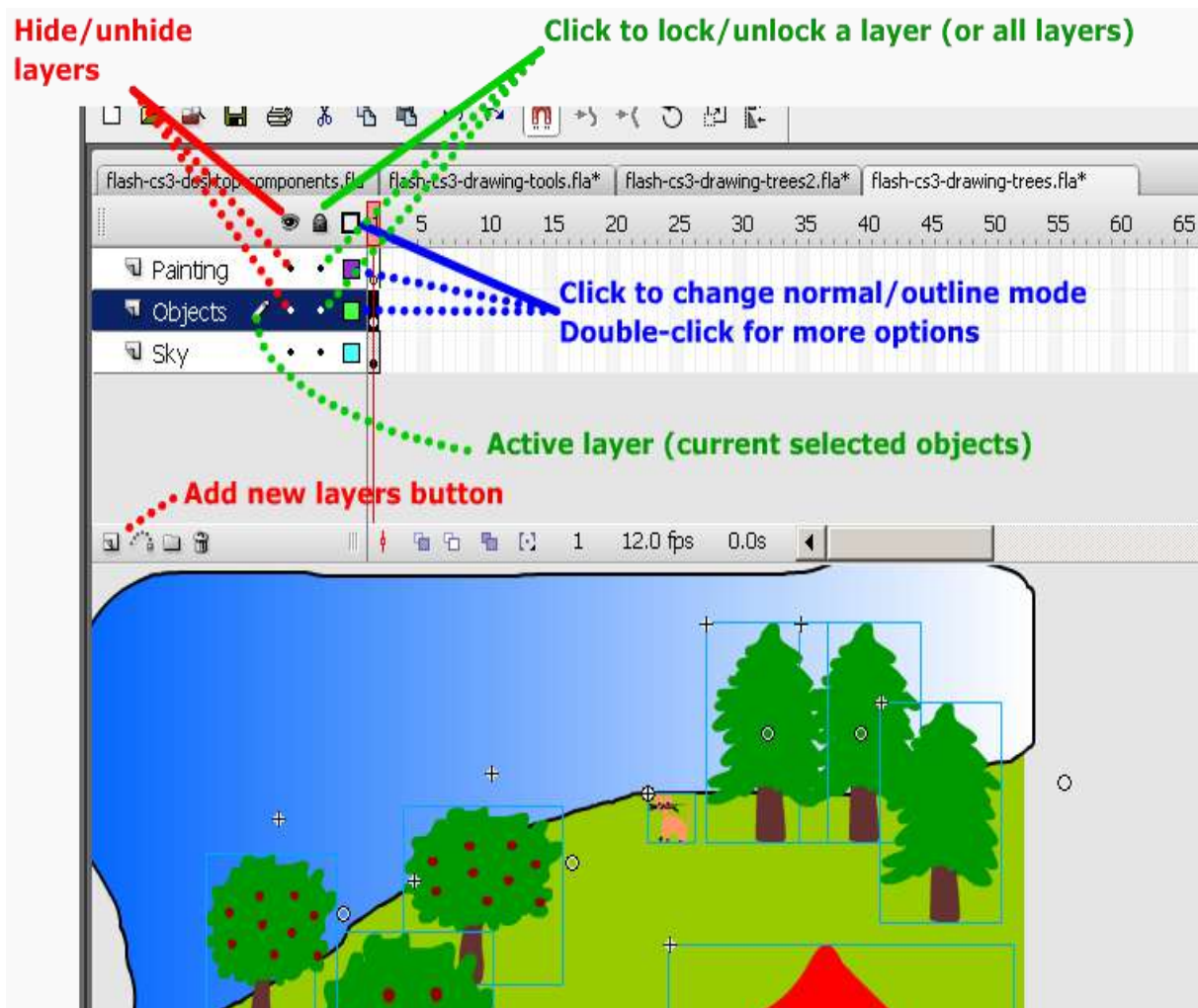
Layers help you deal with more complex Flash projects. Working with layers has several advantages:

- You can draw and edit objects in one layer without affecting objects in another layer.
- You can lock layers (to protect their embedded objects from unwanted editing)
- You can hide layers, make them visible (i.e. you can see their objects in the workspace), or you can display just the outlines of their objects.

The layers tool is part of the Timeline panel.

### Overview picture

The layers tool is in the left part of the timeline. Annotations in the following screen capture highlight a few functionalities we will further explain below.



The Flash CS3 Layers tool

### Drawing in a layer

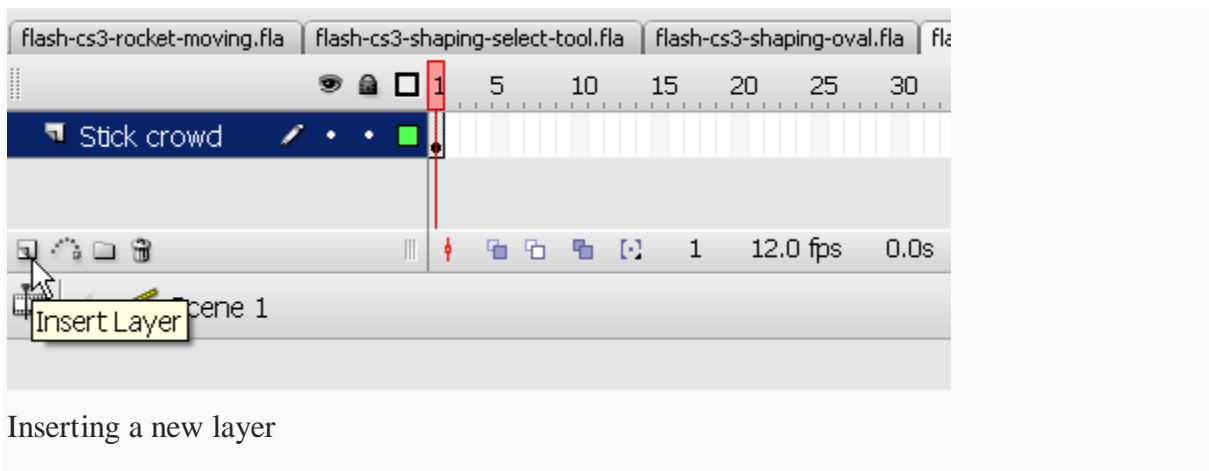
To draw, paint, or otherwise modify a layer simple click on the layer name in the Timeline to make it active. A pencil icon next to it indicates that the layer is active.

### Creating new layers and deleting layers

When you create a Flash document, it only contains a single layer, i.e. less than you need.

To create a new layer, either:

- *Insert->Timeline->Layer*
- Click on layer icon (left most item in the Edit bar just below the timeline)
- *Right-click* on an existing layer, then *Insert Layer*



Inserting a new layer

As soon as you create a new layer, you should give it meaningful name. Right-click on its name (something like Layer 2) select *Properties* and change the name. Alternatively, to display this properties panel, just **double-click** on the layer name.

To delete a layer and its contents: *Right-click->Delete Layer*. You also can lock/hide other layers with this menu. Before you delete a layer make sure that you save its objects if you plan to keep them. You can insert them in the library as symbols or copy them to another layer.

### Show only outlines of a layer

- Click on the rectangle next to the layer name. If this rectangle turns empty then you only should see outlines of its objects.
- You also can change the outline color by double-clicking on the rectangle. E.g. if your background is green (like the grassy hills in our example), the outline should be of a different color.



## Lock and hide layers

Click on the dots below the appropriate hide/lock/display icons in the panel to apply locking/hiding/displaying to a single layer, or on the icons themselves to apply an operation to all layers (e.g. lock them all).

TIP: Always **lock all layers** and then just unlock the layer on which you are working. This way you can prevent yourself from making mistakes.

## Moving layers

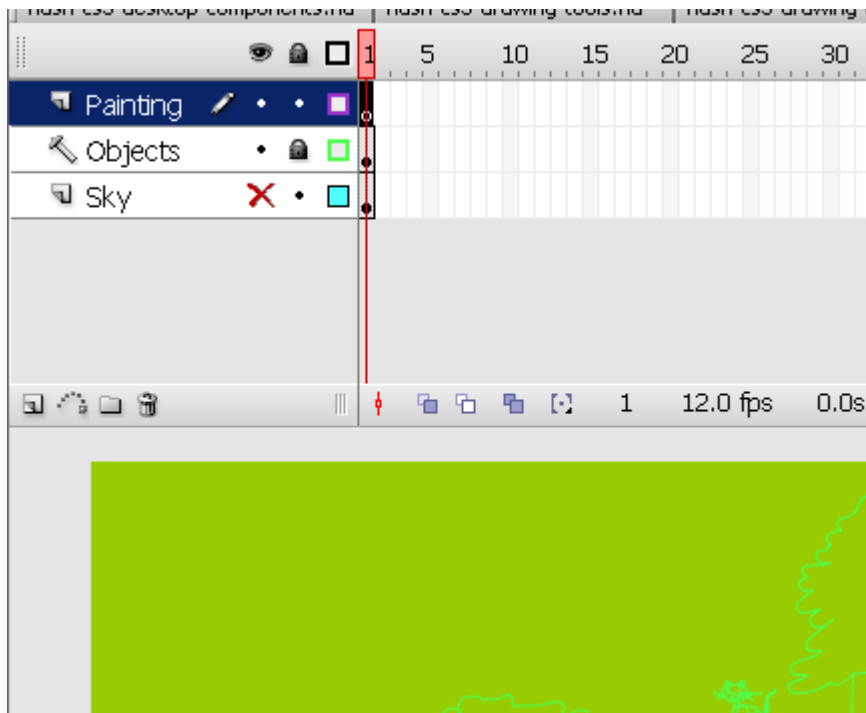
To move a layer in the stack simply grab it with the mouse and drag it up or down. Position of the layer has an influence on the order objects are drawn. E.g. if a moving object should pass in front of a tree and it doesn't, then drag the animation layer up or down.

Drawing order depends on the load order defined in the *Publish Settings* (File menu)

## Example

The following screen capture shows hidden and locked layers:

- The painting layer is active (the pencil is shown)
- The objects layer only shows outlines and in addition it is locked (the lock sign is on and the rectangle is empty. Its objects are drawn in light green, i.e. the color of the rectangle)
- The Sky layer is hidden (The red "X" sign is on).



The Flash CS3 Layers tool, shown with outlines of layer and a hidden layer

## Layer folders

Once your documents get really complex, you can organize layers into folders, e.g. one folder per task: Static objects, animations, background etc.

To create layer folders, either:

- click on the folder icon in the Edit bar (third item)
- or use *Insert->Timeline->Layer Folder*

You then can drag around layers. Hiding, locking etc. works more or less like with folders (try it out ...)

However, at some point you also will have to decide whether you really want to work with an "everything is in the main time line model". Consider organizing and planning your project with embedded movie clip objects. Putting everything in the main time line is like programming with "goto's".

## Scenes

Once your animation gets bigger, you most certainly should break it down to several scenes. There is no urgency to work with scenes if you are new to Flash, but you should know about this now. Scenes are played in the order you defined them.

### **To insert a new scene**

Menu *Insert->Scene*

### **To rename/reorder the scenes**

- Menu *Window->Other Panels->Scene* (SHIFT-F2)
- Then drag up or down the scenes
- To rename, double-click on a scene name in this panel.

### **To navigate between scenes**

- Either via the scenes panel, or the Edit Bar (displayed below the timeline). If you can't see it: *Window->Toolbars->Edit Bar*.

One advantage of using scenes is that you can just test a single scene (menu *Control->Test->Scene*).

### **Sound types**

Flash can handle several sound formats:

- AAC (Advanced Audio Coding):
- AIFF (Audio Interchange File Format) - Mac only
- MP3 (Moving Pictures Expert Group Level-Layer-3 Audio)
- AVI (Audio Video Interleave)
- WAV (Waveform Audio Format)
- AU (Sun)

(Some formats may depend on whether QuickTime is installed on your computer).

Best bet is to use MP3 format, since it is very popular. E.g. it is easy to find music or sound textures on the Internet.

Flash CS3 and CS4 provide some sounds in a library (Menu: *Window -> Common Libraries -> TNT sounds*). CS4 has a much better choice...

### **Sound imports to frames of the timeline**

---

We shall explain the whole procedure using a simple animation example.

### **Background sounds**

Smaller sound files should be imported to the library.

## To import à sound file

- File->Import->Import To library (or drag and drop).

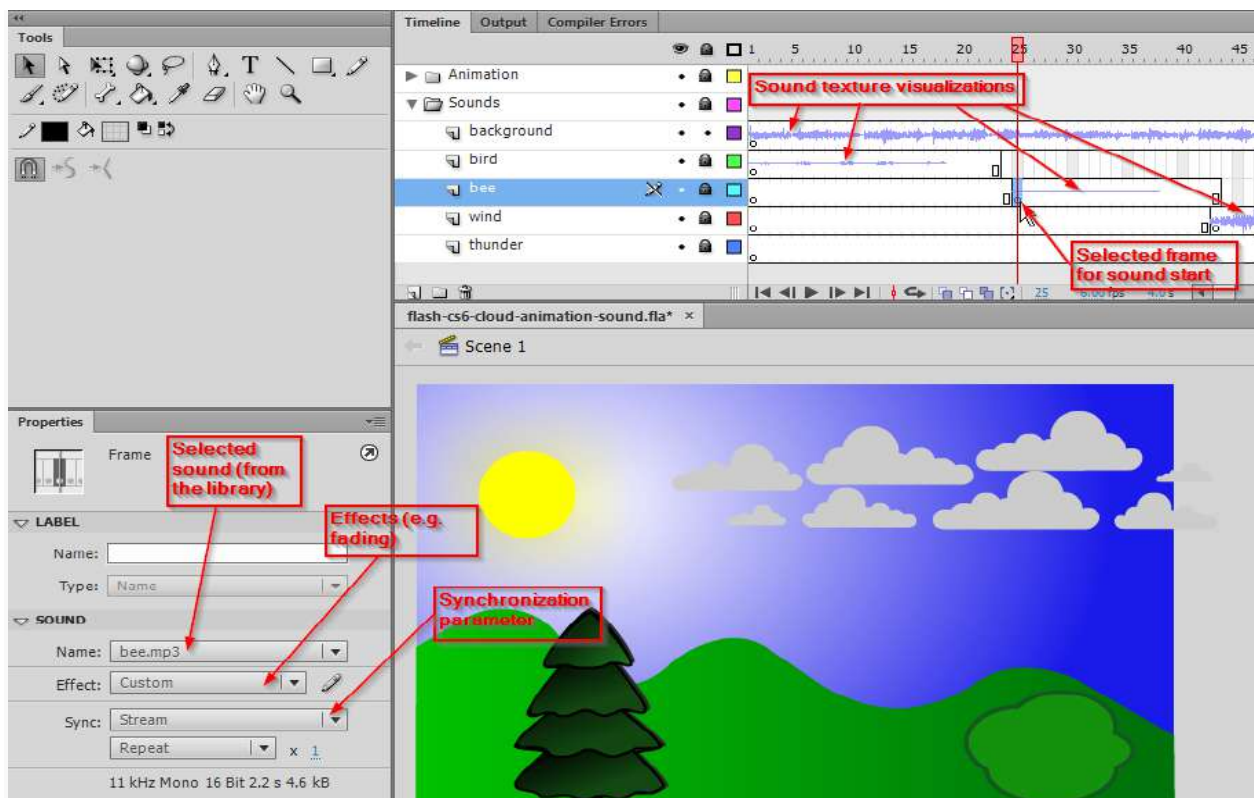
## Attaching sound to a frame

### Step 1 - Create a new layer and import sound to a frame

You can attach sound to any frame via the properties panel

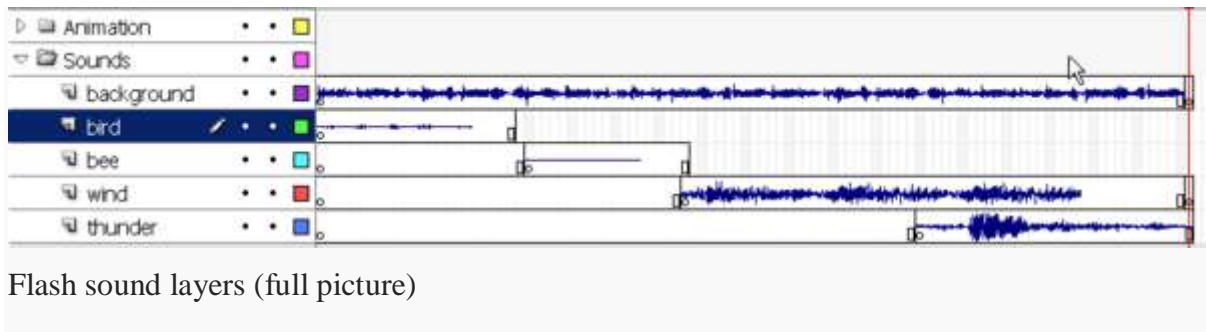
- Create a new layer for this sound (not mandatory, but good practice)
- Insert a **keyframe** (F7) where you want the sound to start
- Select a sound from the sound pull-down menu in the properties panel.
- Configure it in the same panel (see next)

Ideally, each sound should have its own layer. This way it is much easier to control fade in/outs, when to stop etc.



## Flash sound layers

You also can see exactly how far the sound will extend on the timeline. Hit F5 or F7 (if you later want to stop the sound) somewhere to the right.



Flash sound layers (full picture)

## Step 2 - Configuration of sounds

In the configuration panel you can change certain parameters and also edit a bit.

Sync: Will defined how sound is synchronized with the timeline.

- **Event:** Sound plays until it is done (independently of the rest). It has its own "timeline". Also, if this sound is triggered again (e.g. a user enters the same frame), a new sound will play even if the old one is not over.
- **Start:** Similar as event. Will play the sound when the frame loads but will not play it if the old sound is still playing. Note: This doesn't always work as expected. Probably best to use together with the Stop (see below).
- **Stop:** Will stop the sound of a layer at this frame (therefore include it *after* a sound frame). Insert a new keyframe (F7) where you want it to stop and just edit the properties.
- **Stream:** Will try to match the length of sound with the other layers, e.g. 20 frames of sound should play during animation of 20 frames. After that it should stop. Sound as stream should not be looped. Use this for example for comic strips (talking characters).

Repeat:

- You can repeat the sound as many times as you like (or even have it loop forever).

Effect:

- You can choose from various fade in/out and left/right options, but you probably want to do your own custom fades (see next).

## Editing sounds

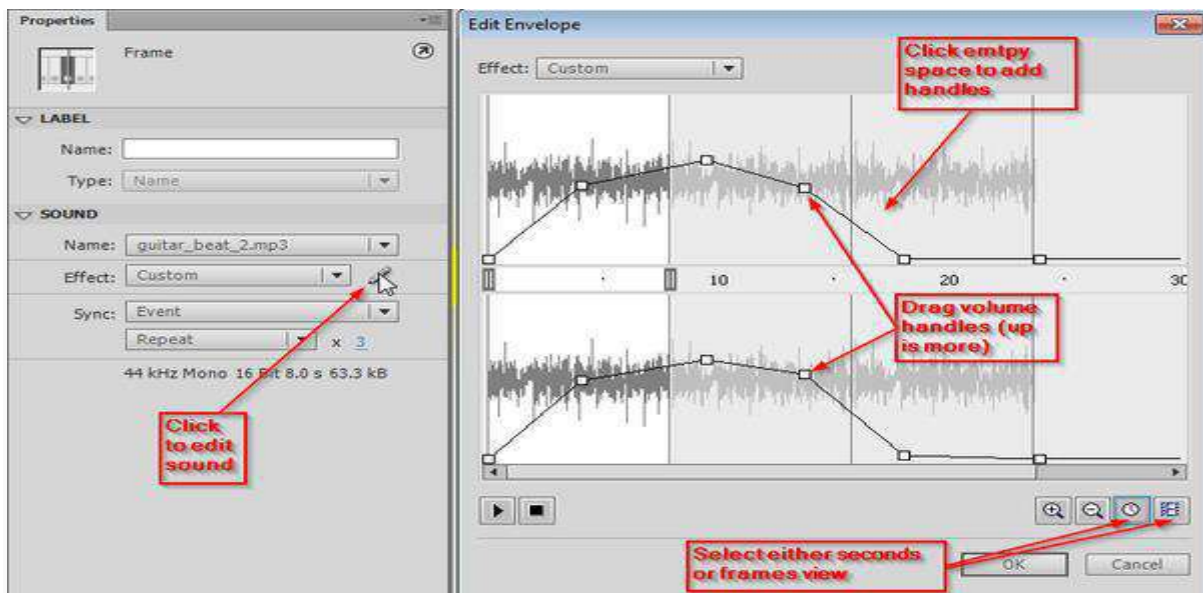
### Editing sound with the Edit Envelope editor

- Click in the sound layer in some frame where you have sound
- In the Properties Panel, Click the *Edit ...* button next to the *Effect:* field

- This opens the **Edit Envelope** sound volume editor.

### Manipulation of the sound envelope

- You can drag left/right *Time In* and *Time Out* controls in middle pane. I.e. you can cut off sound from the either the beginning or the end of the sound track.
- You can drag down volume controls (black lines on top) for the left and the right stereo channel
  - Click to insert a new distortion point for these volume controls
  - Up: means louder / maximum sound
  - Down: means more silent / no sound
- Use the arrow (down left) to test
- At bottom right there are zoom buttons and a switch that either shows seconds or frames.



Flash CS6 Sound envelope editor

### Attaching sound to buttons

You can attach sounds to buttons in the same manner as above.

- Double-click on the button in the library panel
- Edit the button's timeline (e.g. the mouse over, down and hit frames)
- For each sound you want to attach, create a layer
- Then insert a new keyframe (F7) and attach the sound
- You may try to stop a sound (insert a new keyframe)



Flash CS3 Attaching sound to buttons

## Button with sound files

### Load and play sounds with ActionScript

It is better to load sounds with ActionScript if your sound file is large, e.g. a background music or if you want to trigger a sound as a result of some complex user interaction. Select the frame where the sound should start (typically in a "script" layer), then insert this kind of code with F9.

**To load a sound from an external file:**

```
var request:URLRequest=new URLRequest("track.mp3");
var your_sound:Sound=new Sound();
your_sound.load(request);
```

**Alternatively, to load a sound from the library:**

- Export the sound for ActionScript 3:
  - Right click on the sound in the library, select properties
  - Select the ActionScript tab
  - Tick *Export for ActionScript*
  - Define the classname, e.g name class for a file called "noise.mp3" something like "Noise\_sound".

- Then create a new sound from this class (just this single line)

```
var cool_noise_sound:Sound=new Noise_sound();
```

To play your sounds:

```
your_sound.play();  
cool_noise_sound.play ();
```

To play 5 loops:

```
your_sound.play(0,5);
```

To stop all sounds (this is a static method, just insert the line as is).

```
SoundMixer.stopAll();
```

Stopping a single sound takes some more code, *your\_sound.stop()* will not work, since you will have to stop a *Sound channel* (as opposed to just the sound file). Use the following code fragment. See the on/off button example just below for a complete example.

```
var channel:SoundChannel;  
channel =s.play();  
.....  
channel.stop();
```

### **For an on/off button**

Select the symbol which is aimed to be the button, in the frame where the on / off has to happen. Select the code snippet "click to play/stop sound" in the audio and video category.

### **Play sounds randomly with one button**

---

To import video in Flash, you should begin by selecting the *Import Video* option from the File > Import menu in Flash 8. Upon doing this, you will see the *Select Video* screen, which



allows you to browse to a video clip. Once you've picked the clip you'd like to convert, you can proceed to the next step, Deployment. Deployment is simply choosing the manner that you will include the video into your flash movie.

- *Embed Video* is the selection that is chosen when you first come onto the Deployment screen. This will place your movie directly into the Library of your Flash document. It is important to remember that this will significantly increase the size of your SWF file.

*Link Quicktime Video* allow you to import a video clip for a video track in a Quicktime Flash movie. After this, the Quicktime video is placed on the timeline and the import session is finished.

- Upon choosing Embed Video, you are given several choices as to how you would like your video content to be stored in Flash, and also gives you options of how to process the audio portion of your imported video.
- The *Editing the Video* portion allows you to specify clips that you will edit by dragging markers to the video timeline. Upon doing so you can rename the clips, whereupon they show up on the left side in a field that allows you to select different ones and will eventually show up as individual symbols in your Library.
- *Encoding* is the next step of importing video. The profiles are split into two sections, Flash 7 and Flash 8. The Flash 7 option features different presets that are encoded with the Sorensen Spark Codec, the Flash 8 presets are encoded with the VP6 codec. It also features several advanced options that will be detailed here:
  - **Frame Rate:** Frame rate is the speed at which your movie displays. The faster the frame rate, the smoother the transition of movement between frames. It is important to remember that in order to create the best synchronization between the imported video and your existing flash movie, the framerates between your imported video and host FLA file should be the same. The next two options give you a choice: The *Same as FLA* option encodes the Flash video at the rate of the host FLA file, while the *Same as Source* reverses this and encodes at the frame rate of the imported video. There are also additional specific frame rates you can choose if you so desire. Remember, a higher framerate is better for animation, but larger in file size.
  - **Keyframes** are used in video to determine the difference between one clip and another. There are two options here, Automatic, in which Flash decides how often a

keyframe is created, and Custom, where you can choose your own options. If you don't know what this means, keep the setting to Automatic.

- The *Quality* option allows you to set the data rate of your Flash movie. The higher the bps (bits per second) setting, the higher quality the video will be, but also the larger the file size will be. Deciding on how your audience will see your movie clip is important in determining your Quality - if they are downloading the movie over a modem, a lower Quality may be in order. If they are watching it via CD-ROM or broadband internet connection, a higher setting would be prudent.
- *Resize Video* allows you to change the dimensions of your Flash Video. It is important to remember to keep the ratio of your video the same to avoid unwanted distortion. There are also several presets to help you here if you don't quite know what you are doing.
- *Encode Audio* - here is where you decide the data rate of the audio portion of your imported video will be. Like video, the higher the setting the better the quality of the audio, and larger proportionate file size.
- *Crop and Trim* allows you to change the borders of your video clip.
- At this point, you've determined pretty much everything you would need to know to import your video. Move on to the *Finish Video* screen and hit the Finish Button at the bottom to begin importing your video.

- 
- **NOTE :- THE CONTENTS ARE HEREBY REFERENCED FROM TUTORIALSPOINT AND OTHER SOURCES AVAILABLE FREELY ONLINE ON INTERNET FOR THE COMPLETION OF THESE DIGITAL CONTENTS FOR THE SUBJECT “MULTIMEDIA APPLICATIONS”**

## DATA COMMUNICATION

### 1. Introduction

**DATA COMMUNICATIONS** Data communications are the exchange of data between two devices via some form of transmission medium such as a wire cable. For data communications to occur, the communicating devices must be part of a communication system made up of a combination of hardware (physical equipment) and software (programs). The effectiveness of a data communications system depends on four fundamental characteristics: delivery, accuracy, timeliness, and jitter.

1. Delivery.

The system must deliver data to the correct destination. Data must be received by the intended device or user and only by that device or user.

2. Accuracy.

The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable.

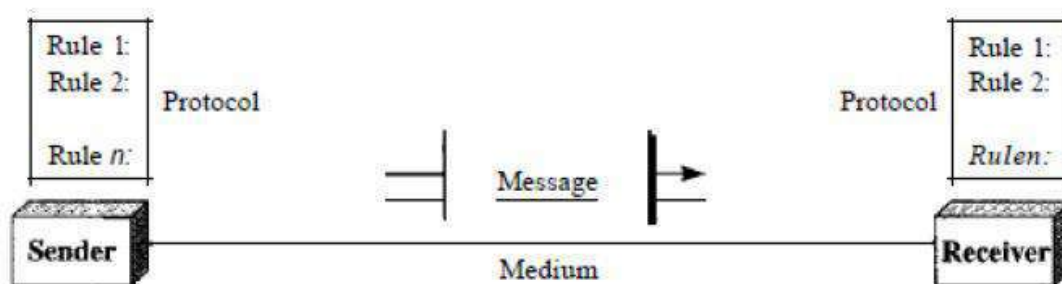
3. Timeliness.

The system must deliver data in a timely manner. Data delivered late are useless. In the case of video and audio, timely delivery means delivering data as they are produced, in the same order that they are produced, and without significant delay. This kind of delivery is called real-time transmission.

4. Jitter.

Jitter refers to the variation in the packet arrival time. It is the uneven delay in the delivery of audio or video packets.

Components:



A data communications system has five components.

1. Message. The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.

2. Sender. The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.

3. Receiver. The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.

4. Transmission medium. The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves

5. Protocol. A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices.

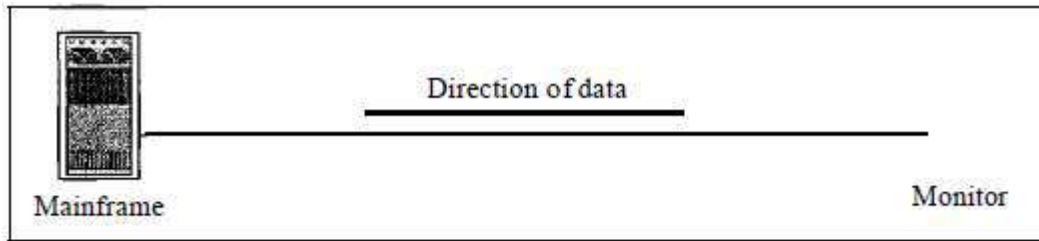
#### DATA TRANSMISSION MODES

Communication between two devices can be simplex, half-duplex, or full-duplex.

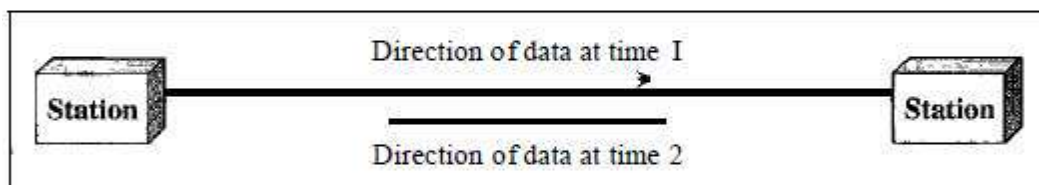
**Simplex:** In simplex mode, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit; the other can only receive. Keyboards and traditional monitors are examples of simplex devices. The keyboard can only introduce input; the monitor can only accept output. The simplex mode can use the entire capacity of the channel to send data in one direction.

**Half-Duplex:** In half-duplex mode, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa. In a half-duplex transmission, the entire capacity of a channel is taken over by whichever of the two devices is transmitting at the time. Walkie-talkies and CB (citizens band) radios are both half-duplex systems. The halfduplex mode is used in cases where there is no need for communication in both directions at the same time; the entire capacity of the channel can be utilized for each direction.

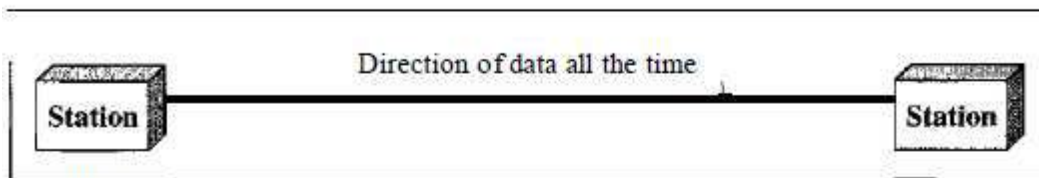
**Full-Duplex:** In full-duplex both stations can transmit and receive simultaneously. The full-duplex mode is like a two way street with traffic flowing in both directions at the same time. In full-duplex mode, signals going in one direction share the capacity of the link: with signals going in the other direction. One common example of full-duplex communication is the telephone network. When two people are communicating by a telephone line, both can talk and listen at the same time. The full-duplex mode is used when communication in both directions is required all the time. The capacity of the channel, however, must be divided between the two directions.



a. Simplex



b. Half-duplex



c. Full-duplex

**NETWORKS** A network is a set of devices (often referred to as nodes) connected by communication links. A node can be a computer, printer, or any other device capable of sending and/or receiving data generated by other nodes on the network.

**Network Criteria** A network must be able to meet a certain number of criteria. The most important of these are performance, reliability, and security.

**Performance:** Performance can be measured in many ways, including transit time and response time. Transit time is the amount of time required for a message to travel from one device to another. Response time is the elapsed time between an inquiry and a response. The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software.

**Reliability:** Network reliability is measured by the frequency of failure, the time it takes a link to recover from a failure, and the network's robustness in a catastrophe.

**Security:** Network security issues include protecting data from unauthorized access, protecting data from damage and development, and implementing policies and procedures for recovery from breaches and data losses.

## PHYSICAL STRUCTURES TYPES OF CONNECTIONS:

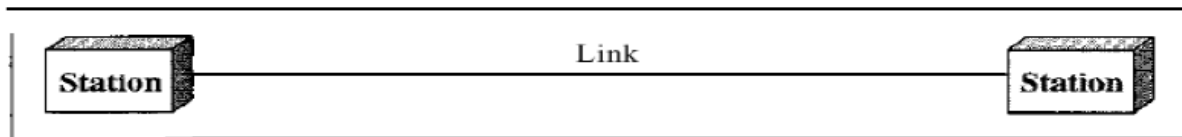
A network is two or more devices connected through links. A link is a communications pathway that transfers data from one device to another.

There are two possible types of connections:

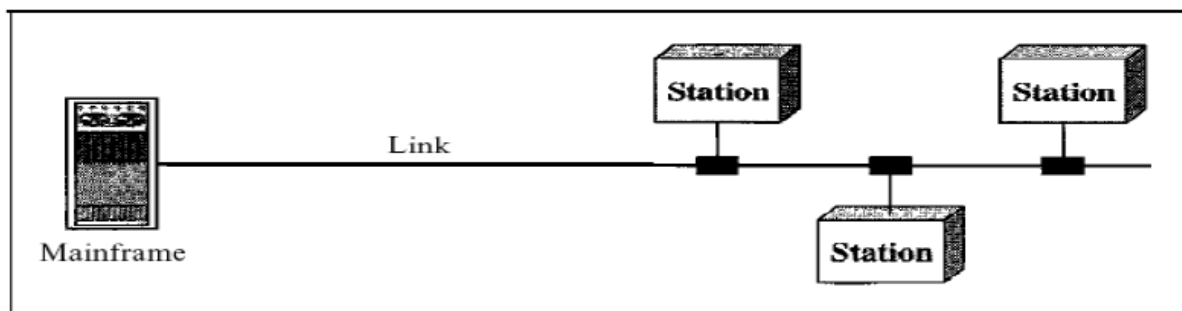
point-to-point and multipoint.

**Point-to-Point** A point-to-point connection provides a dedicated link between two devices. The entire capacity of the link is reserved for transmission between those two devices. Most point-to-point connections use an actual length of wire or cable to connect the two ends, but other options, such as microwave or satellite links, are also possible. When you change television channels by infrared remote control, you are establishing a point-to-point connection between the remote control and the television's control system.

**Multipoint** A multipoint (also called multidrop) connection is one in which more than two specific devices share a single link. In a multipoint environment, the capacity of the channel is shared, either spatially or temporally. If several devices can use the link simultaneously, it is a spatially shared connection. If users must take turns, it is a timeshared connection.



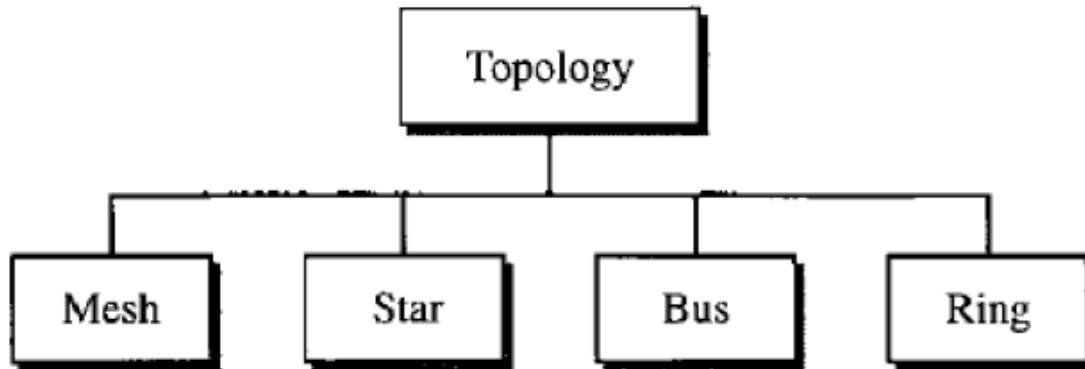
a. Point-to-point



b. Multipoint

## PHYSICAL TOPOLOGY:

The term physical topology refers to the way in which a network is laid out physically. One or more devices connect to a link; two or more links form a topology. The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another. There are four basic topologies possible: mesh, star, bus, and ring



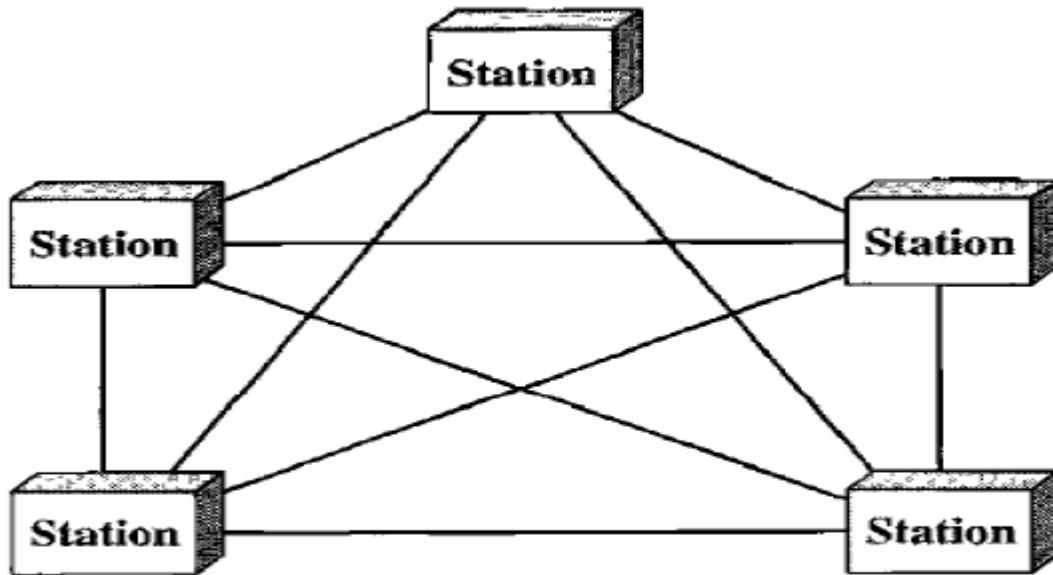
1. Mesh: In a mesh topology, every device has a dedicated point-to-point link to every other device. The term dedicated means that the link carries traffic only between the two devices it connects. To find the number of physical links in a fully connected mesh network with  $n$  nodes, we first consider that each node must be connected to every other node. Node 1 must be connected to  $n - 1$  nodes, node 2 must be connected to  $n - 1$  nodes, and finally node  $n$  must be connected to  $n - 1$  nodes. We need  $n(n - 1)$  physical links. However, if each physical link allows communication in both directions (duplex mode), we can divide the number of links by 2. In other words, we can say that in a mesh topology, we need  $n(n - 1) / 2$  duplex-mode links. To accommodate that many links, every device on the network must have  $n - 1$  input/output ports to be connected to the other  $n - 1$  stations.

Advantages:

1. The use of dedicated links guarantees that each connection can carry its own data load, thus eliminating the traffic problems that can occur when links must be shared by multiple devices.
2. A mesh topology is robust. If one link becomes unusable, it does not incapacitate the entire system.
3. There is the advantage of privacy or security. When every message travels along a dedicated line, only the intended recipient sees it. Physical boundaries prevent other users from gaining access to messages.
4. Point-to-point links make fault identification and fault isolation easy. Traffic can be routed to avoid links with suspected problems. This facility enables the network manager to discover the precise location of the fault and aids in finding its cause and solution.

Disadvantages:

1. Disadvantage of a mesh are related to the amount of cabling because every device must be connected to every other device.
2. Installation and reconnection are difficult.
3. The sheer bulk of the wiring can be greater than the available space (in walls, ceilings, or floors) can accommodate.
4. The hardware required to connect each link (I/O ports and cable) can be prohibitively

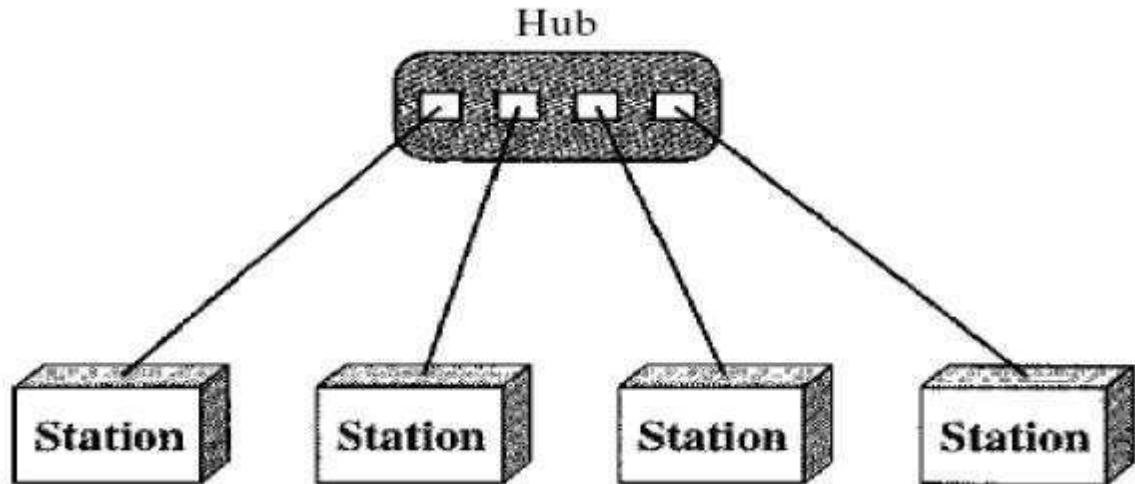


2. Star Topology: In a star topology, each device has a dedicated point-to-point link only to a central controller, usually called a hub. The devices are not directly linked to one another. Unlike a mesh topology, a star topology does not allow direct traffic between devices. The controller acts as an exchange: If one device wants to send data to another, it sends the data to the controller, which then relays the data to the other connected device .  
Advantages: 1. A star topology is less expensive than a mesh topology. In a star, each device needs only one link and one I/O port to connect it to any number of others.  
2. Easy to install and reconfigure.  
3. Far less cabling needs to be housed, and additions, moves, and deletions involve only one connection: between that device and the hub.  
4. Other advantage include robustness. If one link fails, only that link is affected. All other links remain active. This factor also lends itself to easy fault identification and fault isolation. As long as the hub is working, it can be used to monitor link problems and bypass defective links.

Disadvantages:

1. One big disadvantage of a star topology is the dependency of the whole topology on one single point, the hub. If the hub goes down, the whole system is dead. Although a star requires far less cable than a mesh, each node must be linked to a central hub. For this reason, often more cabling is required in a star than in some other topologies (such as ring or bus).





3. **BUS:** A bus topology is multipoint. One long cable acts as a backbone to link all the devices in a network. Nodes are connected to the bus cable by drop lines and taps. A drop line is a connection running between the device and the main cable. A tap is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core. As a signal travels along the backbone, some of its energy is transformed into heat. Therefore, it becomes weaker and weaker as it travels farther and farther. For this reason there is a limit on the number of taps a bus can support and on the distance between those taps.

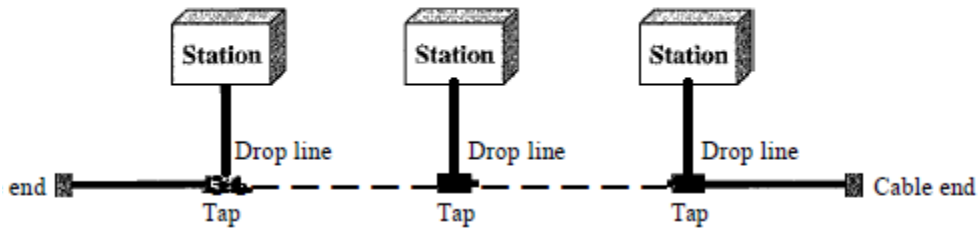
**Advantages:** Advantages of a bus topology include ease of installation. Backbone cable can be laid along the most efficient path, then connected to the nodes by drop lines of various lengths. In this way, a bus uses less cabling than mesh or star topologies. In a star, for example, four network devices in the same room require four lengths of cable reaching all the way to the hub. In a bus, this redundancy is eliminated. Only the backbone cable stretches through the entire facility. Each drop line has to reach only as far as the nearest point on the backbone.

**Disadvantages:**

Disadvantages include difficult reconnection and fault isolation. A bus is usually designed to be optimally efficient at installation. It can therefore be difficult to add new devices. Signal reflection at the taps can cause degradation in quality. This degradation can be controlled by limiting the number and spacing of devices connected to a given length of cable. Adding new devices may therefore require modification or replacement of the backbone. In addition, a fault or break in the bus cable stops all transmission, even between devices on the same side of the problem. The damaged area reflects signals back in the direction of origin, creating noise in both directions.

### *A bus topology connecting three stations*

---

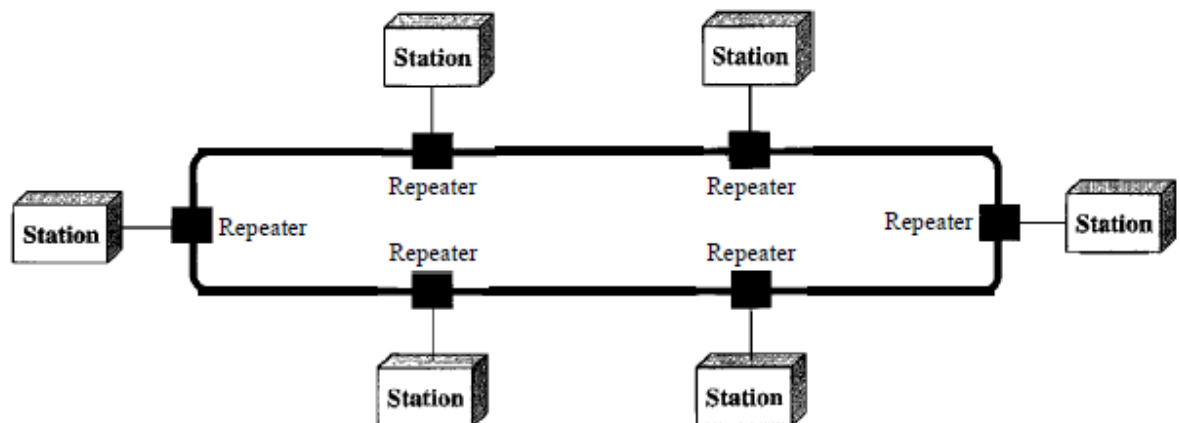


#### 4. RING:

In a ring topology, each device has a dedicated point-to-point connection with only the two devices on either side of it. A signal is passed along the ring in one direction, from device to device, until it reaches its destination. Each device in the ring incorporates a repeater. When a device receives a signal intended for another device, its repeater regenerates the bits and passes them along.

Advantages: A ring is relatively easy to install and reconfigure. Each device is linked to only its immediate neighbors (either physically or logically). To add or delete a device requires changing only two connections. The only constraints are media and traffic considerations (maximum ring length and number of devices). In addition, fault isolation is simplified. Generally in a ring, a signal is circulating at all times. If one device does not receive a signal within a specified period, it can issue an alarm. The alarm alerts the network operator to the problem and its location.

Disadvantages: Unidirectional traffic can be a disadvantage. In a simple ring, a break in the ring (such as a disabled station) can disable the entire network. This weakness can be solved by using a dual ring or a switch capable of closing off the break. Ring topology was prevalent when IBM introduced its local-area network Token Ring. Today, the need for higher-speed LANs has made this topology less popular.



## NETWORK CATEGORIES

### Local Area Networks (LAN):

Local area networks, generally called LANs, are privately-owned networks within a single building or campus of up to a few kilometers in size. They are widely used to connect personal computers and workstations in company offices and factories to share resources (e.g., printers) and exchange information. LANs are distinguished from other kinds of networks by three characteristics

- (1) Their size.
- (2) Their transmission technology
- (3) Their topology.

LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance. Knowing this bound makes it possible to use certain kinds of designs that would not otherwise be possible. It also simplifies network management. LANs may use a transmission technology consisting of a cable to which all the machines are attached, like the telephone company party lines once used in rural areas. Traditional LANs run at speeds of 10 Mbps to 100 Mbps, have low delay (microseconds or nanoseconds), and make very few errors. Newer LANs operate at up to 10 Gbps.

### Metropolitan Area Network (MAN):

A metropolitan area network, or MAN, covers a city. The best-known example of a MAN is the cable television network available in many cities. This system grew from earlier community antenna systems used in areas with poor over-the-air television reception. In these early systems, a large antenna was placed on top of a nearby hill and signal was then piped to the subscribers' houses. At first, these were locally-designed, ad hoc systems. Then companies began jumping into the business, getting contracts from city governments to wire up an entire city. The next step was television programming and even entire channels designed for cable only. Often these channels were highly specialized, such as all news, all sports, all cooking, all gardening, and so on. But from their inception until the late 1990s, they were intended for television reception only. Cable television is not the only MAN. Recent developments in high-speed wireless Internet access resulted in another MAN, which has been standardized as IEEE 802.16.

### Wide Area Network (WAN):

A wide area network, or WAN, spans a large geographical area, often a country or continent. It contains a collection of machines intended for running user (i.e., application) programs. These machines are called as hosts. The hosts are connected by a communication subnet, or just subnet for short. The hosts are owned by the customers (e.g., people's personal computers), whereas the communication subnet is typically owned and operated by a telephone company or Internet service provider. The job of the subnet is to carry messages from host to host, just as the telephone system carries words from speaker to listener. Separation of the pure communication aspects of the network (the subnet) from the application aspects (the hosts), greatly simplifies the complete network design. In most wide area networks, the subnet consists of two distinct components: transmission lines and switching elements. Transmission lines move

bits between machines. They can be made of copper wire, optical fiber, or even radio links. In most WANs, the network contains numerous transmission lines, each one connecting a pair of routers. If two routers that do not share a transmission line wish to communicate, they must do this indirectly, via other routers. When a packet is sent from one router to another via one or more intermediate routers, the packet is received at each intermediate router in its entirety, stored there until the required output line is free, and then forwarded. A subnet organized according to this principle is called a store-and-forward or packet-switched subnet. Nearly all wide area networks (except those using satellites) have store-and-forward subnets. When the packets are small and all the same size, they are often called cells. The principle of a packet-switched WAN is so important. Generally, when a process on some host has a message to be sent to a process on some other host, the sending host first cuts the message into packets, each one bearing its number in the sequence. These packets are then injected into the network one at a time in quick succession. The packets are transported individually over the network and deposited at the receiving host, where they are reassembled into the original message and delivered to the receiving process. Not all WANs are packet switched. A second possibility for a WAN is a satellite system. Each router has an antenna through which it can send and receive. All routers can hear the output from the satellite, and in some cases they can also hear the upward transmissions of their fellow routers to the satellite as well. Sometimes the routers are connected to a substantial point-to-point subnet, with only some of them having a satellite antenna. Satellite networks are inherently broadcast and are most useful when the broadcast property is important.

## 2. ANALOG AND DIGITAL

### Analog Data:

The term analog data refers to information that is continuous;

For example, an analog clock that has hour, minute, and second hands gives information in a continuous form; the movements of the hands are continuous.

Analog data, such as the sounds made by a human voice, take on continuous values. When someone speaks, an analog wave is created in the air. This can be captured by a microphone and converted to an analog signal or sampled and converted to a digital signal.

### Digital Data:

Digital data refers to information that has discrete states. For example, a digital clock that reports the hours and the minutes will change suddenly from 8:05 to 8:06.

Digital data takes on discrete values. For example, data are stored in computer memory in the form of 0s and 1s.

They can be converted to a digital signal or modulated into an analog signal for transmission across a medium.

### Analog and Digital Signals:

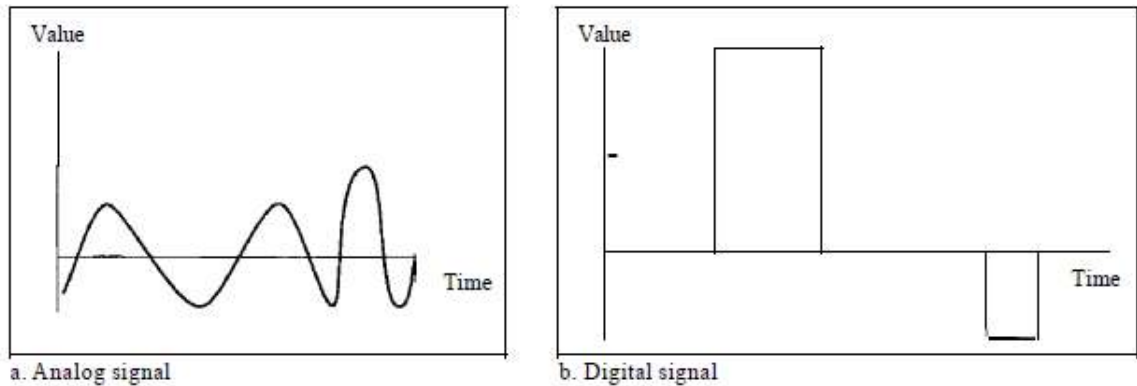
Like the data they represent, signals can be either analog or digital. An analog signal has infinitely many levels of intensity over a period of time. As the wave moves from value A to value B, it passes through and includes an infinite number of values along its path.

A digital signal, on the other hand, can have only a limited number of defined values. Although each value can be any number, it is often as simple as 1 and 0. The simplest way to show signals is by plotting them on a pair of perpendicular axes. The vertical axis represents the value or strength of a signal. The horizontal axis represents time. Figure below illustrates an analog signal and a digital signal. The curve representing the analog signal passes through an infinite number of points. The vertical lines of the digital signal, however, demonstrate the sudden jump that the signal makes from value to value.

---

Figure 3.1 *Comparison of analog and digital signals*

---



Periodic and Nonperiodic Signals: A periodic signal completes a pattern within a measurable time frame, called a period, and repeats that pattern over subsequent identical periods. The completion of one full pattern is called a cycle. A nonperiodic signal changes without exhibiting a pattern or cycle that repeats over time. PERIODIC ANALOG SIGNALS: Periodic analog signals can be classified as simple or composite. A simple periodic analog signal, a sine wave, cannot be decomposed into simpler signals. A composite periodic analog signal is composed of multiple sine waves. Sine Wave The sine wave is the most fundamental form of a periodic analog signal. When we visualize it as a simple oscillating curve, its change over the course of a cycle is smooth and consistent, a continuous, rolling flow. Figure below shows a sine wave. Each cycle consists of a single arc above the time axis followed by a single arc below it. Characteristics of Signals

1. Peak Amplitude

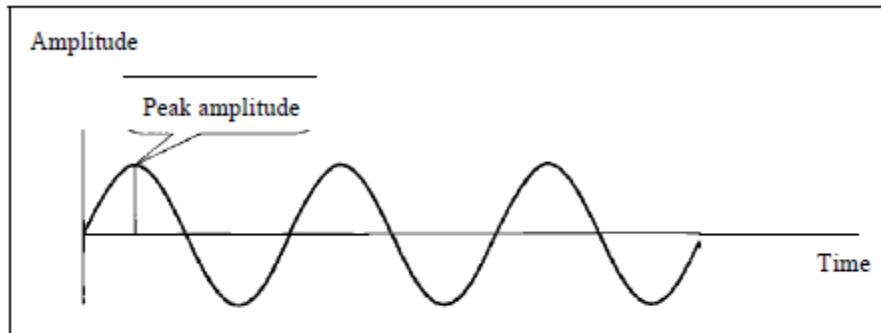
The peak amplitude of a signal is the absolute value of its highest intensity, proportional to the energy it carries. For electric signals, peak amplitude is normally measured in volts.

Figure below shows two signals and their peak amplitudes.

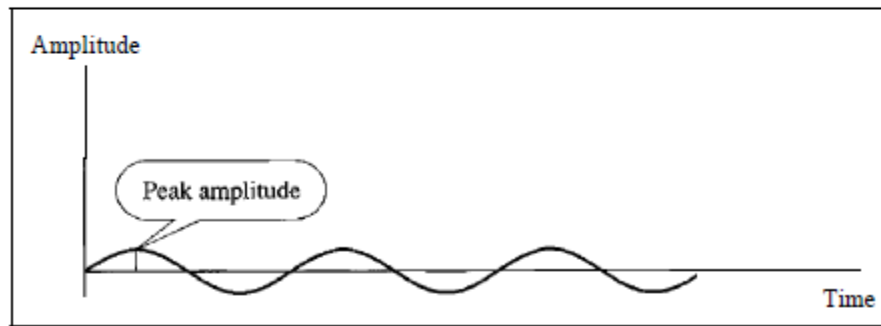
---

*Two signals with the same phase and frequency, but different amplitudes*

---



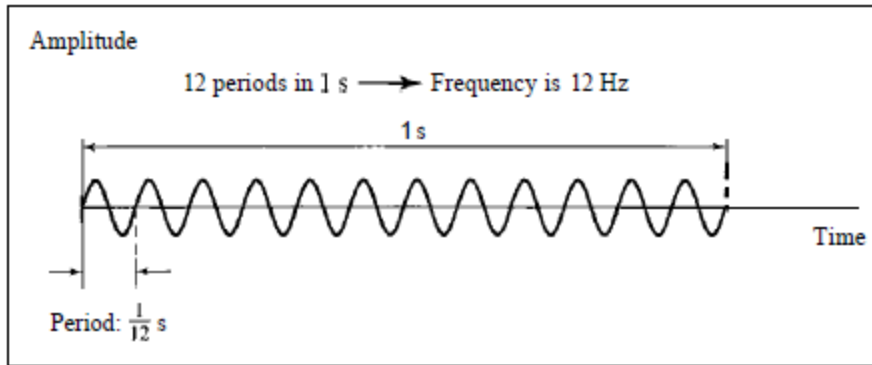
a. A signal with high peak amplitude



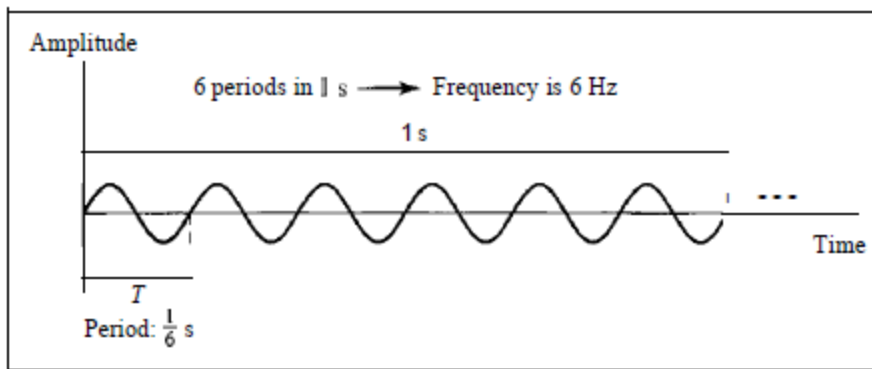
b. A signal with low peak amplitude

2. Period and Frequency Period refers to the amount of time, in seconds, a signal needs to complete 1 cycle. Frequency refers to the number of periods in 1 s. Note that period and frequency are just one characteristic defined in two ways. Period is the inverse of frequency, and frequency is the inverse of period, as the following formulas show.  $f=1/T$  and  $T=1/f$  Period is formally expressed in seconds. Frequency is formally expressed in Hertz (Hz), which is cycle per second.

*Two signals with the same amplitude and phase, but different frequencies*



a. A signal with a frequency of 12 Hz



b. A signal with a frequency of 6 Hz

3. Phase

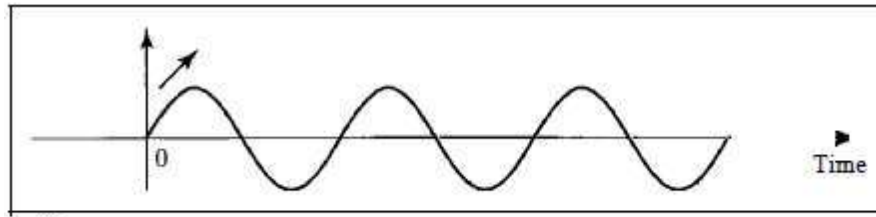
The term phase describes the position of the waveform relative to time 0. If we think of the wave as something that can be shifted backward or forward along the time axis, phase describes the amount of that shift. It indicates the status of the first cycle. Phase is measured in degrees or radians [ $360^\circ$  is  $2\pi$  rad;  $1^\circ$  is  $2\pi/360$  rad, and 1 rad is  $360/(2\pi)$ ]. A phase shift of  $360^\circ$  corresponds to a shift of a complete period; a phase shift of  $180^\circ$  corresponds to a shift



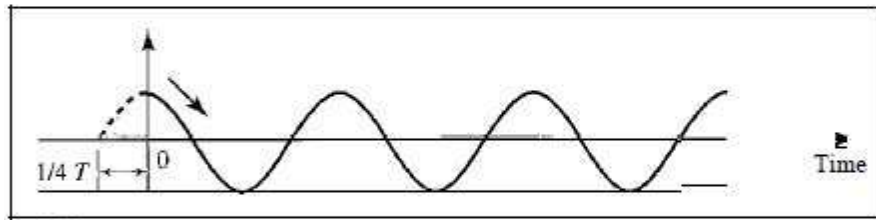
of one-half of a period; and a phase shift of  $90^\circ$  corresponds to a shift of one-quarter of a

*Three sine waves with the same amplitude and frequency, but different phases*

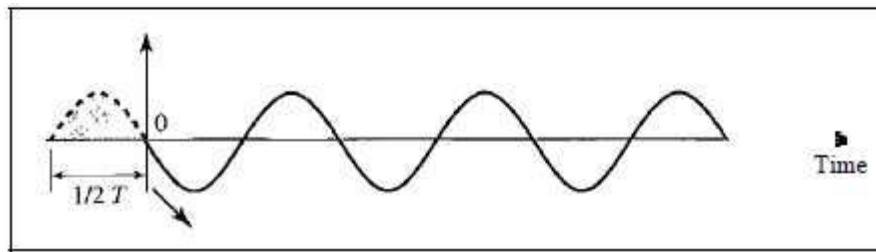
---



a. 0 degrees



b. 90 degrees



c. 180 degrees

period.

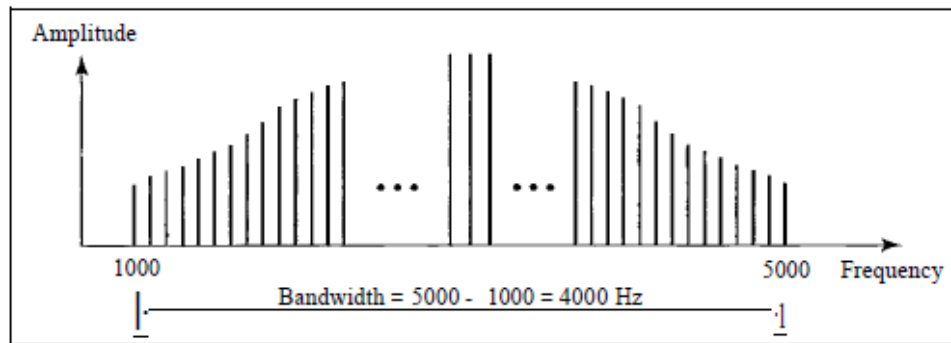
- I. A sine wave with a phase of  $0^\circ$  starts at time 0 with a zero amplitude. The amplitude is increasing
  - II. A sine wave with a phase of  $90^\circ$  starts at time 0 with a peak amplitude. The amplitude is decreasing
  - III. A sine wave with a phase of  $180^\circ$  starts at time 0 with a zero amplitude. The amplitude is decreasing
4. Wavelength Wavelength is another characteristic of a signal traveling through a transmission medium. Wavelength binds the period or the frequency of a simple sine wave to the propagation speed of the medium. While the frequency of a signal is independent of the medium, the wavelength depends on both the frequency and the medium. Wavelength is a property of any type of signal. In data communications, we often use wavelength to describe the transmission of light in an optical fiber. The wavelength is the distance a simple signal can travel in one period. Wavelength can be calculated if one is given the propagation speed (the speed of light) and the period of the signal.

Bandwidth The range of frequencies contained in a composite signal is its bandwidth. The bandwidth is normally a difference between two numbers. For example, if a composite signal contains frequencies between 1000 and 5000, its bandwidth is  $5000 - 1000$ , or 4000.

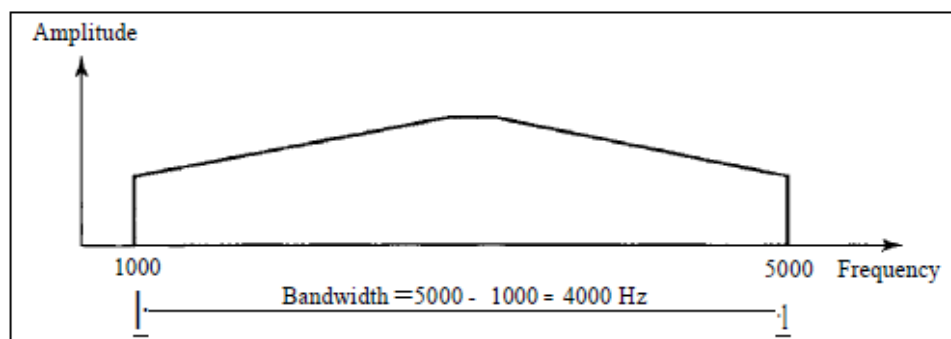
---

Figure 3.12 *The bandwidth of periodic and nonperiodic composite signals*

---

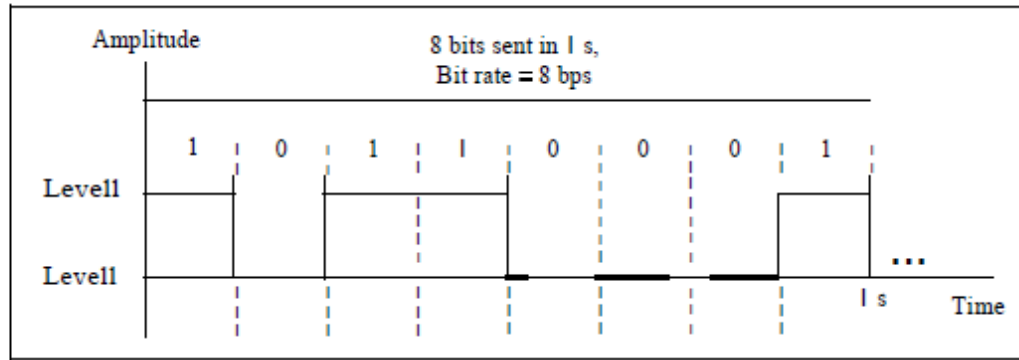


a. Bandwidth of a periodic signal

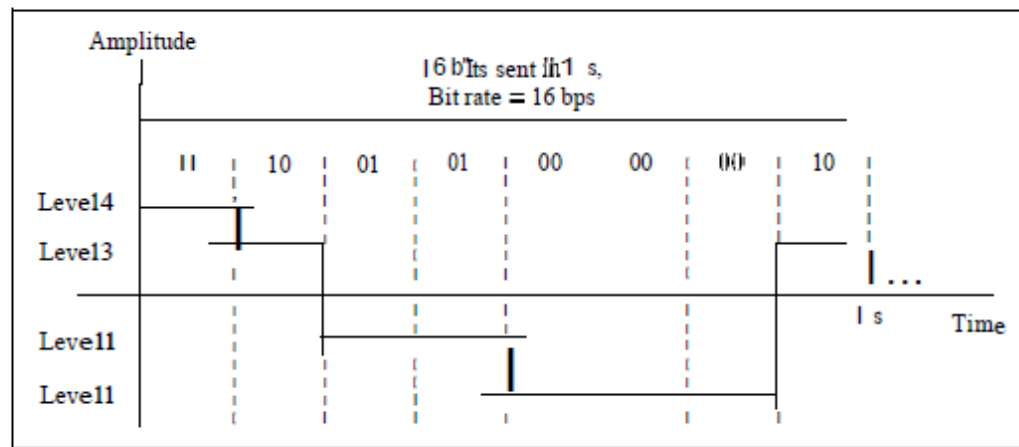


b. Bandwidth of a nonperiodic signal

DIGITAL SIGNALS In addition to being represented by an analog signal, information can also be represented by a digital signal. For example, a 1 can be encoded as a positive voltage and a 0 as zero voltage. A digital signal can have more than two levels. In this case, we can send more than 1 bit for each level.



a. A digital signal with two levels



b. A digital signal with four levels

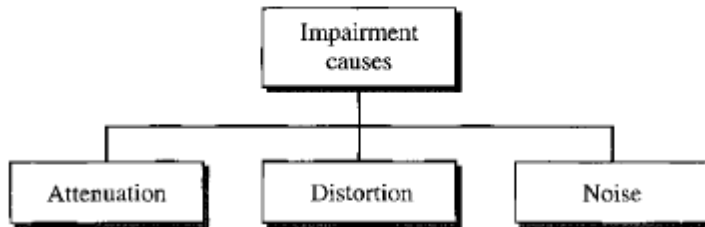
**Bit Rate** Most digital signals are nonperiodic, and thus period and frequency are not appropriate characteristics. Another term-bit rate is used to describe digital signals. The bit rate is the number of bits sent in 1s, expressed in bits per second (bps). Figure 3.16 shows the bit rate for two signal

#### TRANSMISSION IMPAIRMENT

Signals travel through transmission media, which are not perfect. The imperfection causes signal impairment. This means that the signal at the beginning of the medium is not the same as the signal at the end of the medium. What is sent is not what is received. Three causes of impairment are attenuation, distortion, and noise.

## Causes of impairment

---

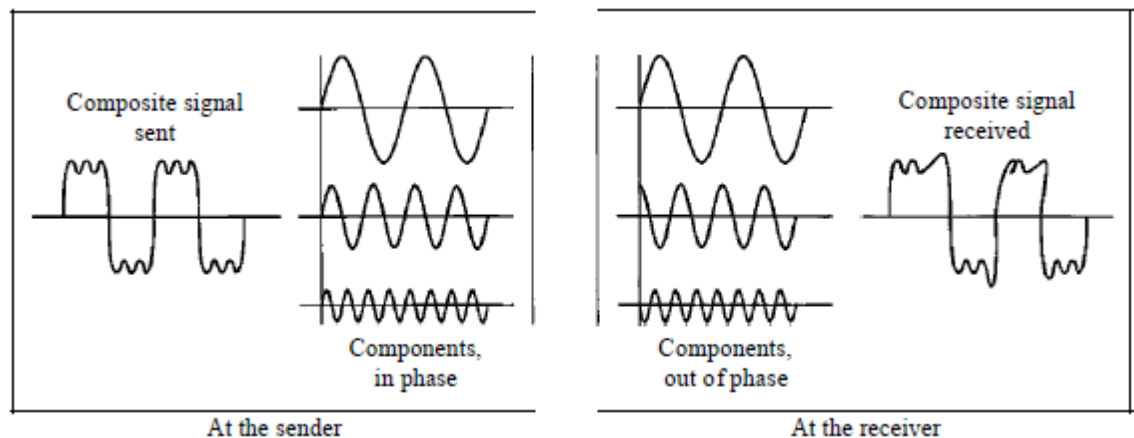


### 1. Attenuation

Attenuation means a loss of energy. When a signal, simple or composite, travels through a medium, it loses some of its energy in overcoming the resistance of the medium. That is why a wire carrying electric signals gets warm, if not hot, after a while. Some of the electrical energy in the signal is converted to heat. To compensate for this loss, amplifiers are used to amplify the signal. Attenuation is measured in terms of Decibels. The decibel (dB) measures the relative strengths of two signals or one signal at two different points. Note that the decibel is negative if a signal is attenuated and positive if a signal is amplified.  $dB = 10 \log_{10} P_2/P_1$  Variables  $P_1$  and  $P_2$  are the powers of a signal at points 1 and 2, respectively.

### 2. Distortion:

Distortion means that the signal changes its form or shape. Distortion can occur in a composite signal made of different frequencies. Each signal component has its own propagation speed through a medium and, therefore, its own delay in arriving at the final destination. Differences in delay may create a difference in phase if the delay is not exactly the same as the period duration. In other words, signal components at the receiver have phases different from what they had at the sender. The shape of the composite signal is therefore not the same.



### 3. Noise

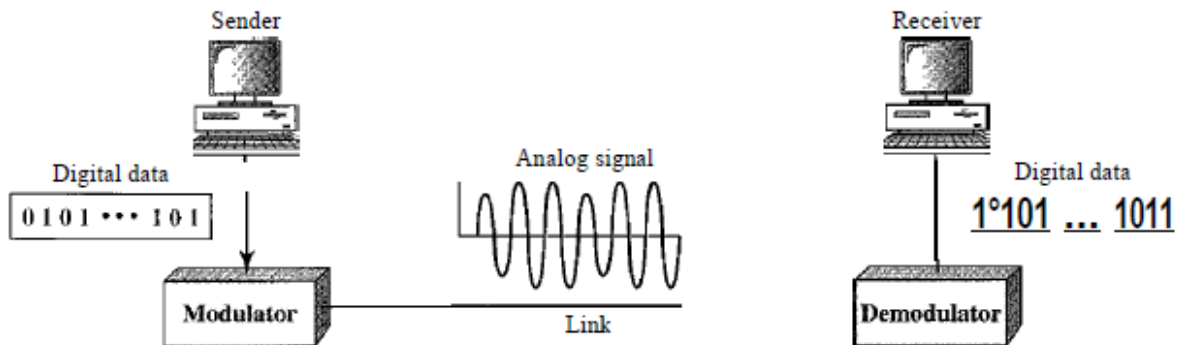
Noise is another cause of impairment. Several types of noise, such as thermal noise, induced noise, crosstalk, and impulse noise, may corrupt the signal. Thermal noise is the random

motion of electrons in a wire which creates an extra signal not originally sent by the transmitter. Induced noise comes from sources such as motors and appliances. These devices act as a sending antenna, and the transmission medium acts as the receiving antenna. Crosstalk is the effect of one wire on the other. One wire acts as a sending antenna and the other as the receiving antenna. Impulse noise is a spike (a signal with high energy in a very short time) that comes from power lines, lightning, and so on. Signal-to-Noise Ratio (SNR) The signal-to-noise ratio is defined as  $SNR = \text{Average Signal power} / \text{Average Noise Power}$  SNR is actually the ratio of what is wanted (signal) to what is not wanted (noise). A high SNR means the signal is less corrupted by noise; a low SNR means the signal is more corrupted by noise. Because SNR is the ratio of two powers, it is often described in decibel units, SNR dB, defined as  $SNR_{dB} = 10 \log_{10} SNR$

### 3. Digital and Analog Transmission

#### DIGITAL-TO-ANALOG CONVERSION

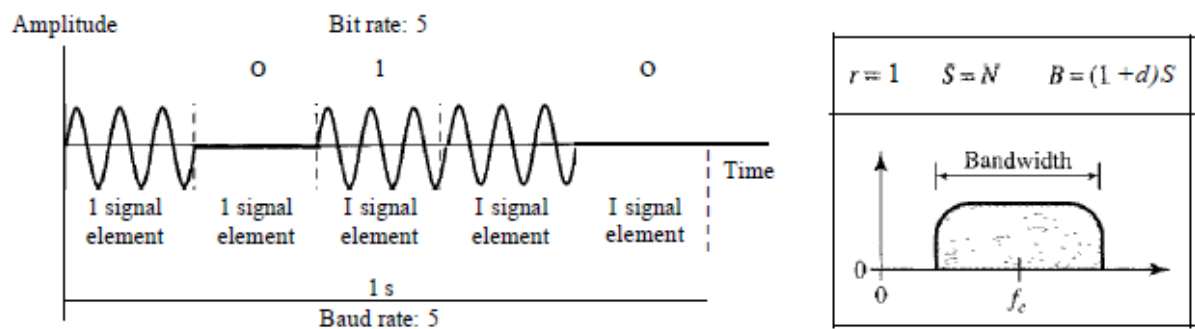
Digital-to-analog conversion is the process of changing one of the characteristics of an analog signal based on the information in digital data. Figure 5.1 shows the relationship between the digital information, the digital-to-analog modulating process, and the resultant analog signal.



A sine wave is defined by three characteristics: amplitude, frequency, and phase. When we vary anyone of these characteristics, we create a different version of that wave. So, by changing one characteristic of a simple electric signal, we can use it to represent digital data. Any of the three characteristics can be altered in this way, giving us at least three mechanisms for modulating digital data into an analog signal: amplitude shift keying (ASK), frequency shift keying (FSK), and phase shift keying (PSK). Aspects of Digital-to-Analog Conversion.

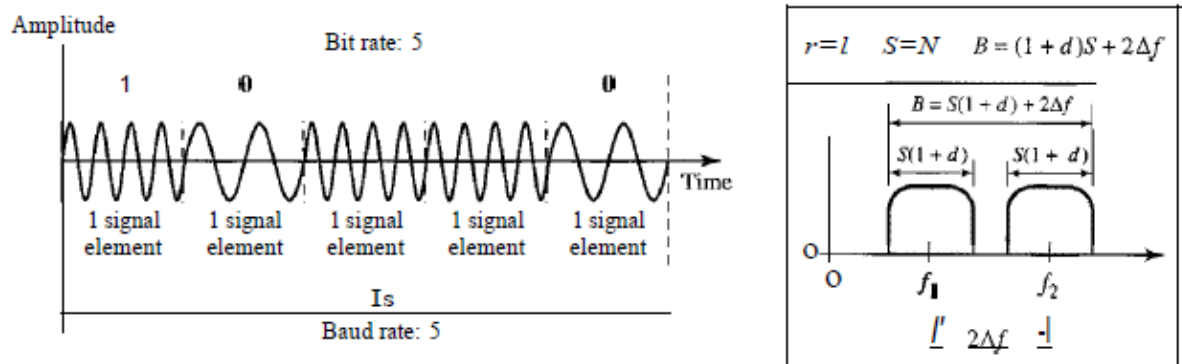
#### 1. Amplitude Shift Keying (ASK)

In amplitude shift keying, the amplitude of the carrier signal is varied to create signal elements. Both frequency and phase remain constant while the amplitude changes. Although we can have several levels (kinds) of signal elements, each with a different amplitude, ASK is normally implemented using only two levels. This is referred to as binary amplitude shift keying or on-off keying (OOK). The peak amplitude of one signal level is 0; the other is the same as the amplitude of the carrier frequency. Figure 5.3 gives a conceptual view of binary ASK.



## 2. Frequency Shift Keying (FSK)

In frequency shift keying, the frequency of the carrier signal is varied to represent data. The frequency of the modulated signal is constant for the duration of one signal element, but changes for the next signal element if the data element changes. Both peak amplitude and phase remain constant for all signal elements. One way to think about binary FSK (or BFSK) is to consider two carrier frequencies. In Figure 5.6, we have selected two carrier frequencies,  $f_1$  and  $f_2$ . We use the first carrier if the data element is 0; we use the second if the data element is 1. However, note that this is an unrealistic example used only for demonstration purposes. Normally the carrier frequencies are very high, and the difference between them is very small.



## 3. Phase Shift Keying (PSK)

In phase shift keying, the phase of the carrier is varied to represent two or more different signal elements. Both peak amplitude and frequency remain constant as the phase changes. Today, PSK is more common than ASK or FSK. The simplest PSK is binary PSK, in which we have only two signal elements, one with a phase of  $0^\circ$ , and the other with a phase of  $180^\circ$ .

## Analog To Analog Conversion Techniques

Analog-to-analog conversion, or analog modulation, is the representation of analog information by an analog signal. Modulation is needed if the medium is bandpass in nature or if only a bandpass channel is available to us.

An example is radio. The government assigns a narrow bandwidth to each radio station. The analog signal produced by each station is a low-pass signal, all in the same range. To be able to listen to different stations, the low-pass signals need to be shifted, each to a different range.

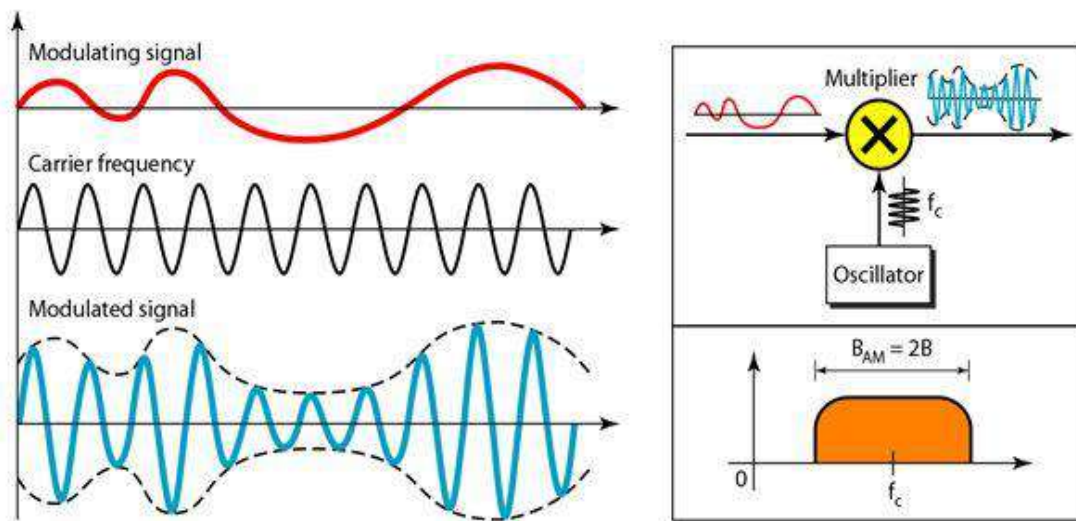
Analog-to-analog conversion can be accomplished in three ways:

### Amplitude Modulation (AM)

## Frequency Modulation (FM) Phase Modulation (PM)

### 1. Amplitude Modulation:

In AM transmission, the carrier signal is modulated so that its amplitude varies with the changing amplitudes of the modulating signal. The frequency and phase of the carrier remain the same. Only the amplitude changes to follow variations in the information. The following figure shows how this concept works. The modulating signal is the envelope of the carrier.



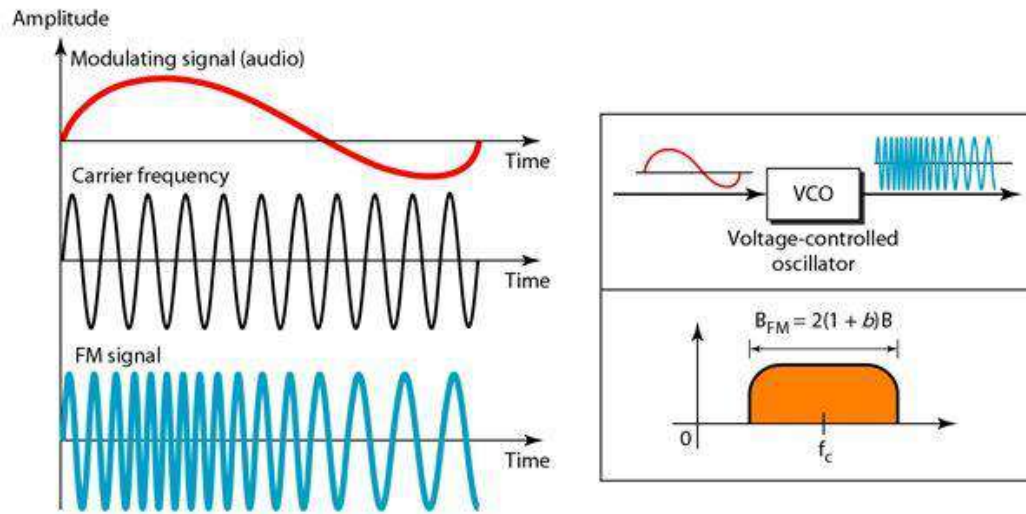
AM is normally implemented by using a simple multiplier because the amplitude of the carrier signal needs to be changed according to the amplitude of the modulating signal.

### 2. Frequency Modulation

In FM transmission, the frequency of the carrier signal is modulated to follow the changing voltage level (amplitude) of the modulating signal. The peak amplitude and phase of the carrier signal remain constant, but as the amplitude of the information signal changes, the frequency of the carrier changes correspondingly.



The following figure shows the relationships of the modulating signal, the carrier signal, and the resultant FM signal. FM is normally implemented by using a voltage-controlled oscillator as with FSK. The frequency of the oscillator changes according to the input voltage which is the amplitude of the modulating signal.



### ***Standard Bandwidth allocation for FM Radio:***

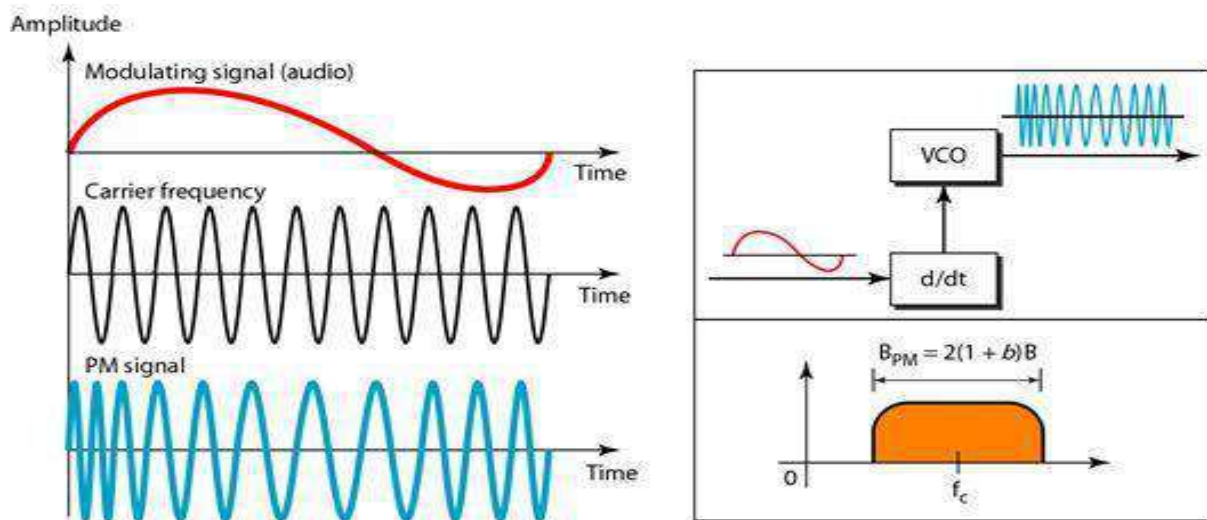
The bandwidth of an audio signal (speech and music) broadcast in stereo is almost 15 kHz. The FCC allows 200 kHz (0.2 MHz) for each station.

### **3. Phase Modulation:**

In PM transmission, the phase of the carrier signal is modulated to follow the changing voltage level (amplitude) of the modulating signal. The peak amplitude and frequency of the carrier signal remain constant, but as the amplitude of the information signal changes, the phase of the carrier changes correspondingly. It is proved mathematically that PM is the same as FM with one difference.

In FM, the instantaneous change in the carrier frequency is proportional to the amplitude of the modulating signal; in PM the instantaneous change in the carrier frequency is proportional to the derivative of the amplitude of the

modulating signal. The following figure shows the relationships of the modulating signal, the carrier signal, and the resultant PM signal.



PM is normally implemented by using a voltage-controlled oscillator along with a derivative. The frequency of the oscillator changes according to the derivative of the input voltage which is the amplitude of the modulating signal.

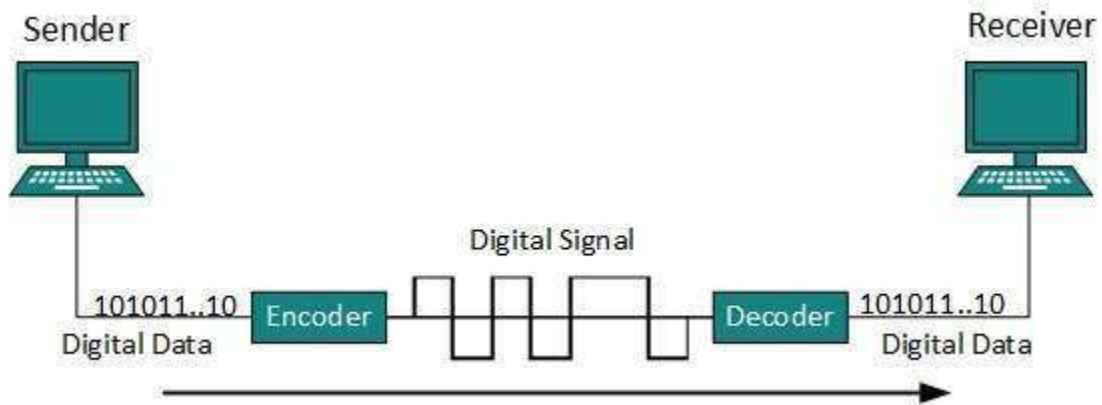
data or information can be stored in two ways, analog and digital. For a computer to use the data, it must be in discrete digital form. Similar to data, signals can also be in analog and digital form. To transmit data digitally, it needs to be first converted to digital form.

## Digital-to-Digital Conversion

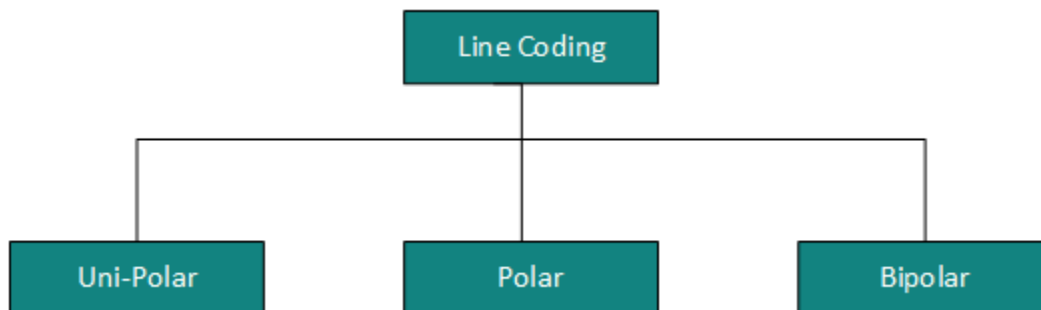
This section explains how to convert digital data into digital signals. It can be done in two ways, line coding and block coding. For all communications, line coding is necessary whereas block coding is optional.

## Line Coding

The process for converting digital data into digital signal is said to be Line Coding. Digital data is found in binary format. It is represented (stored) internally as series of 1s and 0s.

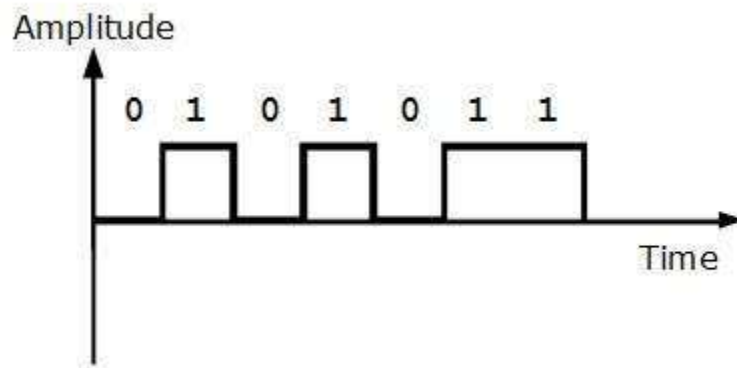


Digital signal is denoted by discrete signal, which represents digital data. There are three types of line coding schemes available:



## Uni-polar Encoding

Unipolar encoding schemes use single voltage level to represent data. In this case, to represent binary 1, high voltage is transmitted and to represent 0, no voltage is transmitted. It is also called Unipolar-Non-return-to-zero, because there is no rest condition i.e. it either represents 1 or 0.



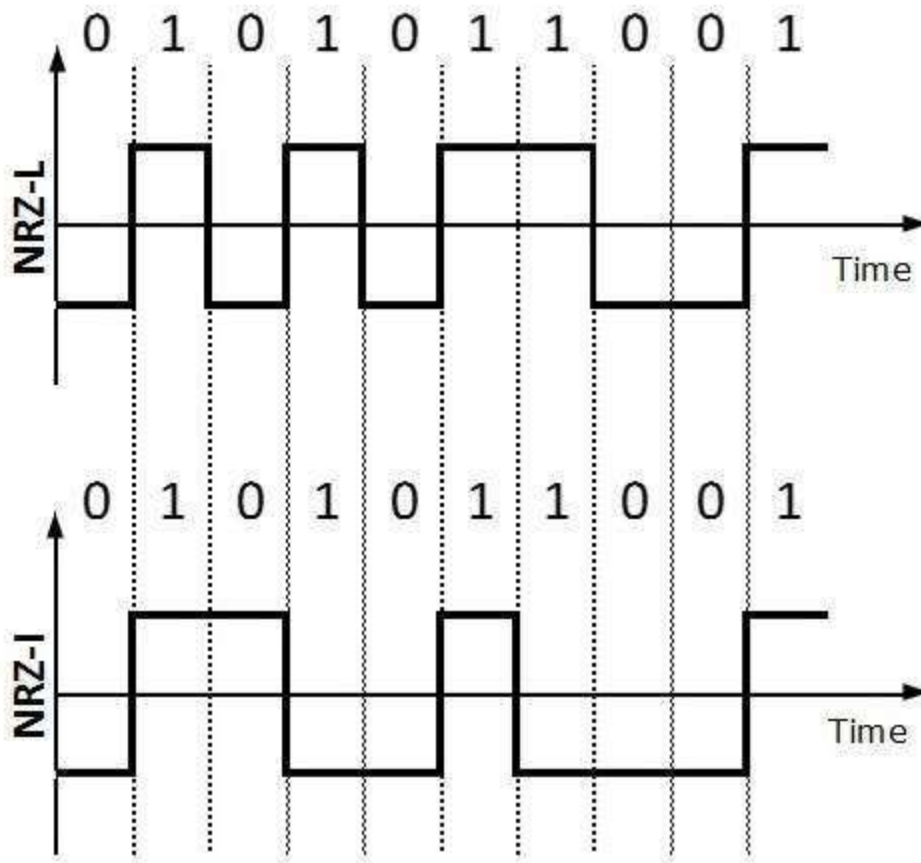
## Polar Encoding

Polar encoding scheme uses multiple voltage levels to represent binary values. Polar encodings is available in four types:

- Polar Non-Return to Zero (Polar NRZ)

It uses two different voltage levels to represent binary values. Generally, positive voltage represents 1 and negative value represents 0. It is also NRZ because there is no rest condition.

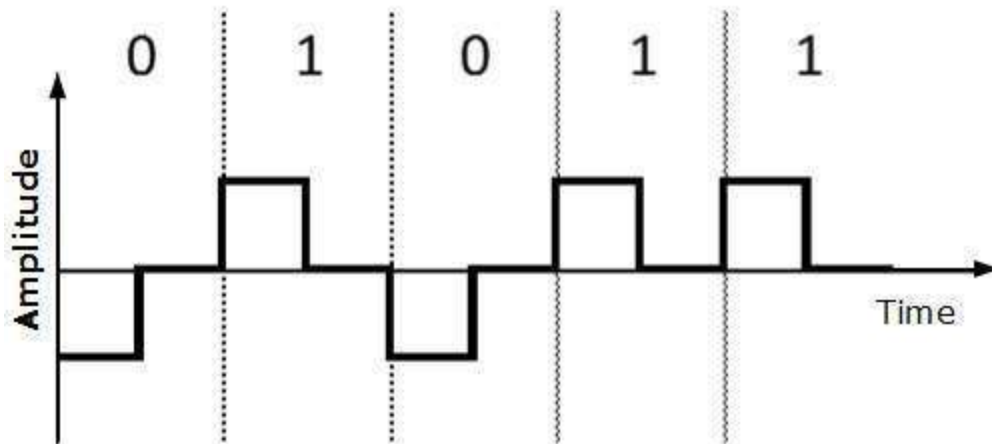
NRZ scheme has two variants: NRZ-L and NRZ-I.



NRZ-L changes voltage level at when a different bit is encountered whereas NRZ-I changes voltage when a 1 is encountered.

- Return to Zero (RZ)

Problem with NRZ is that the receiver cannot conclude when a bit ended and when the next bit is started, in case when sender and receiver's clock are not synchronized.



RZ uses three voltage levels, positive voltage to represent 1, negative voltage to represent 0 and zero voltage for none. Signals change during bits not between bits.

- Manchester

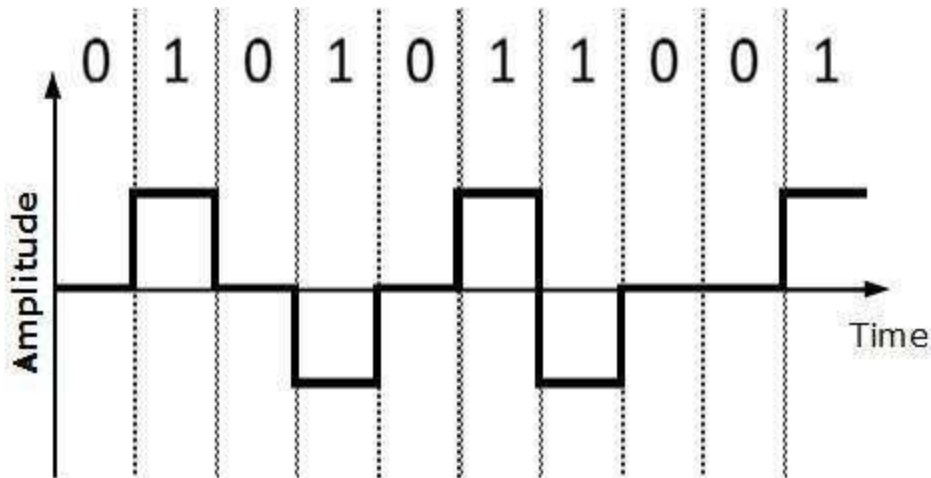
This encoding scheme is a combination of RZ and NRZ-L. Bit time is divided into two halves. It transits in the middle of the bit and changes phase when a different bit is encountered.

- Differential Manchester

This encoding scheme is a combination of RZ and NRZ-I. It also transit at the middle of the bit but changes phase only when 1 is encountered.

## Bipolar Encoding

Bipolar encoding uses three voltage levels, positive, negative and zero. Zero voltage represents binary 0 and bit 1 is represented by altering positive and negative voltages.



## Analog-to-Digital Conversion

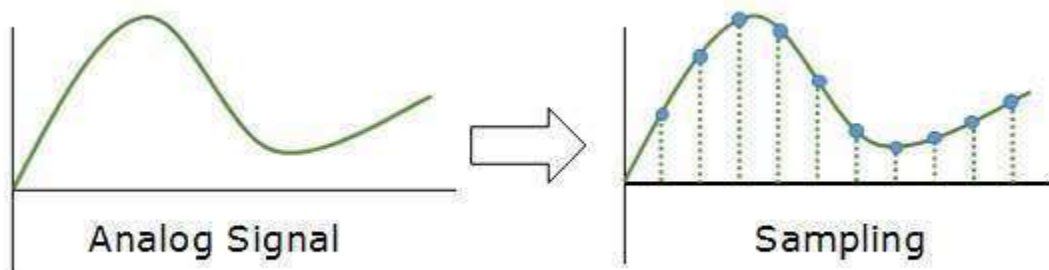
Microphones create analog voice and camera creates analog videos, which are treated as analog data. To transmit this analog data over digital signals, we need analog to digital conversion.

Analog data is a continuous stream of data in the wave form whereas digital data is discrete. To convert analog wave into digital data, we use Pulse Code Modulation (PCM).

PCM is one of the most commonly used method to convert analog data into digital form. It involves three steps:

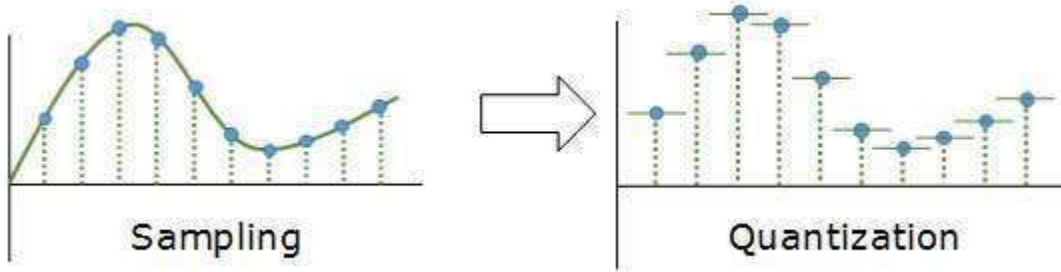
- Sampling
- Quantization
- Encoding.

### Sampling



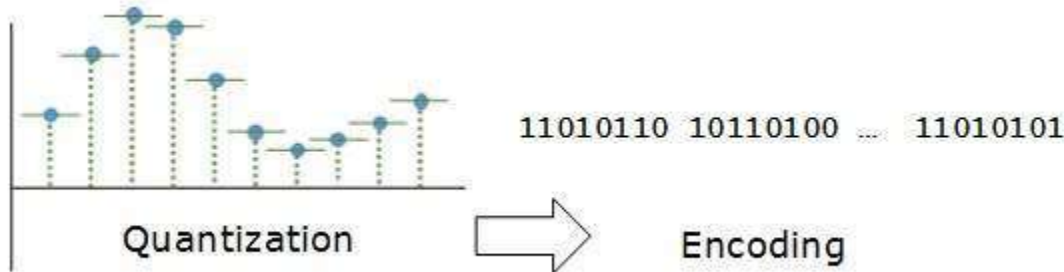
The analog signal is sampled every  $T$  interval. Most important factor in sampling is the rate at which analog signal is sampled. According to Nyquist Theorem, the sampling rate must be at least two times of the highest frequency of the signal.

## Quantization



Sampling yields discrete form of continuous analog signal. Every discrete pattern shows the amplitude of the analog signal at that instance. The quantization is done between the maximum amplitude value and the minimum amplitude value. Quantization is approximation of the instantaneous analog value.

## Encoding



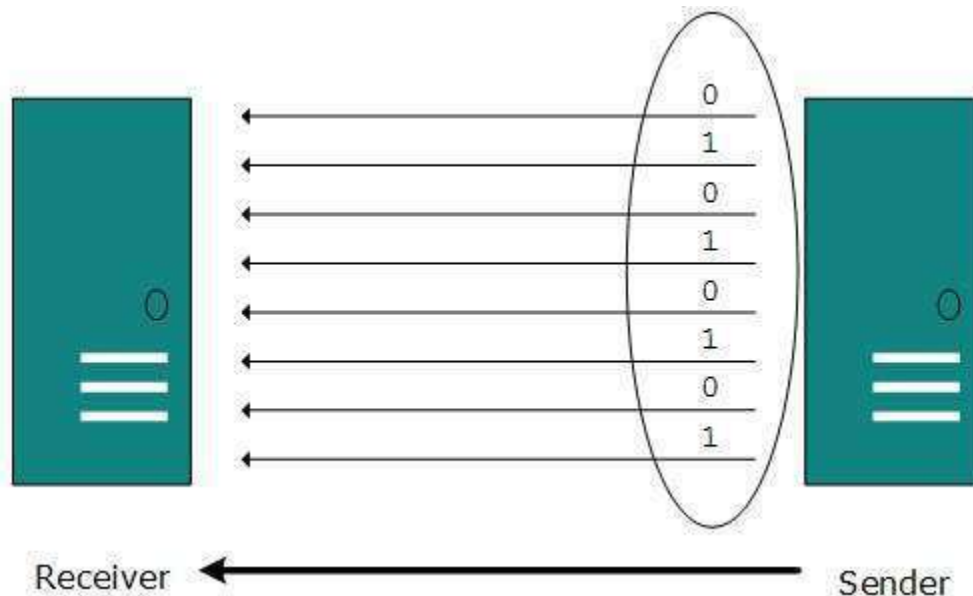
In encoding, each approximated value is then converted into binary format.

## Transmission Modes

The transmission mode decides how data is transmitted between two computers. The binary data in the form of 1s and 0s can be sent in two different modes: Parallel and Serial.



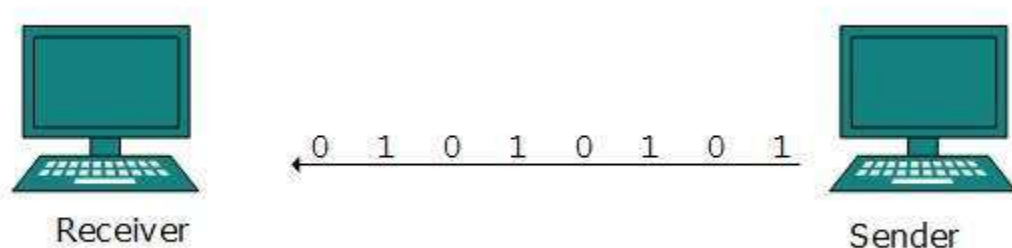
## Parallel Transmission



The binary bits are organized in-to groups of fixed length. Both sender and receiver are connected in parallel with the equal number of data lines. Both computers distinguish between high order and low order data lines. The sender sends all the bits at once on all lines. Because the data lines are equal to the number of bits in a group or data frame, a complete group of bits (data frame) is sent in one go. Advantage of Parallel transmission is high speed and disadvantage is the cost of wires, as it is equal to the number of bits sent in parallel.

## Serial Transmission

In serial transmission, bits are sent one after another in a queue manner. Serial transmission requires only one communication channel.



Serial transmission can be either asynchronous or synchronous.

## Asynchronous Serial Transmission

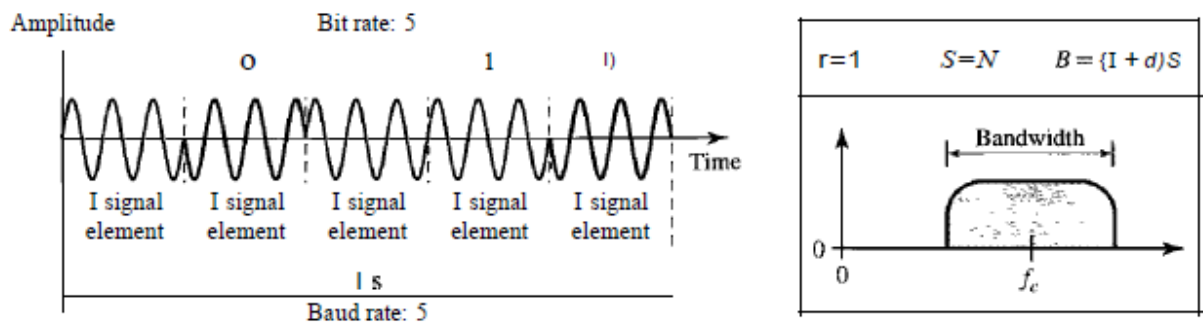
It is named so because there's no importance of timing. Data-bits have specific pattern and they help receiver recognize the start and end data bits. For example, a 0 is prefixed on every data byte and one or more 1s are added at the end.

Two continuous data-frames (bytes) may have a gap between them.

## Synchronous Serial Transmission

Timing in synchronous transmission has importance as there is no mechanism followed to recognize start and end data bits. There is no pattern or prefix/suffix method. Data bits are sent in burst mode without maintaining gap between bytes (8-bits). Single burst of data bits may contain a number of bytes. Therefore, timing becomes very important.

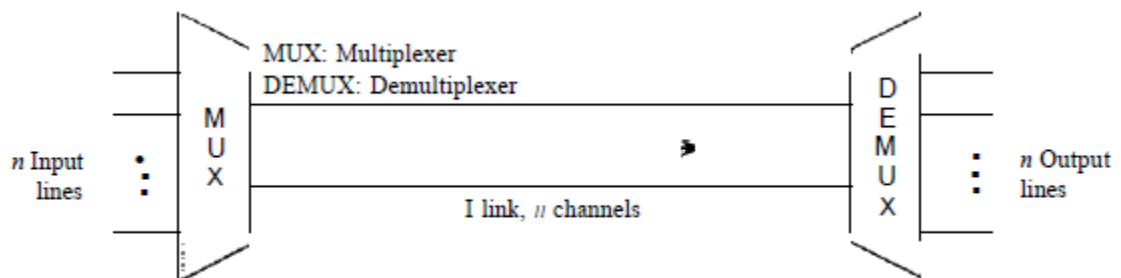
It is up to the receiver to recognize and separate bits into bytes. The advantage of synchronous transmission is high speed, and it has no overhead of extra header and footer bits as in asynchronous transmission.



## 4. MULTIPLEXING-FDM,WDM,TDM

### MULTIPLEXING

Whenever the bandwidth of a medium linking two devices is greater than the bandwidth needs of the devices, the link can be shared. Multiplexing is the set of techniques that allows the simultaneous transmission of multiple signals across a single data link. In a multiplexed system,  $n$  lines share the bandwidth of one link. Figure 6.1 shows the basic format of a multiplexed system. The lines on the left direct their transmission streams to a multiplexer (MUX), which combines them into a single stream (many-to-one). At the receiving end, that stream is fed into a demultiplexer (DEMUX), which separates the stream back into its component transmissions (one-to-many) and directs them to their corresponding lines. In the figure, the word link refers to the physical path. The word channel refers to the portion of a link that carries a transmission between a given pair of lines. One link can have many ( $n$ ) channels.

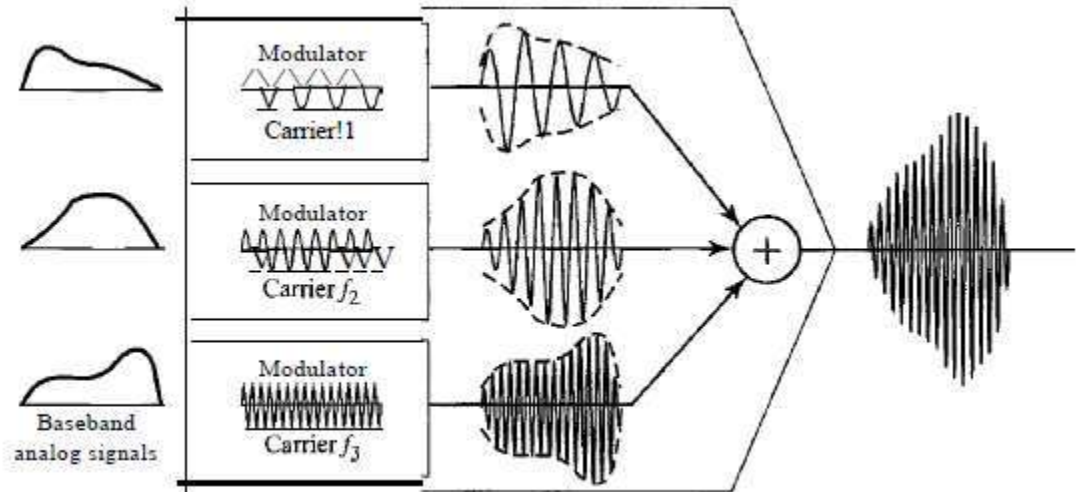


There are three basic multiplexing techniques:

Frequency-division multiplexing, wavelength-division multiplexing, and time-division multiplexing. The first two are techniques designed for analog signals, the third, for digital signals.

#### 1. Frequency-Division Multiplexing

Frequency-division multiplexing (FDM) is an analog technique that can be applied when the bandwidth of a link (in hertz) is greater than the combined bandwidths of the signals to be transmitted. In FDM, signals generated by each sending device modulate different carrier frequencies. These modulated signals are then combined into a single composite signal that can be transported by the link. Carrier frequencies are separated by sufficient bandwidth to accommodate the modulated signal. These bandwidth ranges are the channels through which the various signals travel. Channels can be separated by strips of unused bandwidth-guard bands-to prevent signals from overlapping. In addition, carrier frequencies must not interfere with the original data frequencies. Multiplexing Process Figure 6.4 is a conceptual illustration of the multiplexing process. Each source generates a signal of a similar frequency range. Inside the multiplexer, these similar signals modulates different carrier frequencies. The resulting modulated signals are then combined into a single composite signal that is sent out over a media link that has enough bandwidth to accommodate it.



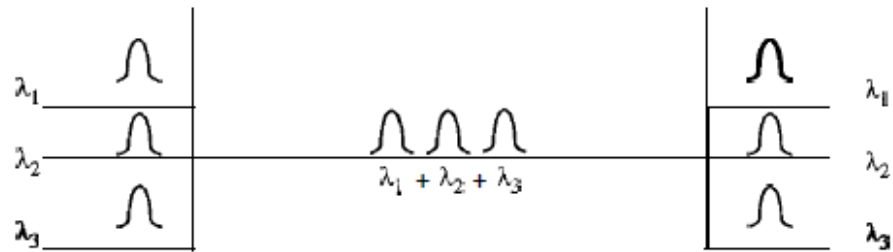
**Demultiplexing Process** The demultiplexer uses a series of filters to decompose the multiplexed signal into its constituent component signals. The individual signals are then passed to a demodulator that separates them from their carriers and passes them to the output lines. Figure 6.5 is a conceptual illustration of demultiplexing process.

## 2. Wavelength-Division Multiplexing

Wavelength-division multiplexing (WDM) is designed to use the high-data-rate capability of fiber-optic cable. The optical fiber data rate is higher than the data rate of metallic transmission cable. Using a fiber-optic cable for one single line wastes the available bandwidth. Multiplexing allows us to combine several lines into one. WDM is conceptually the same as FDM, except that the multiplexing and demultiplexing involve optical signals transmitted through fiber-optic channels. The idea is the same: We are combining different signals of different frequencies. The difference is that the frequencies are very high. Figure 6.10 gives a conceptual view of a WDM multiplexer and demultiplexer. Very narrow bands of light from different sources are combined to make a wider band of light. At the receiver, the signals are separated by the demultiplexer.

One application of WDM is the SONET network in which multiple optical fiber lines are multiplexed and demultiplexed.

Figure 6.10 *Wavelength-division multiplexing*



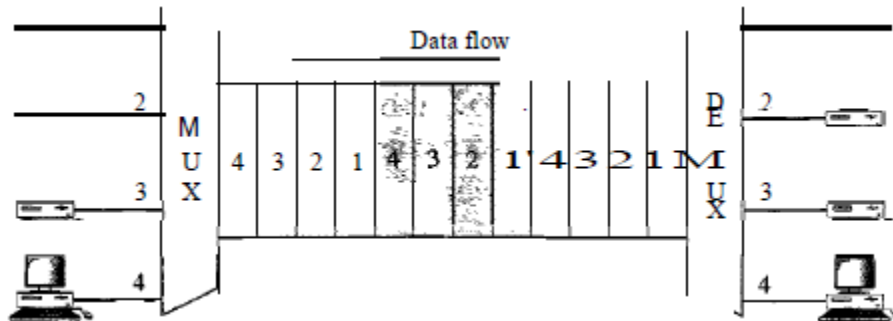

---

**WDM is an analog multiplexing technique to combine optical signals.**

---

3. Time-Division Multiplexing

Time Division multiplexing (TDM) is a digital process that allows several connections to share the high bandwidth of a line. Instead of sharing a portion of the bandwidth as in FDM, time is shared. Each connection occupies a portion of time in the link. Figure 6.12 gives a conceptual view of TDM. Note that the same link is used as in FDM; here, however, the link is shown sectioned by time rather than by frequency. In the figure, portions of signals 1,2,3, and 4 occupy the link sequentially.



## 5. TRANSMISSION MEDIA

**TRANSMISSION MEDIA** A transmission medium can be broadly defined as anything that can carry information from a source to a destination. For example, the transmission medium for two people having a dinner conversation is the air. The air can also be used to convey the message in a smoke signal or semaphore. For a written message, the transmission medium might be a mail carrier, a truck, or an airplane. In data communications the definition of the information and the transmission medium is more specific. The transmission medium is usually free space, metallic cable, or fiber-optic cable. The information is usually a signal that is the result of a conversion of data from another form.

### Guided Media

Guided media, which are those that provide a conduit from one device to another, include twisted-pair cable, coaxial cable, and fiber-optic cable. A signal traveling along any of these media is directed and contained by the physical limits of the medium. Twisted-pair and coaxial cable use metallic (copper) conductors that accept and transport signals in the form of electric current. Optical fiber is a cable that accepts and transports signals in the form of light.

#### 1. Twisted-Pair Cable

A twisted pair consists of two conductors (normally copper), each with its own plastic insulation, twisted together.



One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two. In addition to the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted signals. If the two wires are parallel, the effect of these unwanted signals is not the same in both wires because they are at different locations relative to the noise or crosstalk sources (e.g., one is closer and the other is farther). This results in a difference at the receiver. By twisting the pairs, a balance is

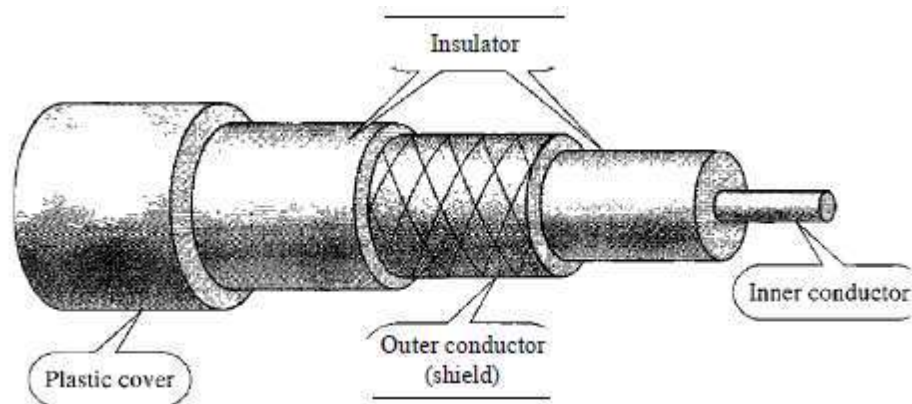
maintained. For example, suppose in one twist, one wire is closer to the noise source and the other is farther; in the next twist, the reverse is true. Twisting makes it probable that both wires are equally affected by external influences (noise or crosstalk). This means that the receiver, which calculates the difference between the two, receives no unwanted signals. The unwanted signals are mostly canceled out. From the above discussion, it is clear that the number of twists per unit of length (e.g., inch) has some effect on the quality of the cable.

#### Applications

Twisted-pair cables are used in telephone lines to provide voice and data channels. The local loop-the line that connects subscribers to the central telephone office-commonly consists of unshielded twisted-pair cables. The DSL lines that are used by the telephone companies to provide high-data-rate connections also use the high-bandwidth capability of unshielded twistedpair cables. Local-area networks, such as IOBase-T and IOOBase-T, also use twisted-pair cables.

#### 2. Coaxial Cable

Coaxial cable (or coax) carries signals of higher frequency ranges than those in twisted pair cable, in part because the two media are constructed quite differently. Instead of having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two. The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover .



#### Applications

Coaxial cable was widely used in analog telephone networks where a single coaxial network could carry 10,000 voice signals. Later it was used in digital telephone networks where a single coaxial cable could carry digital data up to 600 Mbps. However, coaxial cable in telephone networks has largely been replaced today with fiber-optic cable. Cable TV networks also use coaxial cables. In the traditional cable TV network, the entire network used coaxial cable. Later, however, cable TV providers replaced most of the media with fiber-optic cable; hybrid networks use coaxial cable only at the

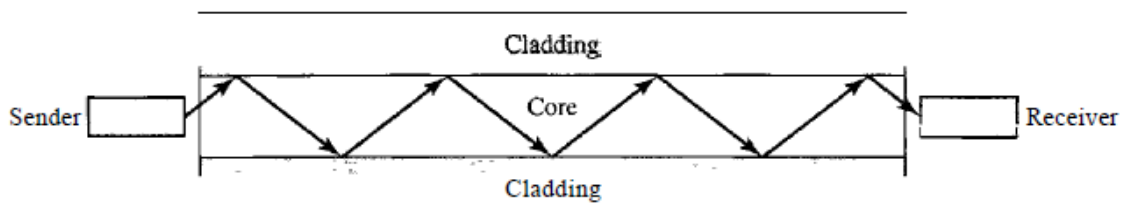
network boundaries, near the consumer premises. Cable TV uses RG-59 coaxial cable. Another common application of coaxial cable is in traditional Ethernet LANs. Because of its high bandwidth, and consequently high data rate, coaxial cable was chosen for digital transmission in early Ethernet LANs.

### 3. Fiber Optic Cable:

A fiber-optic cable is made of glass or plastic and transmits signals in the form of light. To understand optical fiber, we first need to explore several aspects of the nature of light. Light travels in a straight line as long as it is moving through a single uniform medium. If a ray of light traveling through one substance suddenly enters another substance (of a different density), the ray changes direction. Optical fibers use reflection to guide light through a channel. A glass or plastic core is surrounded by a cladding of less dense glass or plastic. The difference in density of the two materials must be such that a beam of light moving through the core is reflected off the cladding instead of being refracted into it.

#### Applications

Fiber-optic cable is often found in backbone networks because its wide bandwidth is cost-effective. Today, with wavelength-division multiplexing (WDM), we can transfer data at a rate of 1600 Gbps. The SONET network provides such a backbone.



#### Advantages and Disadvantages of Optical Fiber

##### Advantages

Fiber-optic cable has several advantages over metallic cable (twisted pair or coaxial).

- Higher bandwidth. Fiber-optic cable can support dramatically higher bandwidths (and hence data rates) than either twisted-pair or coaxial cable. Currently, data rates and bandwidth utilization over fiber-optic cable are limited not by the medium but by the signal generation and reception technology available.
- Less signal attenuation. Fiber-optic transmission distance is significantly greater than that of other guided media. A signal can run for 50 km without requiring regeneration. We need repeaters every 5 km for coaxial or twisted-pair cable.
- Immunity to electromagnetic interference. Electromagnetic noise cannot affect fiber-optic cables.
- Resistance to corrosive materials. Glass is more resistant to corrosive materials than copper.
- Light weight. Fiber-optic cables are much lighter than copper cables.
- Greater immunity to tapping. Fiber-optic cables are more immune to tapping than copper cables. Copper cables create antenna effects that can easily be tapped.



## Disadvantages

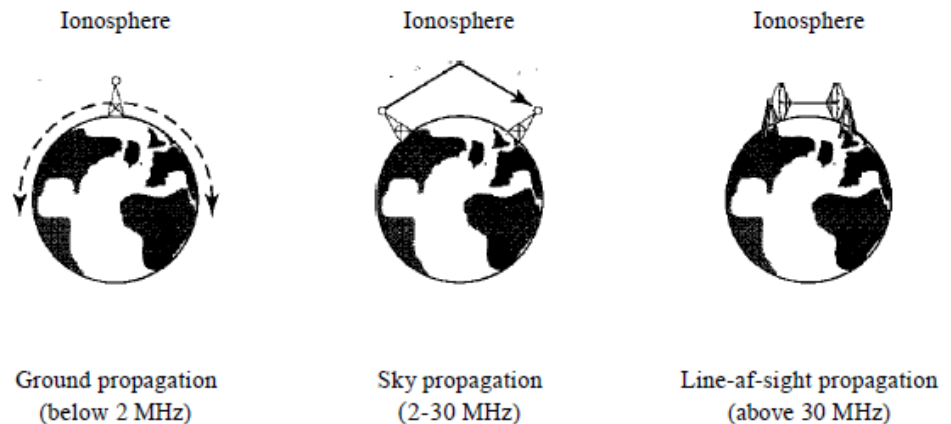
There are some disadvantages in the use of optical fiber.

- Installation and maintenance. Fiber-optic cable is a relatively new technology. Its installation and maintenance require expertise that is not yet available everywhere.
- Unidirectional light propagation. Propagation of light is unidirectional. If we need bidirectional communication, two fibers are needed.
- Cost. The cable and the interfaces are relatively more expensive than those of other guided media. If the demand for bandwidth is not high, often the use of optical fiber cannot be justified.

## UNGUIDED MEDIA:

### WIRELESS

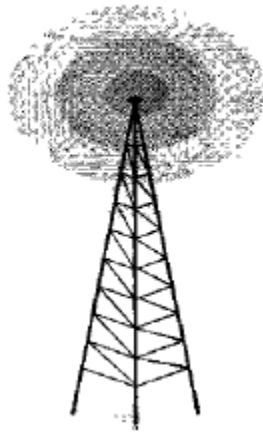
Unguided media transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as wireless communication.



Signals are normally broadcast through free space and thus are available to anyone who has a device capable of receiving them. Unguided signals can travel from the source to destination in several ways: ground propagation, sky propagation, and line-of-sight propagation. In ground propagation, radio waves travel through the lowest portion of the atmosphere, hugging the earth. These low-frequency signals emanate in all directions from the transmitting antenna and follow the curvature of the planet. Distance depends on the amount of power in the signal: The greater the power, the greater the distance. In sky propagation, higher-frequency radio waves radiate upward into the ionosphere where they are reflected back to earth. This type of transmission allows for greater distances with lower output power. In line-of-sight propagation, very high-frequency signals are transmitted in straight lines directly from antenna to antenna. Antennas must be directional, facing each other, and either tall enough or close enough together not to be affected by the curvature of the earth. Line-of-sight propagation is tricky because radio transmissions cannot be completely focused.

## 1. Radio Waves

Waves ranging in frequencies between 3 kHz and 1 GHz are called radio waves. Radio waves, for the most part, are omnidirectional. When an antenna transmits radio waves, they are propagated in all directions. This means that the sending and receiving antennas do not have to be aligned. A sending antenna sends waves that can be received by any receiving antenna. The omnidirectional property has a disadvantage, too. The radio waves transmitted by one antenna are susceptible to interference by another antenna that may send signals using the same frequency or band. Radio waves, particularly those waves that propagate in the sky mode, can travel long distances. This makes radio waves a good candidate for long-distance broadcasting such as AM radio. Radio waves, particularly those of low and medium frequencies, can penetrate walls. This characteristic can be both an advantage and a disadvantage. It is an advantage because, for example, an AM radio can receive signals inside a building. It is a disadvantage because we cannot isolate a communication to just inside or outside a building. The radio wave band is relatively narrow, just under 1 GHz, compared to the microwave band. When this band is divided into sub bands, the sub bands are also narrow, leading to a low data rate for digital communications.



### Applications

The omnidirectional characteristics of radio waves make them useful for multicasting, in which there is one sender but many receivers. AM and FM radio, television, maritime radio, cordless phones, and paging are examples of multicasting.

## 2. Microwaves

Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves. Microwaves are unidirectional. When an antenna transmits microwave waves, they can be narrowly focused. This means that the sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage.

A pair of antennas can be aligned without interfering with another pair of aligned antennas.

The following describes some characteristics of microwave propagation: a. Microwave propagation is line-of-sight. Since the towers with the mounted antennas need to be in direct sight of each other, towers that are far apart need to be very tall. The curvature of the earth as well as other blocking obstacles do not allow two short towers to communicate by using microwaves. Repeaters are often needed for long distance communication.

b. Very high-frequency microwaves cannot penetrate walls. This characteristic can be a disadvantage if receivers are inside buildings.

c. The microwave band is relatively wide, almost 299 GHz. Therefore wider sub bands can be assigned, and a high data rate is possible d. Use of certain portions of the band requires permission from authorities.

### 3. Infrared

Infrared waves, with frequencies from 300 GHz to 400 THz (wavelengths from 1 mm to 770 nm), can be used for short-range communication. Infrared waves, having high frequencies, cannot penetrate walls. This advantageous characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room. When we use our infrared remote control, we do not interfere with the use of the remote by our neighbors. However, this same characteristic makes infrared signals useless for long-range communication. In addition, we cannot use infrared waves outside a building because the sun's rays contain infrared waves that can interfere with the communication.

#### Applications

The infrared band, almost 400 THz, has an excellent potential for data transmission. Such a wide bandwidth can be used to transmit digital data with a very high data rate. The Infrared Data Association (IrDA), an association for sponsoring the use of infrared waves, has established standards for using these signals for communication between devices such as keyboards, mice, PCs, and printers.

For example, some manufacturers provide a special port called the IrDA port that allows a wireless keyboard to communicate with a PC. The standard originally defined a data rate of 75 kbps for a distance up to 8 m. The recent standard defines a data rate of 4 Mbps. Infrared signals defined by IrDA transmit through line of sight; the IrDA port on the keyboard needs to point to the PC for transmission to occur

## 6.ERROR DETECTION AND CORRECTION

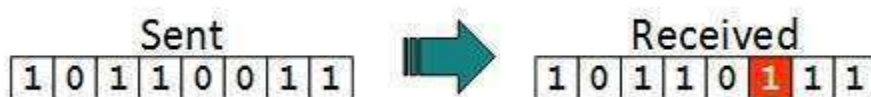
There are many reasons such as noise, cross-talk etc., which may help data to get corrupted during transmission. The upper layers work on some generalized view of network architecture and are not aware of actual hardware data processing. Hence, the upper layers expect error-free transmission between the systems. Most of the applications would not function expectedly if they receive erroneous data. Applications such as voice and video may not be that affected and with some errors they may still function well.

Data-link layer uses some error control mechanism to ensure that frames (data bit streams) are transmitted with certain level of accuracy. But to understand how errors is controlled, it is essential to know what types of errors may occur.

### Types of Errors

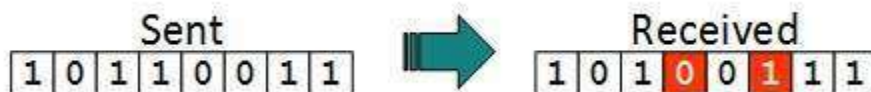
There may be three types of errors:

- **Single bit error**



In a frame, there is only one bit, anywhere though, which is corrupt.

- **Multiple bits error**



Frame is received with more than one bits in corrupted state.

- **Burst error**



Frame contains more than 1 consecutive bits corrupted.

Error control mechanism may involve two possible ways:

- Error detection
- Error correction

## Error Detection

Errors in the received frames are detected by means of Parity Check and Cyclic Redundancy Check (CRC). In both cases, few extra bits are sent along with actual data to confirm that bits received at other end are same as they were sent. If the counter-check at receiver' end fails, the bits are considered corrupted.

### Parity Check

One extra bit is sent along with the original bits to make number of 1s either even in case of even parity, or odd in case of odd parity.

The sender while creating a frame counts the number of 1s in it. For example, if even parity is used and number of 1s is even then one bit with value 0 is added. This way number of 1s remains even. If the number of 1s is odd, to make it even a bit with value 1 is added.

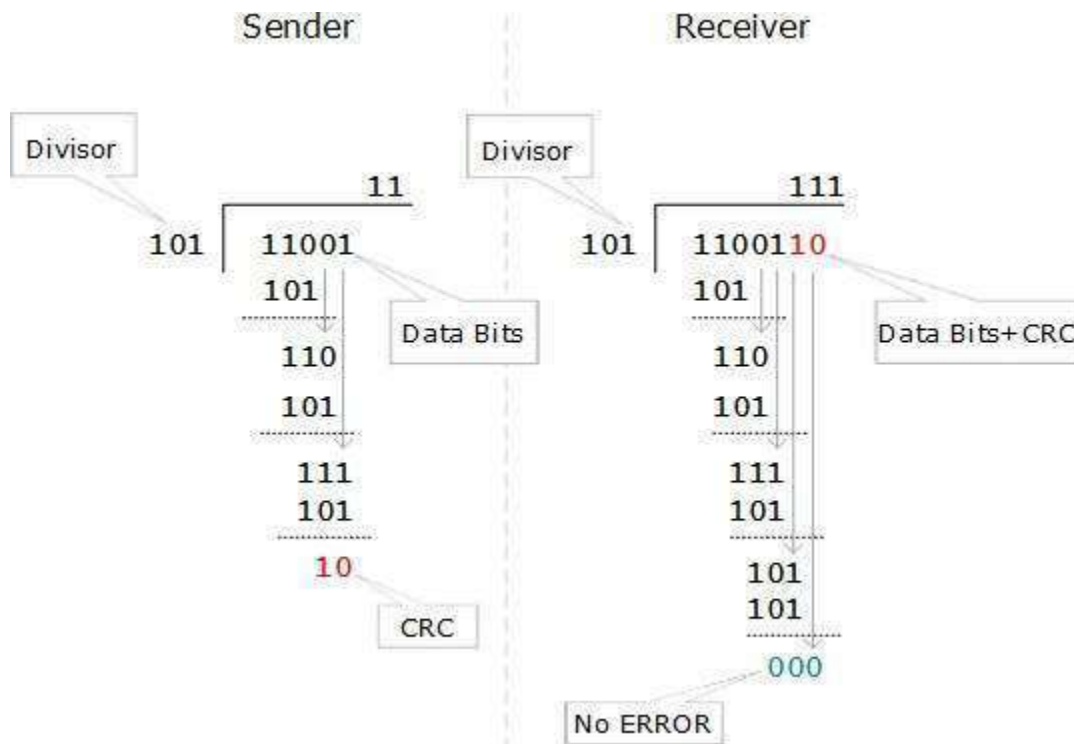


The receiver simply counts the number of 1s in a frame. If the count of 1s is even and even parity is used, the frame is considered to be not-corrupted and is accepted. If the count of 1s is odd and odd parity is used, the frame is still not corrupted.

If a single bit flips in transit, the receiver can detect it by counting the number of 1s. But when more than one bits are erroneous, then it is very hard for the receiver to detect the error.

## Cyclic Redundancy Check (CRC)

CRC is a different approach to detect if the received frame contains valid data. This technique involves binary division of the data bits being sent. The divisor is generated using polynomials. The sender performs a division operation on the bits being sent and calculates the remainder. Before sending the actual bits, the sender adds the remainder at the end of the actual bits. Actual data bits plus the remainder is called a codeword. The sender transmits data bits as codewords.



At the other end, the receiver performs division operation on codewords using the same CRC divisor. If the remainder contains all zeros the data bits are accepted, otherwise it is considered as there some data corruption occurred in transit.

## Error Correction

In the digital world, error correction can be done in two ways:

- **Backward Error Correction** When the receiver detects an error in the data received, it requests back the sender to retransmit the data unit.

- **Forward Error Correction** When the receiver detects some error in the data received, it executes error-correcting code, which helps it to auto-recover and to correct some kinds of errors.

The first one, Backward Error Correction, is simple and can only be efficiently used where retransmitting is not expensive. For example, fiber optics. But in case of wireless transmission retransmitting may cost too much. In the latter case, Forward Error Correction is used.

To correct the error in data frame, the receiver must know exactly which bit in the frame is corrupted. To locate the bit in error, redundant bits are used as parity bits for error detection. For example, we take ASCII words (7 bits data), then there could be 8 kind of information we need: first seven bits to tell us which bit is error and one more bit to tell that there is no error.

For  $m$  data bits,  $r$  redundant bits are used.  $r$  bits can provide  $2^r$  combinations of information. In  $m+r$  bit codeword, there is possibility that the  $r$  bits themselves may get corrupted. So the number of  $r$  bits used must inform about  $m+r$  bit locations plus no-error information, i.e.  $m+r+1$ .

$$2^r \geq m+r+1$$

# DIGITAL ELECTRONICS



## Section 1

### Digital Electronics General View

---



# 1. Digital Electronics General View

## Introduction to Digital Electronics

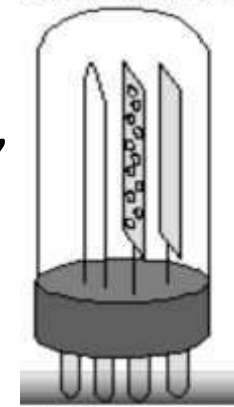
- ♣ Digital electronics is a branch of electronics which deals with digital format of data and codes.
- ♣ Digital stand for digit, digital electronics basically has two conditions which are possible, 0 (low logic) and 1 (high logic).
- ♣ Digital electronic systems use a digital signal that are composed of mathematical features to work.
- ♣ "1" as true and "0" as false are called bit and the group of bits are named byte.
- ♣ Digital electronic circuits are usually made from large assemblies of logic gates.
- ♣ Digital describes electronic technology that generates, stores, and processes data in terms of two states: 1 and number 0.
- ♣ A modem is used to convert the digital information in your computer to analog signals for your device and to convert analog signals to digital information for your computer.

# 1. Digital Electronics General View

## Digital Electronics Quick History

- ❖ Prior to digital technology, electronic transmission was limited to analog technology, which conveys data as electronic signals of varying frequency or amplitude.
- ❖ In the 1930's the prototypes of the computer were constructed from mechanical switches ( vacuum tubes ) and relays. These were comparatively slow, large, produced a great deal of heat.
- ❖ The next stage in the 1940's was the use of electronic diodes, and while these were better but they were unreliable.
- ❖ The next stage was the result of the development in 1947 of the transistor which was much smaller, faster and cooler. Simple transistors were replaced by integrated circuits and that got smaller and smaller and finally deposited silicon to be put into a "chip".

Vacuum Tube

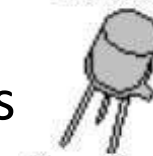


One Switch

Diode

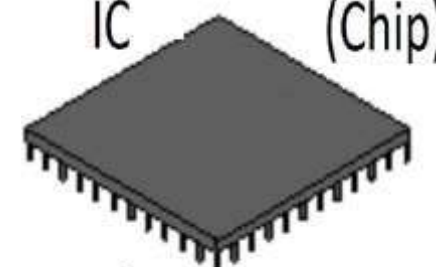


Transistor



One Switch

IC (Chip)



Millions of Transistor

# 1. Digital Electronics General View

## Analog Versus Digital

---

- ♣ Analog = Continuous waves
- ♣ Digital = Discrete waves



Example:

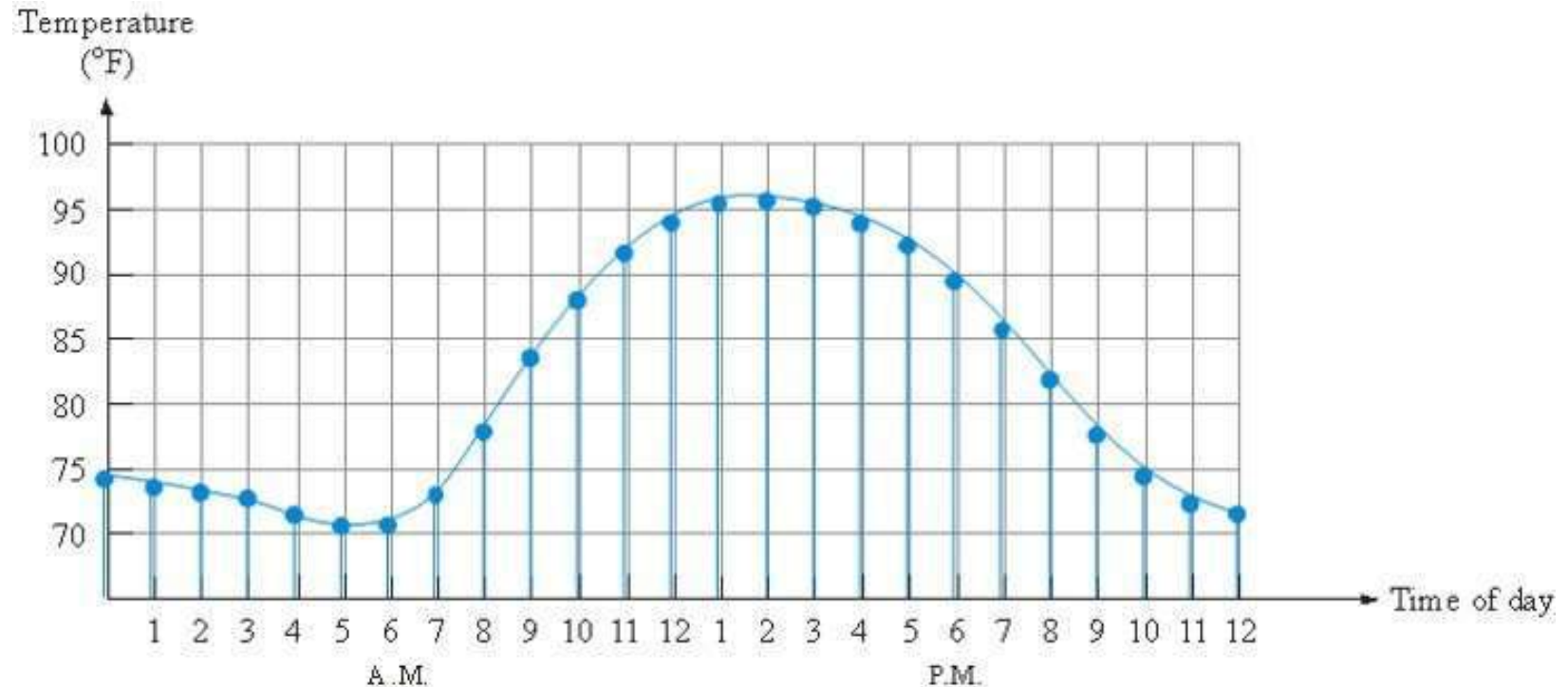
An analog clock, whose hands move smoothly and continuously.

A digital clock, whose digits jump from one value to the next.

# 1. Digital Electronics General View

## Analog Quantities

- ♣ Most natural quantities (such as temperature, pressure, light intensity, ...) are analog quantities that vary continuously.

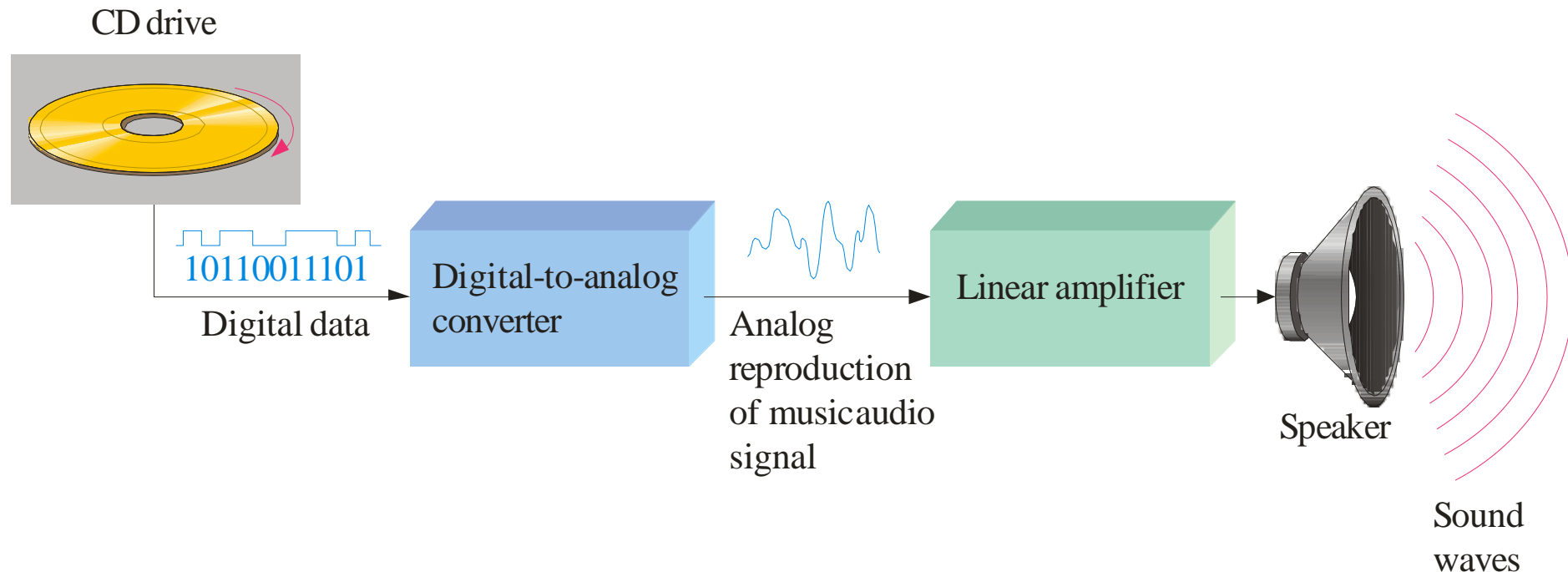


- ♣ Digital systems can process, store, and transmit data more efficiently but can only assign separate values to each point.

# 1. Digital Electronics General View

## Analog and Digital Systems

- ♣ Many systems use a maximum of analog and digital electronics to take advantage of each technology.
- ♣ Example: A typical CD player accepts digital data from the CD drive and converts it to an analog signal for amplification.



# 1. Digital Electronics General View

## Analogue to Digital Conversion

♣ Although some signals are originally digital. A continuous signal can be first converted to a proportional voltage waveform by a suitable transducer, the analogue signal is generated, and then for adapting digital processor, the signal has to be converted into digital form. The diagrams shows an analogue signal and its digital signal. The upper is the analogue signal and the lower is the digital signal.

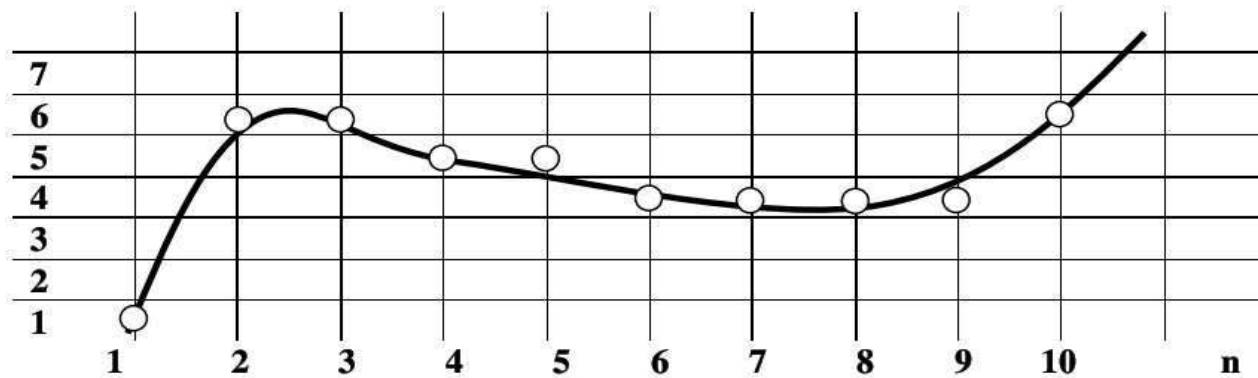
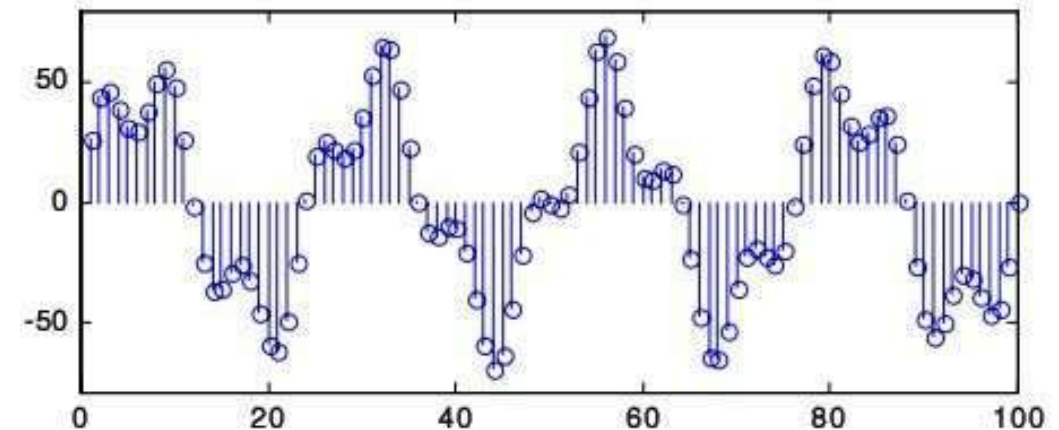
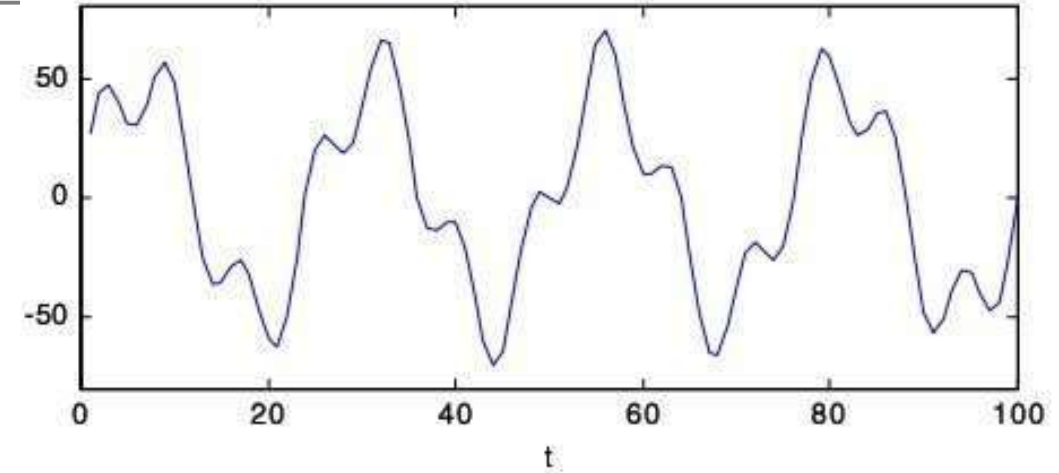


Figure 1.4 Continuous signal is sampled as 8 levels of digital signal.



# 1. Digital Electronics General View

## The Digital Revolution

♣ Recently, many types of devices have been converted from analog to digital.

Analog	Digital
Land Line Communication	Internet based Communication
Elementary Radio Navigation	Satellite Navigation
Analog Indicators	Digital Indicators

♣ In all of these digital devices, info is processed, transmitted and received as long strings of 1s and 0s.

# 1. Digital Electronics General View

## Advantages of Digital Electronics

---

- ♣ Computer-controlled digital systems can be controlled by software, allowing new functions to be added without changing hardware.
- ♣ Information storage can be easier in digital systems than in analog ones.
- ♣ The noise-immunity of digital systems permits data to be stored and retrieved without noise.
- ♣ In a digital system are easier to design and more precise representation of a signal can be obtained by using more binary digits to represent it.
- ♣ More digital circuitry can be fabricated on IC chips.
- ♣ Error management method can be inserted into the signal path. To detect errors, and then either correct the errors, or at least ask for a new copy of the data.

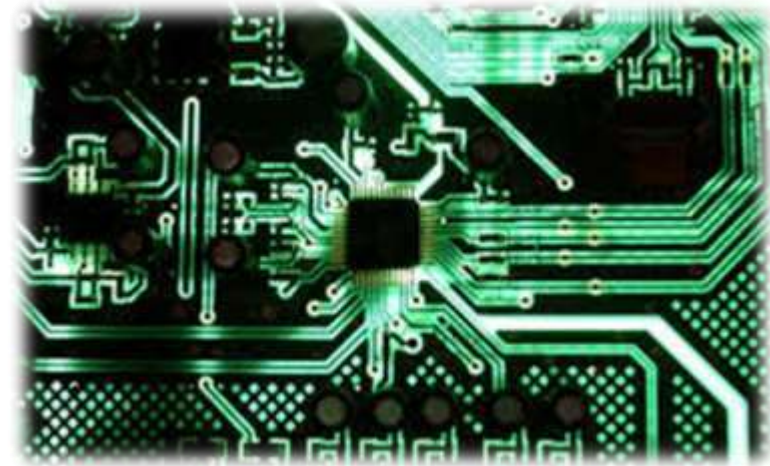


# 1. Digital Electronics General View

## Disadvantages of Digital Electronics

---

- ♣ Conversion to digital format and re-conversion to analog format is needed, which always include the lost of information.
- ♣ In some cases, digital circuits use more energy than analog circuits and produce more heat and need heat sinks.
- ♣ Digital circuits are sometimes more expensive, especially in small quantities.



# DIGITAL ELECTRONICS

## Section 2 Numbering System

---



# 2. Numbering System

## Types of Numbering Systems

---

♣ Many numbering systems are in use in digital technology. The most common are the:

Decimal            **537** <sub>10</sub>

Binary            **101001** <sub>2</sub>

Octal              **148** <sub>8</sub>

Hexadecimal    **4BAF** <sub>16</sub>



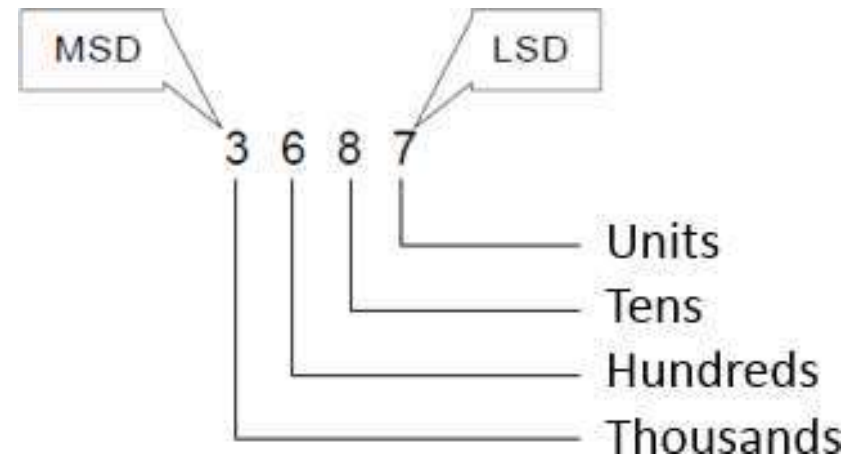
♣ To avoid confusion while using different numeral systems, the base of each individual number may be as specified by writing it as a subscript of the number.

# 2. Numbering System

## Decimal Numbering System

---

- ♣ Uses 10 symbols/digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
- ♣ Decimal Base (or Base 10)



MSD – Most Significant Digit  
LSD – Least Significant Digit

- ♣ Read as:

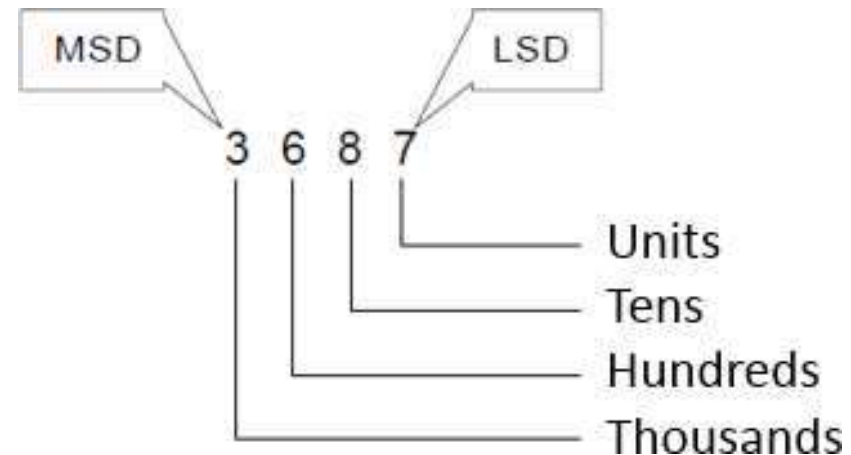
Three thousand six hundred eighty-seven, base ten

# 2. Numbering System

## Decimal Numbering System

---

- ♣ To decompose a decimal base number, we multiply each digit with his weight.



- ♣ Decomposition of the number:

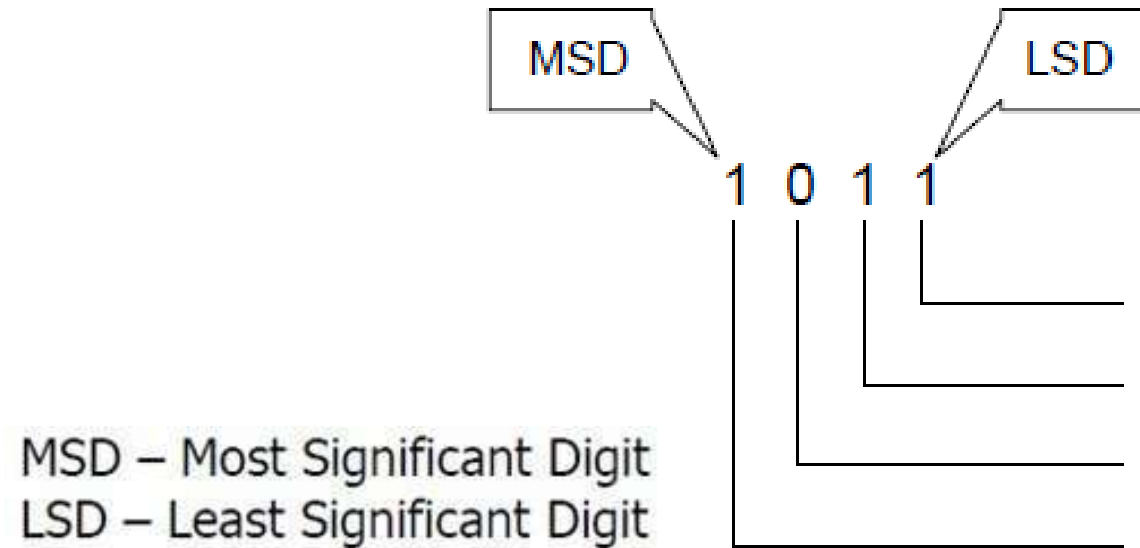
$$\mathbf{3687}_{(10)} = \mathbf{\underline{3} \times 10^3 + \underline{6} \times 10^2 + \underline{8} \times 10^1 + \underline{7} \times 10^0 = 3000 + 600 + 80 + 7}$$

# 2. Numbering System

## Binary Numbering System

---

- ♣ Uses 2 symbols/digits: 0, 1
- ♣ Binary Base (or Base 2)



- ♣ Read as:

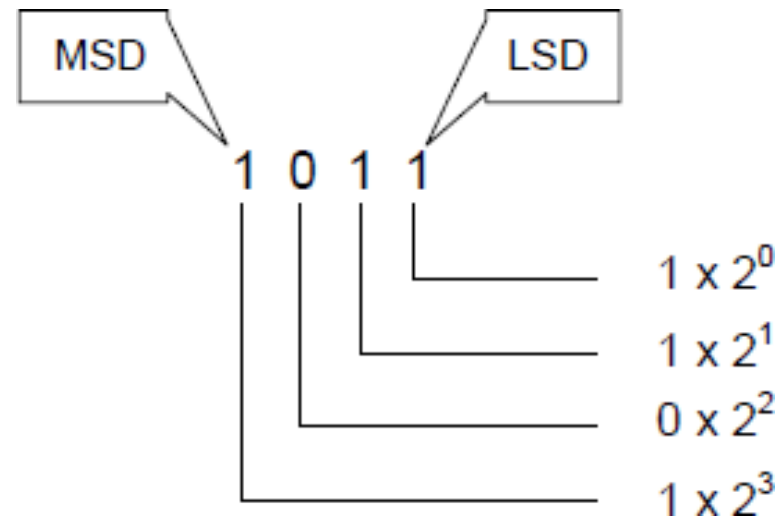
One Zero One One, base Two

# 2. Numbering System

## Binary Numbering System

---

♣ To decompose a binary base number, we multiply each digit with his weight.



♣ Decomposition of the number:

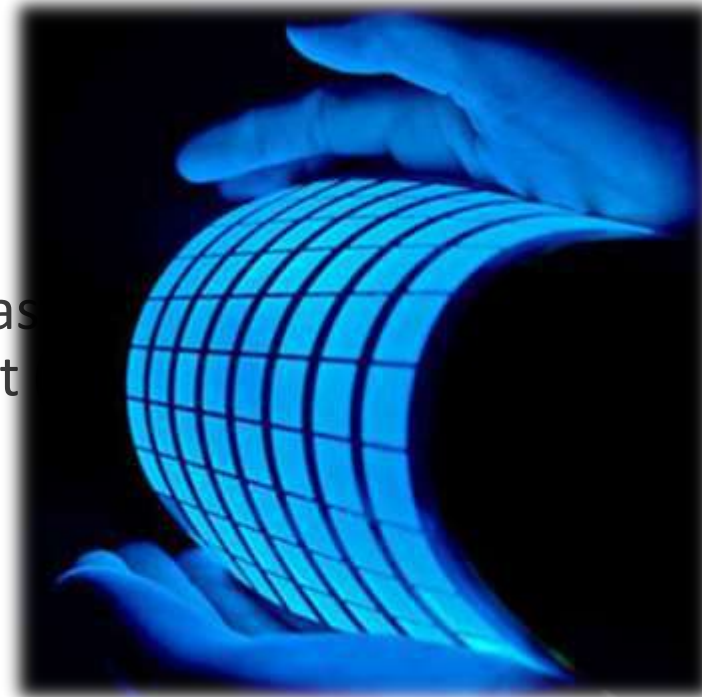
$$\text{Example: } 1011_{(2)} = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

# 2. Numbering System

## Converting Decimal to Binary

---

- ♣ To convert from a decimal integer numeral to binary, the number is divided by two and the remainder is the least-significant bit (LSD);
- ♣ The result is again divided by two, and its remainder is the next least significant bit;
- ♣ The process is repeated until the result cannot be divided anymore and the last result is the most-significant bit

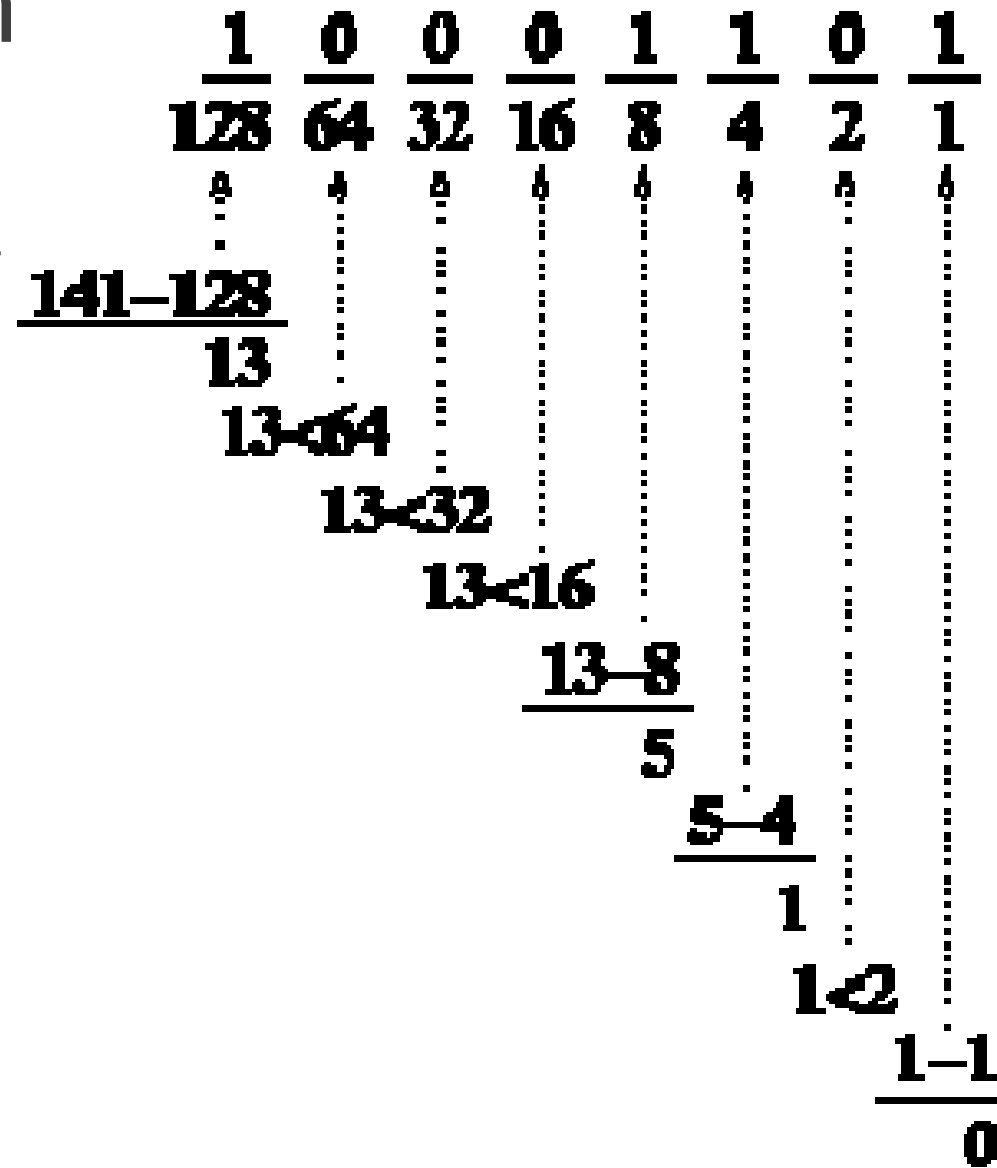






# 2. Numbering System

Converting Decimal to Binary





# 2. Numbering System

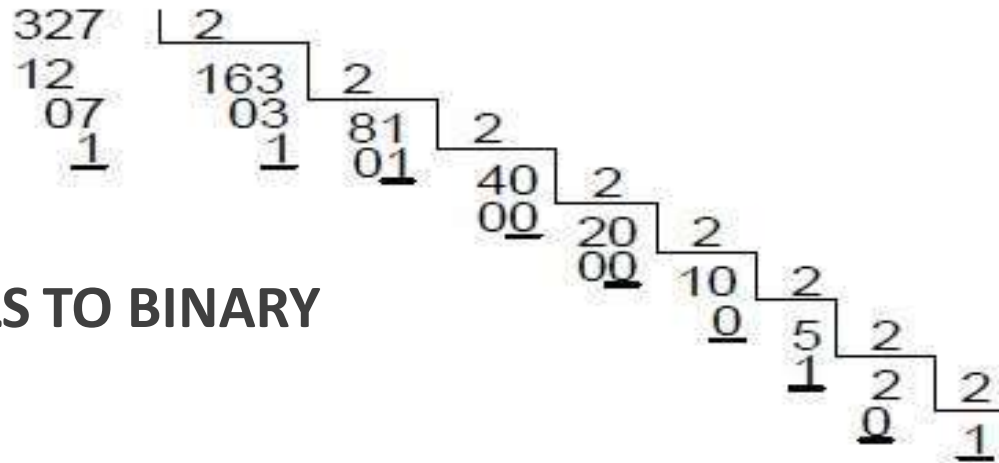
## CONVERTING FRACTIONARY DECIMALS TO BINARY

- ♣ If it is a fractionary decimal number the integer is converted using the previous method and the fractionary part is multiplied by 2;
- ♣ The integer part of that product is the MSD of the fractionary part of the number;
- ♣ If the fractionary part is not zero, it is once more multiplied by 2 and the integer part of that product is the next significant digit and so on until we reach zero for the fractionary part.



# 2. Numbering System

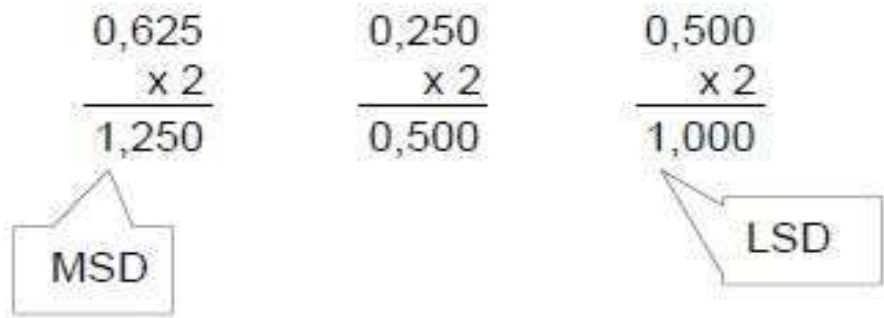
$$327,625_{(10)} \rightarrow (2)$$



## CONVERTING FRACTIONARY DECIMALS TO BINARY

$$327_{(10)} = 101000111_{(2)}$$

For the fractionary part:



$$0,625_{(10)} = 0,101_{(2)}$$

$$327,625_{(10)} \rightarrow 101000111,101_{(2)} (2)$$

# 2. Numbering System

## BINARY CODED DECIMAL BCD

♣ On the BCD (Binary Coded Decimal) system the digits are grouped in

4 bits nibbles, each nibble representing a decimal digit;

♣ The representation of the decimals 10, 11, 12, 13, 14 and 15 is excluded.

♣ The BCD is usually used in frequency counters, digital counters and calculators.

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Example:            5                    2                    9                    Decimal  
                         0101                    0010                    1001                    BCD

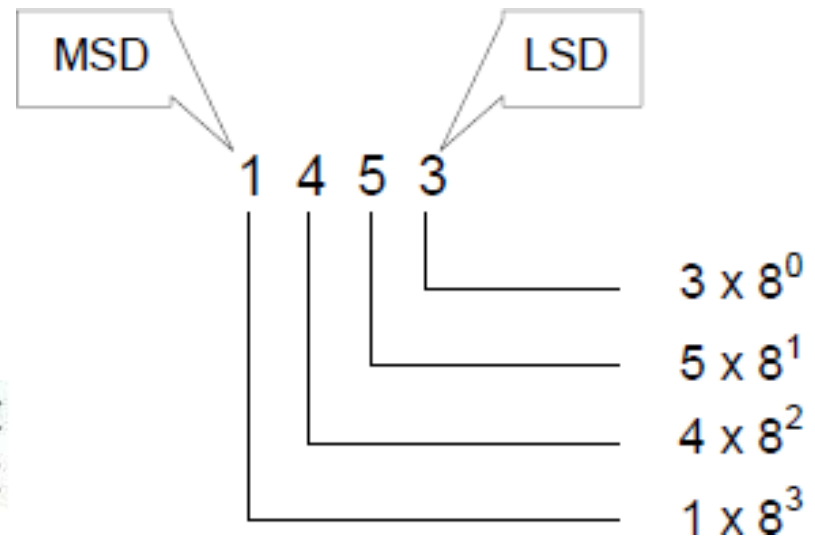
# 2. Numbering System

## OCTAL NUMBERING SYSTEM

---

- ♣ Uses 8 symbols/digits: 0, 1, 2, 3, 4, 5, 6, 7.
- ♣ OCTAL Base or (Base 8)

MSD – Most Significant Digit  
LSD – Least Significant Digit



- ♣ Read as:

One thousand four hundred fifty-three, base eight

# 2. Numbering System

## OCTAL Numbering System

---

♣ OCTAL numbering system makes it easier to represent binary numbers with many digits because an octal number represents a 3 digit binary number:

Example:

$$\begin{aligned} 111_{(2)} &= 7_{(8)} \\ 101_{(2)} &= 5_{(8)} \\ 100_{(2)} &= 4_{(8)} \end{aligned}$$

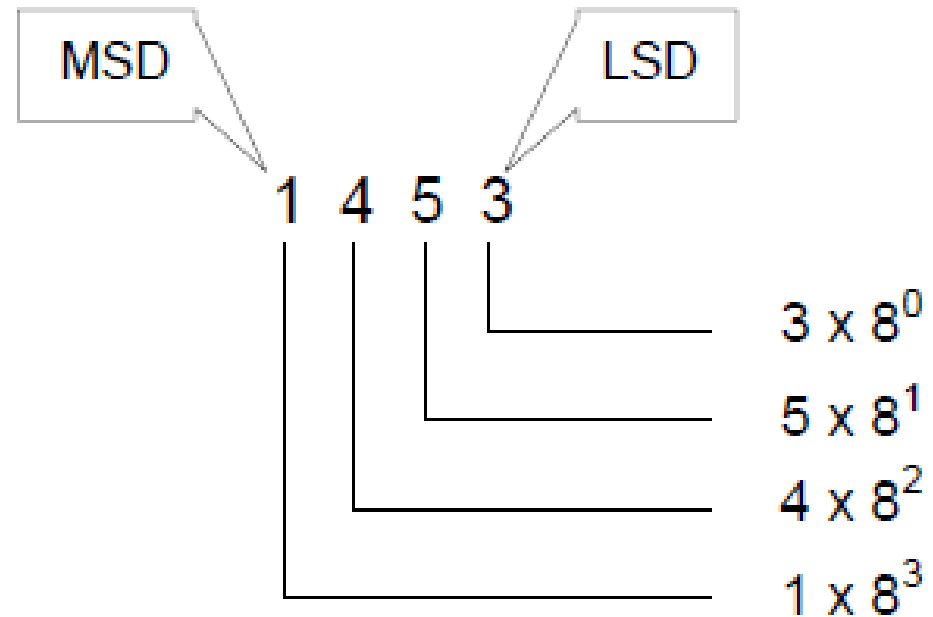


# 2. Numbering System

## OCTAL Numbering System

---

- ♣ To decompose an OCTAL number, we multiply each digit with his weight.



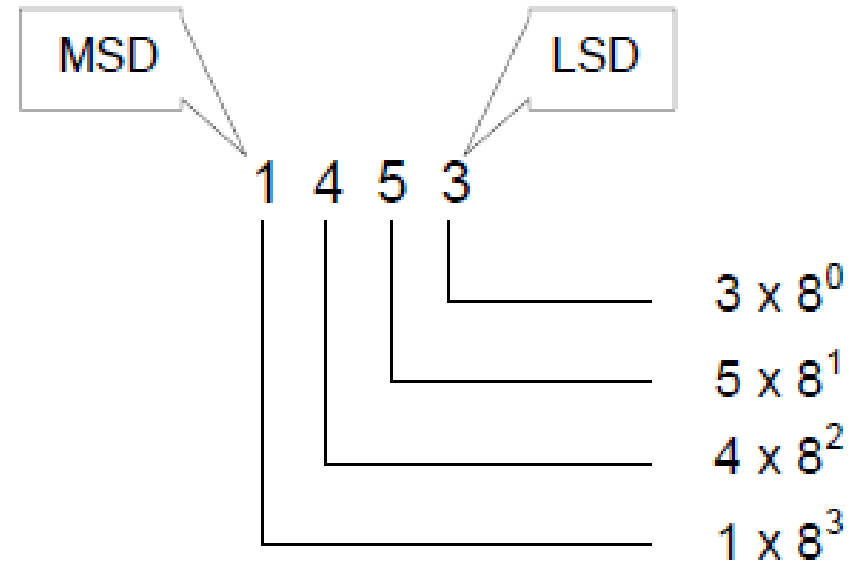
- ♣ Decomposition of the number:

$$\text{Example: } 1453_{(8)} = 1 \times 8^3 + 4 \times 8^2 + 5 \times 8^1 + 3 \times 8^0$$

# 2. Numbering System

## Converting OCTAL to Decimal

♣ To convert an OCTAL number to decimal, we multiply each digit with his weight and sum them.



$$\text{Example: } 1453_{(8)} = 1 \times 8^3 + 4 \times 8^2 + 5 \times 8^1 + 3 \times 8^0 = 512 + 256 + 40 + 3 = 811$$

## 2. Numbering System

### Decomposing a Fractionary OCTAL Number

---

Example:  $756,205_{(8)} = 7 \times 8^2 + 5 \times 8^1 + 6 \times 8^0 + 2 \times 8^{-1} + 0 \times 8^{-2} + 5 \times 8^{-3}$

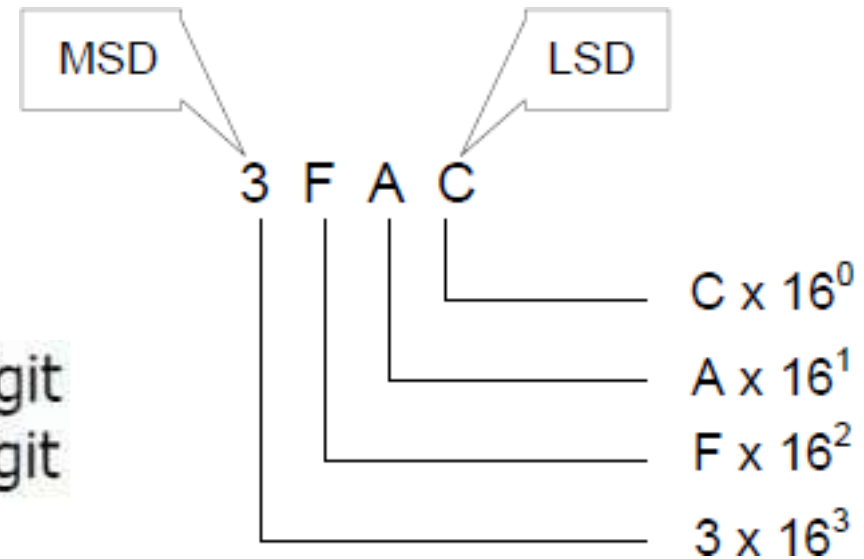
# 2. Numbering System

## Hexadecimal Numbering System

---

- ♣ Uses 16 symbols/digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- ♣ Hexadecimal base or (Base sixteen)

MSD – Most Significant Digit  
LSD – Least Significant Digit



- ♣ Read as:

Three F A C, base sixteen

# 2. Numbering System

## HEXADECIMAL Numbering System

---

- ♣ Makes it even easier to represent large binary numbers (an hexadecimal represents a 4 digit binary);

Example:

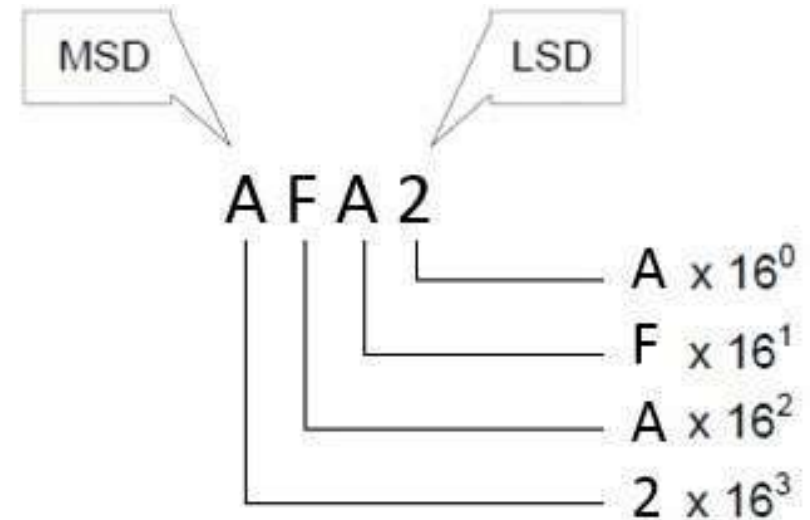
$$1111_{(2)} = F_{(16)}$$
$$1010_{(2)} = A_{(16)}$$
$$1001_{(2)} = 9_{(16)}$$

# 2. Numbering System

## Hexadecimal Numbering System

---

♣ To decompose a hexadecimal number, we multiply each digit with his weight.



♣ Decomposition of the number:

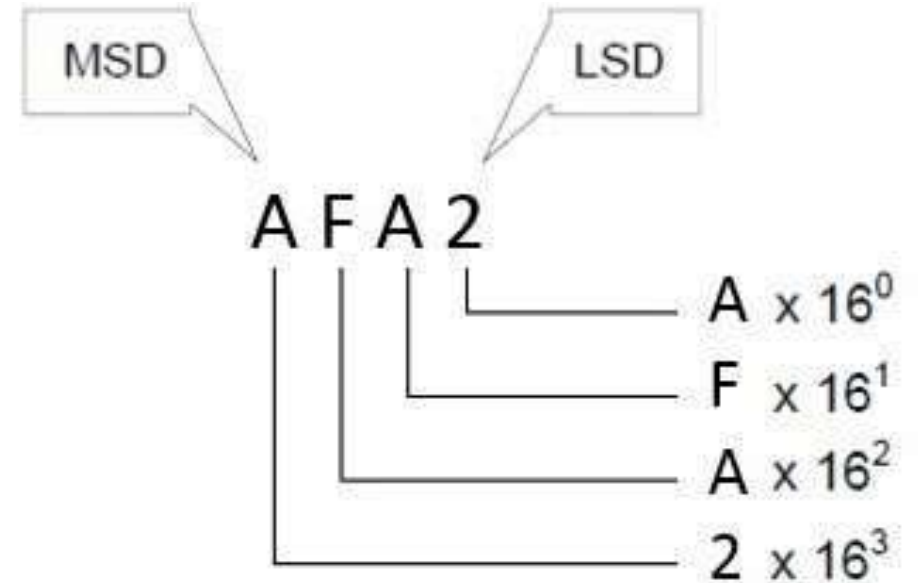
$$\text{Example: } AFA2_{(16)} = A \times 16^3 + F \times 16^2 + A \times 16^1 + 2 \times 16^0$$

# 2. Numbering System

## Converting Hexadecimal to Decimal

---

- ♣ To convert an Hexadecimal number to decimal, we multiply each digit with his weight and sum them.



Example:  $AFA2_{(16)} = A \times 16^3 + F \times 16^2 + A \times 16^1 + 2 \times 16^0 = A \ 4096 + F \ 256 + A \ 16 + 2$

## 2. Numbering System

### Decomposing a Fractionary Hexadecimal Number

---

Example:  $F16,FAC_{(16)} = F \times 16^2 + 1 \times 16^1 + 6 \times 16^0 + F \times 16^{-1} + A \times 16^{-2} + C \times 16^{-3}$



# DIGITAL ELECTRONICS

## Section 3

### Binary System Operations

---



# 3. Binary System Operations

---

♣ Sum

♣ Multiplication

♣ Subtraction

♣ Division

# 3. Binary System Operations

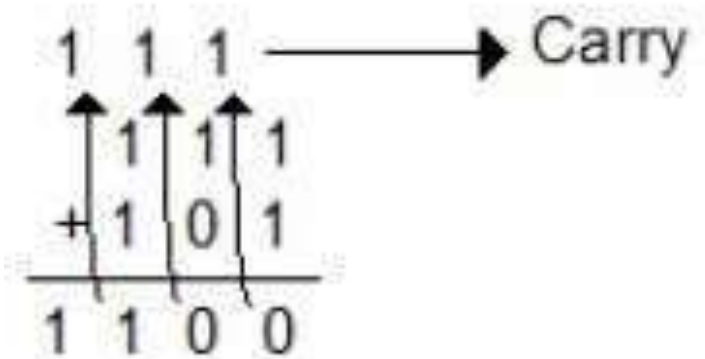
## SUM (X+Y)

---

X + Y	Result
0 + 0	0 carry 0
0 + 1	1 carry 0
1 + 0	1 carry 0
1 + 1	0 carry 1

Example:

$$7_{(10)} + 5_{(10)} = 12_{(10)}$$
$$7_{(10)} = 111_{(2)}$$
$$5_{(10)} = 101_{(2)}$$



Checking:

$$1100_{(2)} = 1 \times 2^3 + 1 \times 2^2 + 0 + 0 = 8 + 4 = 12_{(10)}$$

# 3. Binary System Operations

## MULTIPLICATIO (X·Y)

---

<b>X . Y</b>	<b>Result</b>
0 x 0	0
0 x 1	0
1 x 0	0
1 x 1	1

Example:

$$6_{(10)} \times 5_{(10)} = 30_{(10)}$$

$$110_{(2)} \times 101_{(2)} = 11110_{(2)}$$

$$110_{(2)} = 6_{(10)}$$

$$101_{(2)} = 5_{(10)}$$

$$\begin{array}{r} 110 \\ \times 101 \\ \hline 110 \\ 000 \\ +110 \\ \hline 11110 \end{array}$$

Example:

$$11110_{(2)} = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 16 + 8 + 4 + 2 = 30_{(10)}$$

# 3. Binary System Operations

## SUBTRACTION (X-Y)

---

<b>X - Y</b>	<b>Result</b>
0 - 0	0 carry 0
0 - 1	1 "borrow" 1
1 - 0	1 carry 0
1 - 1	0 carry 0

Example:

$$20_{(10)} - 7_{(10)} = 13_{(10)}$$

$$10100_{(2)} = 16 + 4 = 20_{(10)}$$

$$111_{(2)} = 4 + 2 + 1 = 7_{(10)}$$

$$\begin{array}{r} 10100 \\ + 111 \\ \hline 1111 \longrightarrow \text{Borrow} \\ \hline | 101 \end{array}$$

Check:

$$10100_{(2)} - 111_{(2)} = 01101_{(2)}$$

$$1101_{(2)} = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 4 + 0 + 1 = 13_{(10)}$$

# 3. Binary System Operations

## DIVISION (X/Y)

♣ No table, it is done by using the mathematical rules of multiplication and subtraction.

Example:

$$11000_{(2)} \div 111_{(2)}$$

$$\begin{array}{r} 11000 \quad | \quad 111 \\ -111 \quad \downarrow \\ \hline 01010 \\ -111 \\ \hline 00011 \end{array}$$

$$\left\{ \begin{array}{l} Q = 11_{(2)} \\ R = 11_{(2)} \end{array} \right.$$

Check:

$$11000_{(2)} = 1 \times 2^4 + 1 \times 2^3 + 0 + 0 + 0 = 16 + 8 = 24_{(10)}$$

$$111_{(2)} = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 4 + 2 + 1 = 7_{(10)}$$

$$\begin{array}{r} 24 \quad | \quad 7 \\ 3 \quad \quad 3 \end{array}$$

$$\left\{ \begin{array}{l} Q = 3_{(10)} \\ R = 3_{(10)} \end{array} \right.$$

# DIGITAL ELECTRONICS

## Section 4 Boolean Algebra

---



# 4. Boolean Algebra

## Introduction to Boolean Algebra

- ♣ Much of what we will be discussing was formalized by George Boole (1815–1864) in his paper An Investigation of the Laws of Thought.
- ♣ The branch of mathematics involving digital logic is aptly named Boolean Algebra.
- ♣ Developed to investigate the fundamental laws of the human mind 's operations relating to reasoning.
- ♣ Its purpose is to define symbols to represent phenomenon's that will originate more complex mathematical expressions – Boolean Functions or Boolean Expressions;
- ♣ But unlike traditional Algebra, instead of dealing with quantities, Boolean Algebra deals with Logic.
- ♣ Symbols like (+) or (x) represent logic relations between signals and pules, instead of sum and multiplication algorithms.
- ♣ The symbols 0 and 1 represent Logic States and not quantities (0 is Low or False and 1 is High or True).



# 4. Boolean Algebra

## Sum of Two States

---

$$0+0=0$$

$$0+1=1$$

$$1+0=1$$

$$1+1=1$$

$$1+1=1$$

SUM of two states:

- There are only two possibilities: true or false;
- A square wave with High value adding to another results in a square wave with High value.

$$0 + 1 + 1 = 1$$

$$1 + 1 + 1 = 1$$

$$0 + 1 + 1 + 1 = 1$$

$$1 + 0 + 1 + 1 + 1 = 1$$

It does not matter how many or few terms we add together, either.

# 4. Boolean Algebra

## Product of Two States

---

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

### Multiplication of two states:

- There are only two possibilities: true or false;
- A square wave with High value multiplying to another results in a square wave with High value.

$$1 \times 1 \times 1 \times 1 = 1$$

$$1 \times 1 \times 0 \times 1 = 0$$

$$0 \times 1 \times 0 \times 1 = 0$$

It does not matter how many or few terms we multiply together.

# DIGITAL ELECTRONICS

## Section 5 Basic Logic Operation

---



# 5. Basic Logic Operation

## What is Logic Gate?

---

- ♣ Digital gate is a Digital Device used to perform the logic operation
- ♣ Logic gates (or simply gates) are the fundamental building blocks of digital circuitry
- ♣ Electronic gates require a power supply.  
Gate INPUTS are driven by voltages having two nominal values, e.g. 0V and 5V representing logic 0 and logic 1 respectively.  
  
The OUTPUT of a gate provides two nominal values of voltage only, e.g. 0V and 5V representing logic 0 and logic 1 respectively.
- ♣ In general, there is only one output to a logic gate except in some special cases.

# 5. Basic Logic Operation

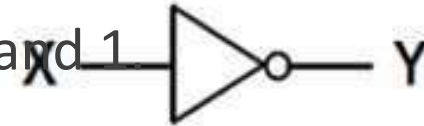
## Truth Table

---

- ♣ Truth tables are used to help show the function of a logic gate.
- ♣ Relation between outputs and inputs.
- ♣ Number of entries on the table is  $2^n$  with  $n$  being the number of inputs and base 2 because it is related to the binary numbering 0 and 1

NOT Gate

---



Truth Table:

	X	Y
	0	1
One input -> $2^1=2$ entries	1	0

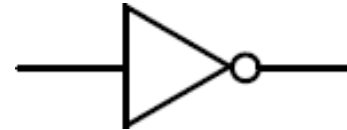
# 5. Basic Logic Operation

## Types of Logic Gate

---

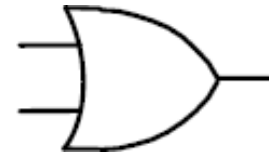
### ♣ NOT Gate

- Output is the invert of the input;
- Always one input.



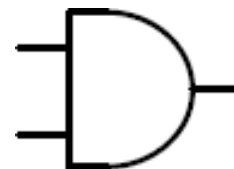
### ♣ OR Gate

- Output is the binary sum of the inputs.



### ♣ AND Gate

- Output is the binary multiplication of the inputs.

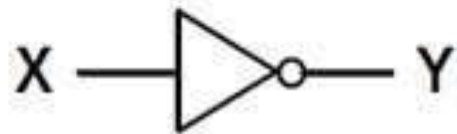


# 5. Basic Logic Operation

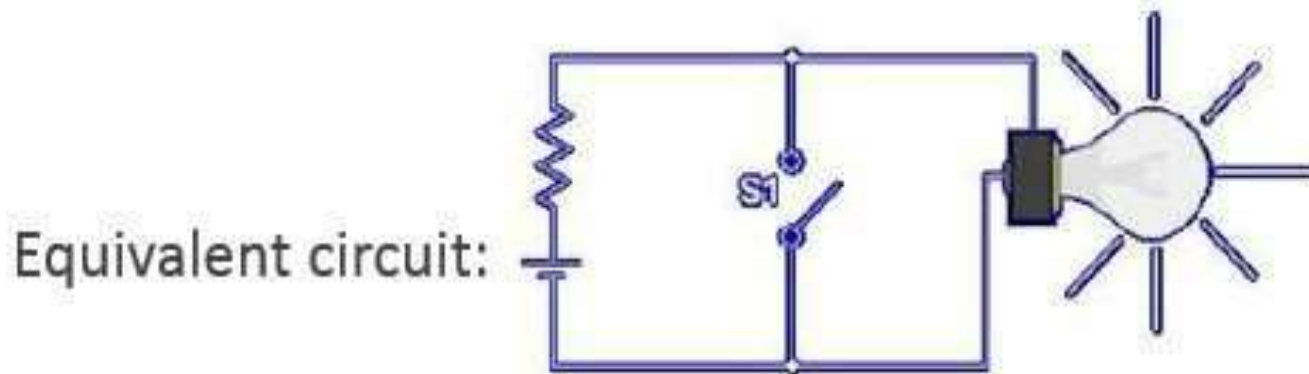
## NOT Gate

---

♣ The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is X, the inverted output is known as NOT X. This is also shown as X', or X with a bar over the top, as shown at the outputs.



Boolean Equation:  $Y = \overline{X}$



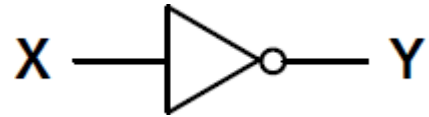
Truth Table:  
One input  $\rightarrow 2^1=2$  entries

X	Y
0	1
1	0

# 5. Basic Logic Operation

## NOT Gate

---



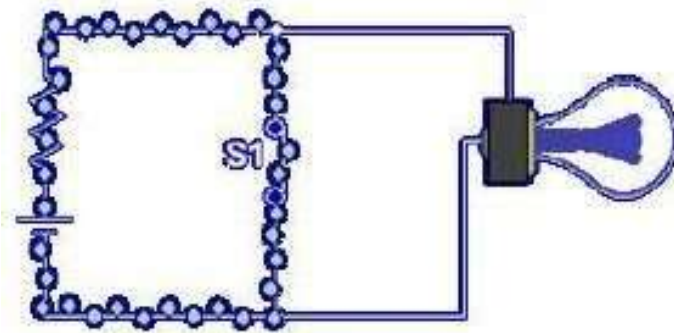
Boolean Equation:  $Y = \overline{X}$

Truth Table:

X	Y
0	1
1	0

One input  $\rightarrow 2^1=2$  entries

Equivalent circuit:





# 5. Basic Logic Operation

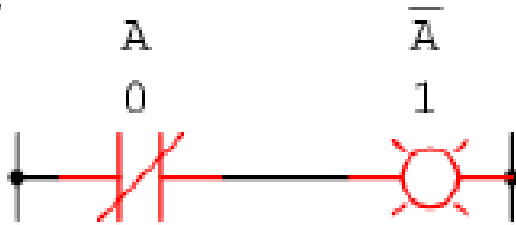
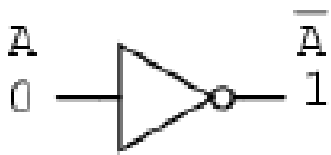
## NOT Gate

---

- Boolean variables are 1 or 0, and each one has its complement or inverse.

**If:  $A=0$**

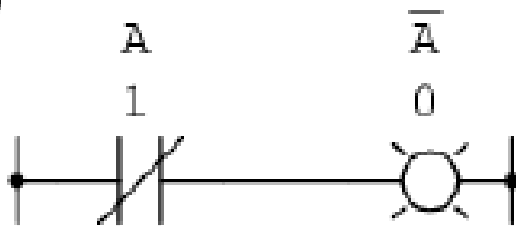
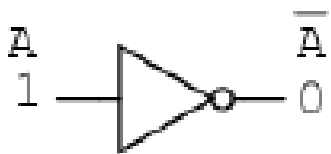
**Then:  $\bar{A}=1$**



NOT gate.

**If:  $A=1$**

**Then:  $\bar{A}=0$**



The Boolean Expression is:

$$A = \bar{\bar{A}}$$

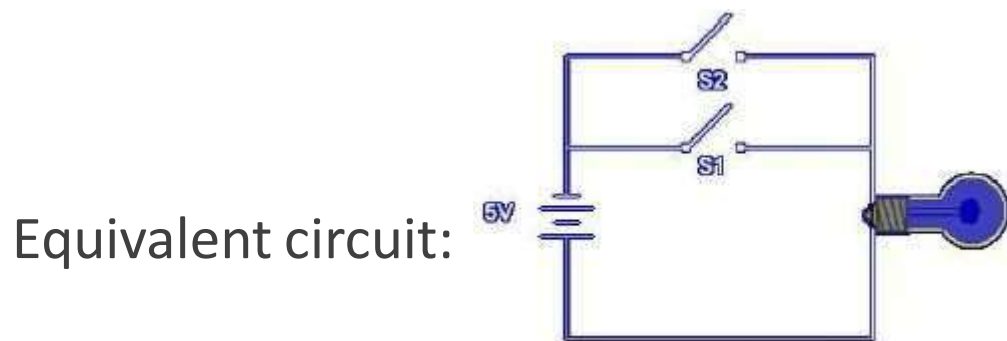
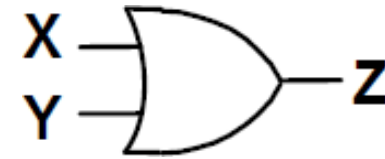
# 5. Basic Logic Operation

## OR Gate

---

♣ The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high. A plus (+) is used to show the OR operation.

$$X + Y = Z$$



Truth Table:

Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

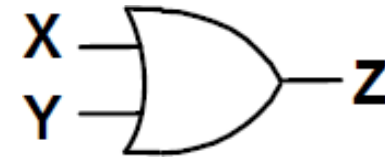
# 5. Basic Logic Operation

## OR Gate

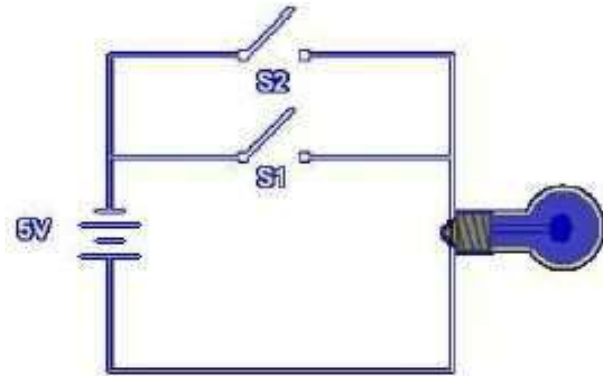
---

♣ The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high. A plus (+) is used to show the OR operation.

$$X + Y = Z$$



Equivalent circuit:



Truth Table:

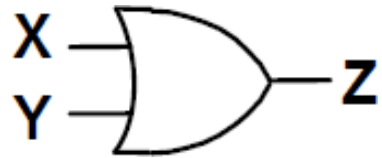
Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

# 5. Basic Logic Operation

## OR Gate

---



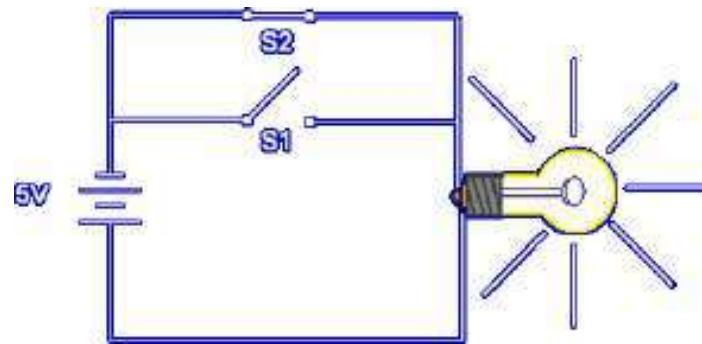
Truth Table:

Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

$$X + Y = Z$$

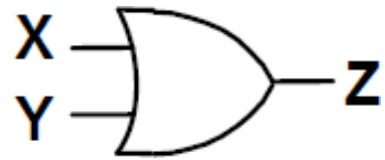
Equivalent circuit:



# 5. Basic Logic Operation

## OR Gate

---



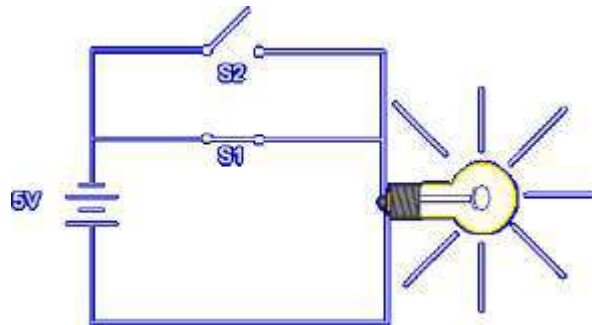
$$X + Y = Z$$

Truth Table:

Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Equivalent circuit:



# 5. Basic Logic Operation

## OR Gate

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

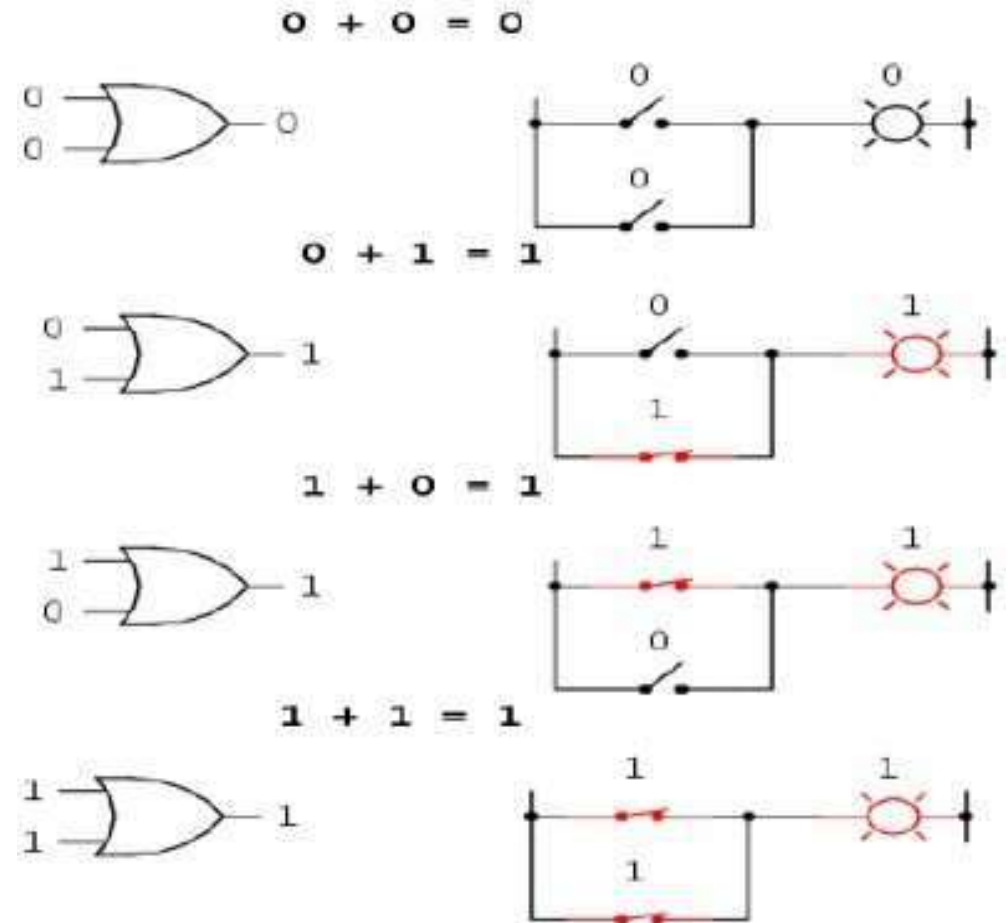


It represents the Truth Table for an OR gate!

For the OR gate the Boolean Expression is:

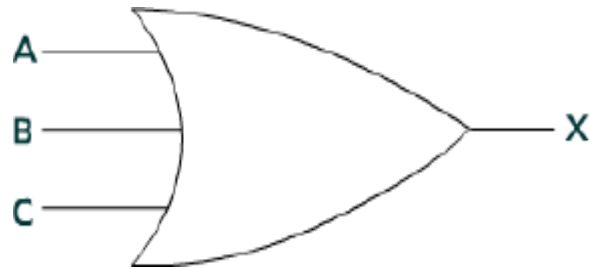
$$A + B = Z$$

A and B are the inputs, Z the output.



# 5. Basic Logic Operation

## OR Gate



$$C + B + A = X$$

Three inputs  $\rightarrow 2^3=8$  entries

Inputs			Output
<i>C</i>	<i>B</i>	<i>A</i>	<i>X</i>
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

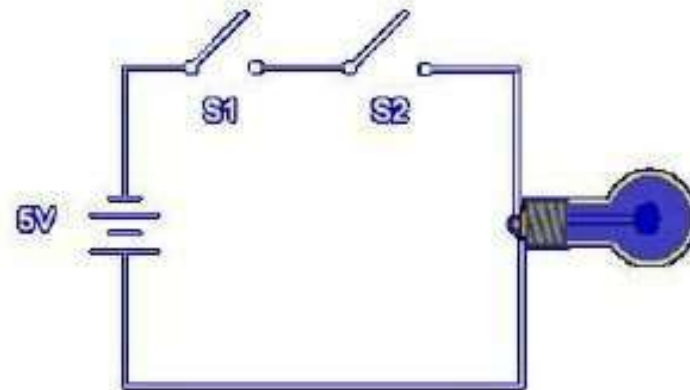
# 5. Basic Logic Operation

## AND Gate

♣ The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation (X.Y), Bear in mind that this dot is sometimes omitted (XY).



Equivalent Circuit:



Truth Table:

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

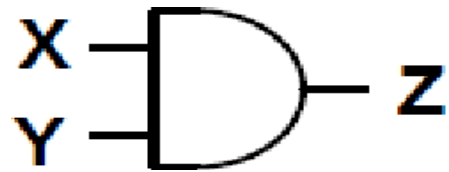
Two inputs  $\rightarrow 2^2=4$  entries

$$X \cdot Y = Z$$



# 5. Basic Logic Operation

## AND Gate



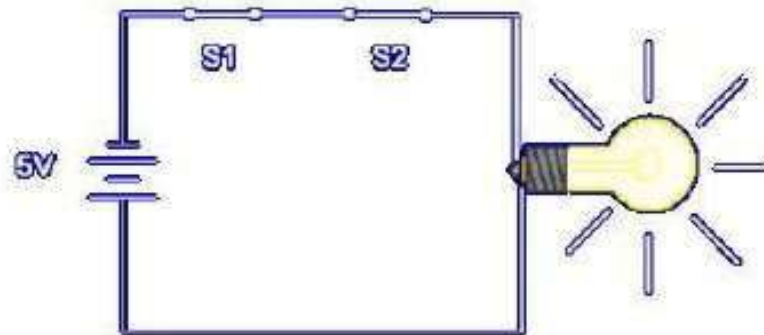
$$X \cdot Y = Z$$

Truth Table:

Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Equivalent Circuit:



# 5. Basic Logic Operation

## AND Gate

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

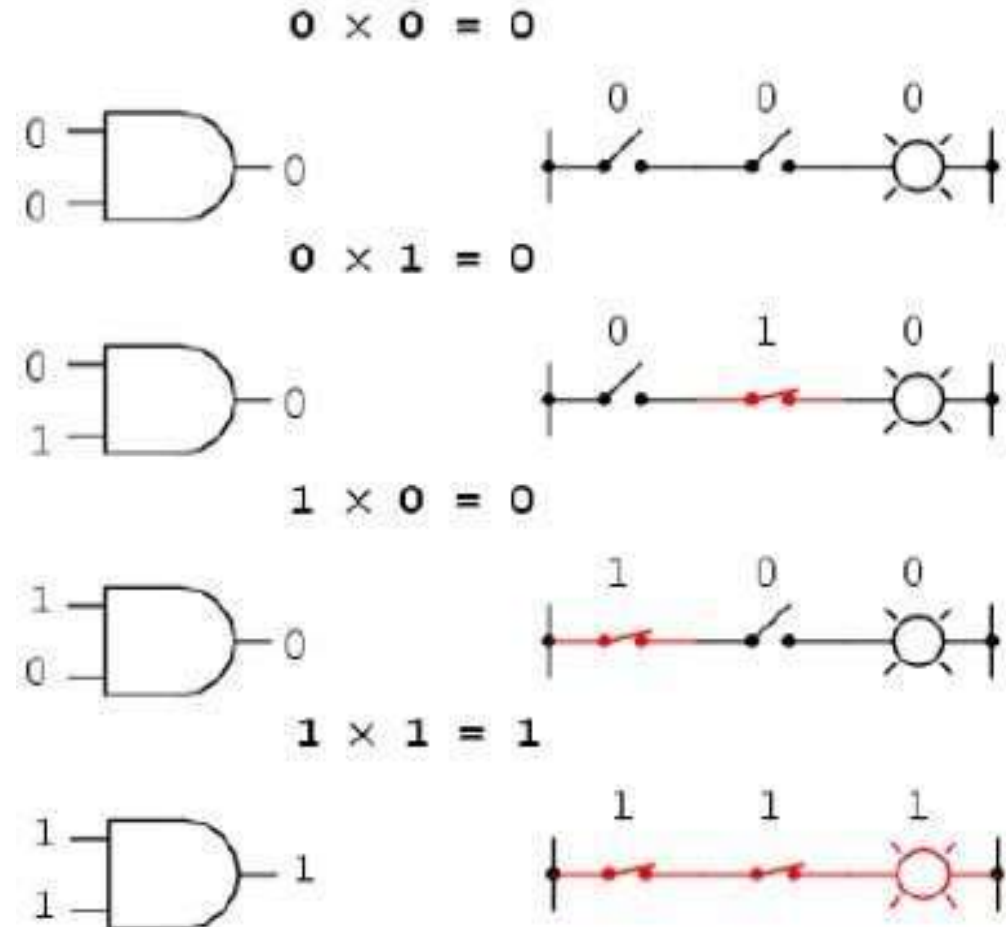


AND gate!

For the AND gate the Boolean Expression is:

$$A \cdot B = Z$$

A and B are the inputs, Z the output.



# 5. Basic Logic Operation

## AND Gate

---



$$C \cdot B \cdot A = X$$

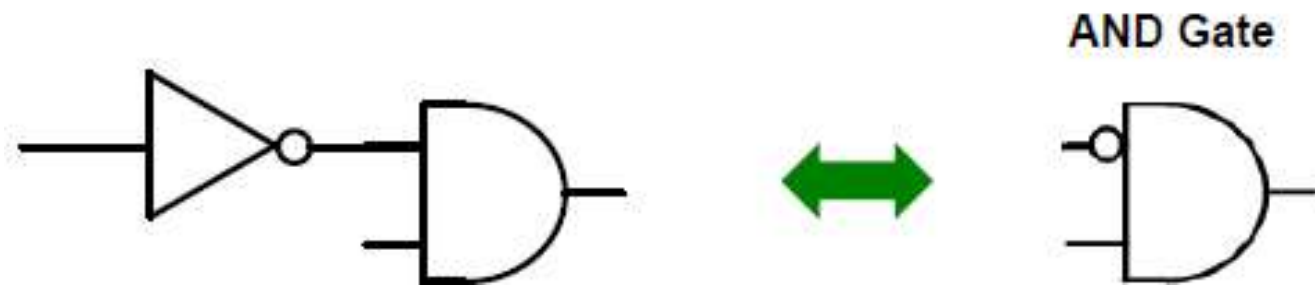
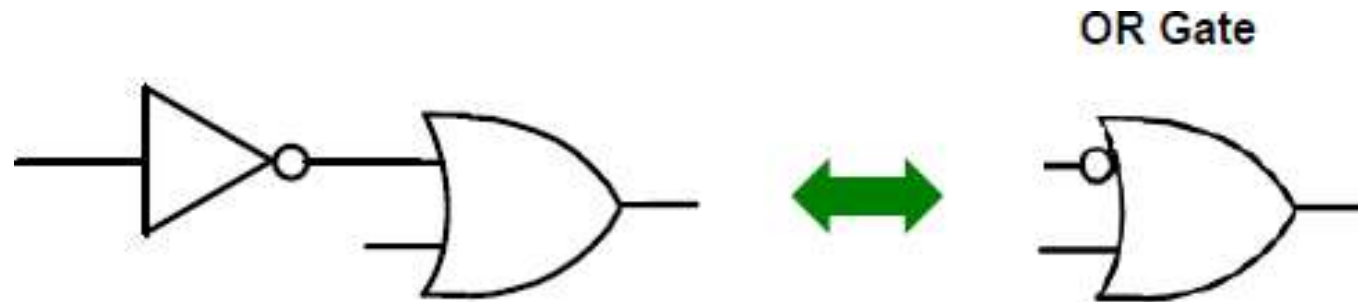
Three inputs  $\rightarrow 2^3=8$  entries

Inputs			Output
<i>C</i>	<i>B</i>	<i>A</i>	<i>X</i>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

# 5. Basic Logic Operation

## Inverted Inputs

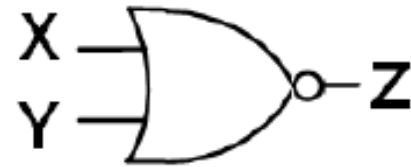
---



# 5. Basic Logic Operation

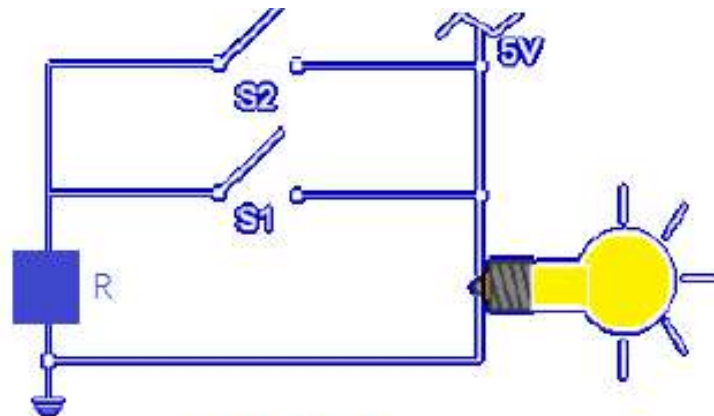
## NOR Gate

❖ This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high. The symbol is an OR gate with a small circle on the output. The small circle represents inversion.



$$\overline{X + Y} = Z$$

Equivalent Circuit:



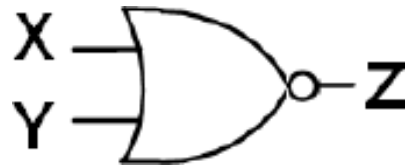
Truth Table:

Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

# 5. Basic Logic Operation

## NOR Gate

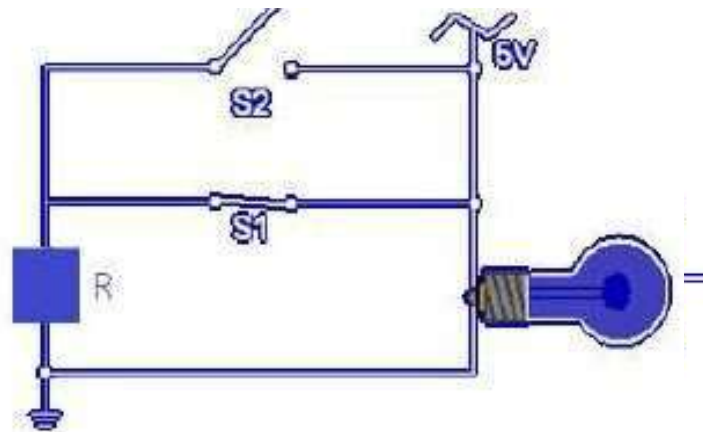


Truth Table:

Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Equivalent Circuit:



$$\overline{X + Y} = Z$$

# 5. Basic Logic Operation

## NOR Gate

---



$$\overline{A + B + C} = X$$

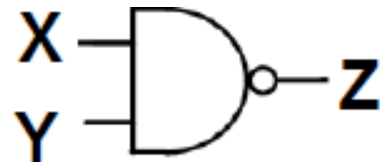
Three inputs  $\rightarrow 2^3=8$  entries

Inputs			Output
<i>C</i>	<i>B</i>	<i>A</i>	<i>X</i>
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

# 5. Basic Logic Operation

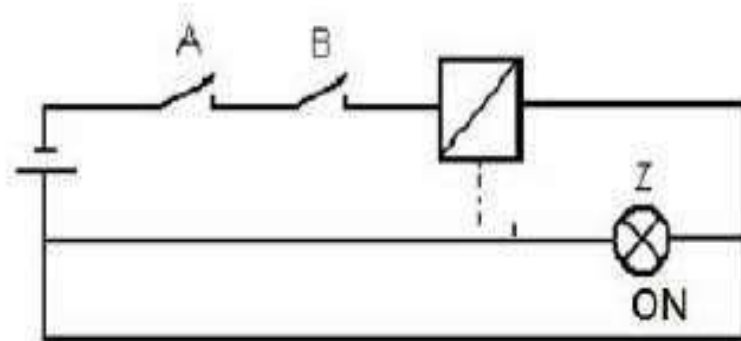
## NAND Gate

♣ This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if any of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.



$$\overline{X \cdot Y} = Z$$

Equivalent Circuit:



Truth Table:

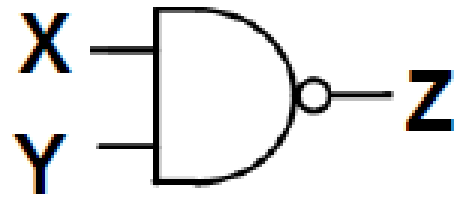
Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



# 5. Basic Logic Operation

## NAND Gate

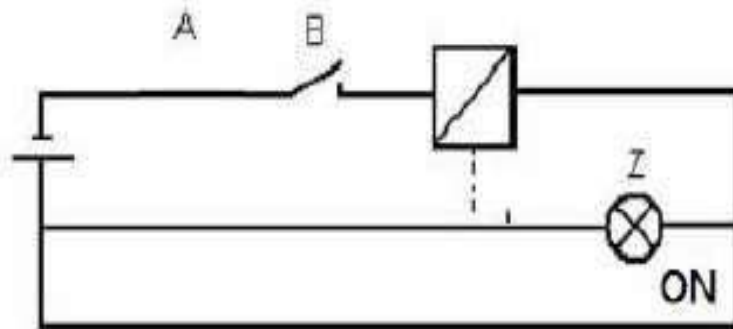


Truth Table:

Two inputs  $\rightarrow 2^2=4$  entries

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

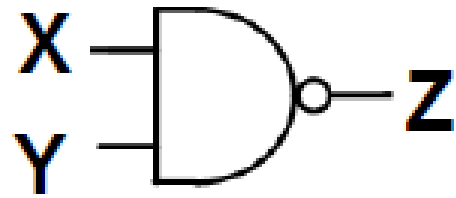
Equivalent Circuit:



$$\overline{X \cdot Y} = Z$$

# 5. Basic Logic Operation

## NAND Gate

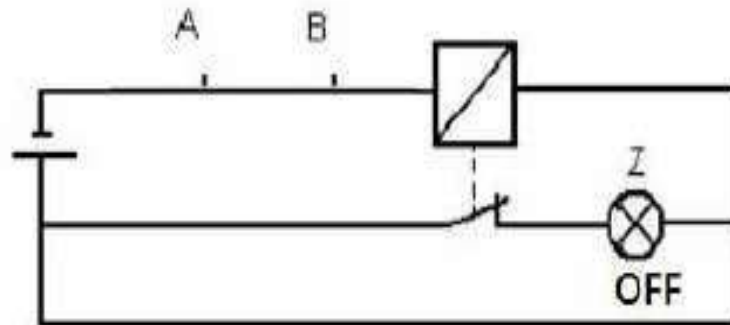


Truth Table:

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

Two inputs  $\rightarrow 2^2=4$  entries

Equivalent Circuit:

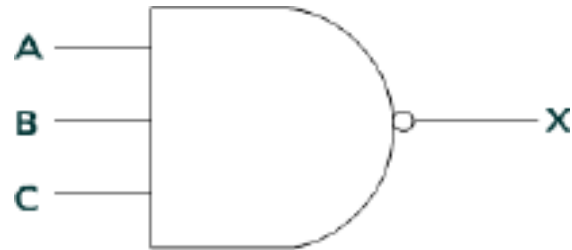


$$\overline{X \cdot Y} = Z$$

# 5. Basic Logic Operation

## NAND Gate

---



$$\overline{A \cdot B \cdot C} = X$$

Three inputs  $\rightarrow 2^3=8$  entries

Inputs			Output
<i>C</i>	<i>B</i>	<i>A</i>	<i>X</i>
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	0

# 5. Basic Logic Operation

## Complex Operations

---

♣ Up until now we remembered the following logic gates:

NOT, OR, AND, NOR, NAND

♣ There are two more gates available to do more complex operations:

- XOR (Exclusive-OR), and its output is HIGH only if the number of HIGH inputs is ODD.
- XNOR (Exclusive-NOR), and its output is HIGH only if the number of HIGH inputs is EVEN.

# 5. Basic Logic Operation

## Complex Operations

---

### ♣ XOR Gate

The 'Exclusive-OR' gate is a circuit which will give a high output if its two inputs are different.

An encircled plus sign ( $\oplus$ ) is used to show the EOR operation.



$$X \oplus Y = Z$$

Truth Table:

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

# 5. Basic Logic Operation

## Complex Operations

### ♣ XOR Gate

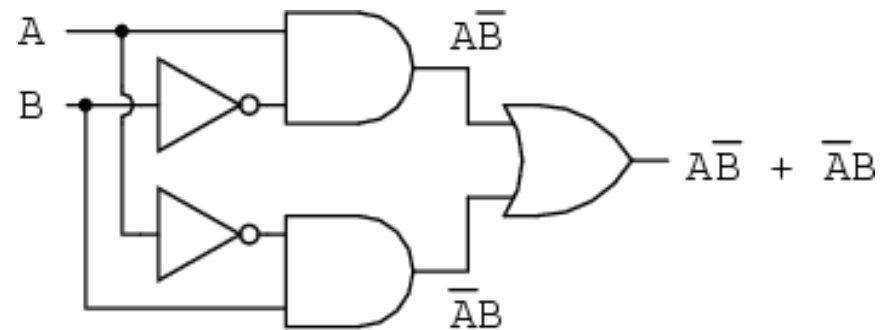


Boolean Equation:  $Z = A \oplus B$

XOR works as a Controlled Inverter, because if  $Y = 0$ ,  $Z = X$  and if  $Y = 1$ ,  $Z = \bar{X}$ .

Truth Table:

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0



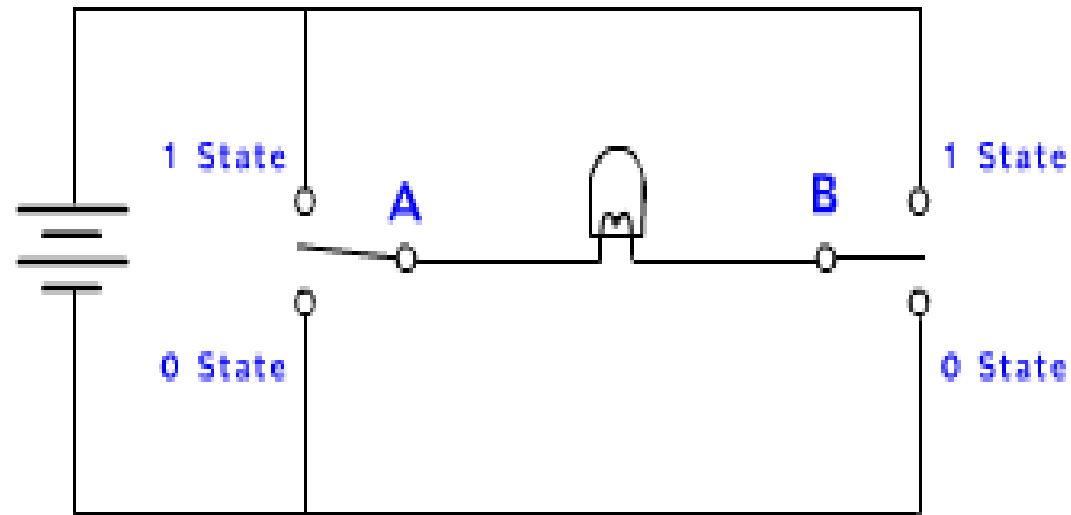
$$A \oplus B = A\bar{B} + \bar{A}B$$

# 5. Basic Logic Operation

## Complex Operations

---

♣ XOR Gate Equivalent Circuit:



Off = 0

On = 1

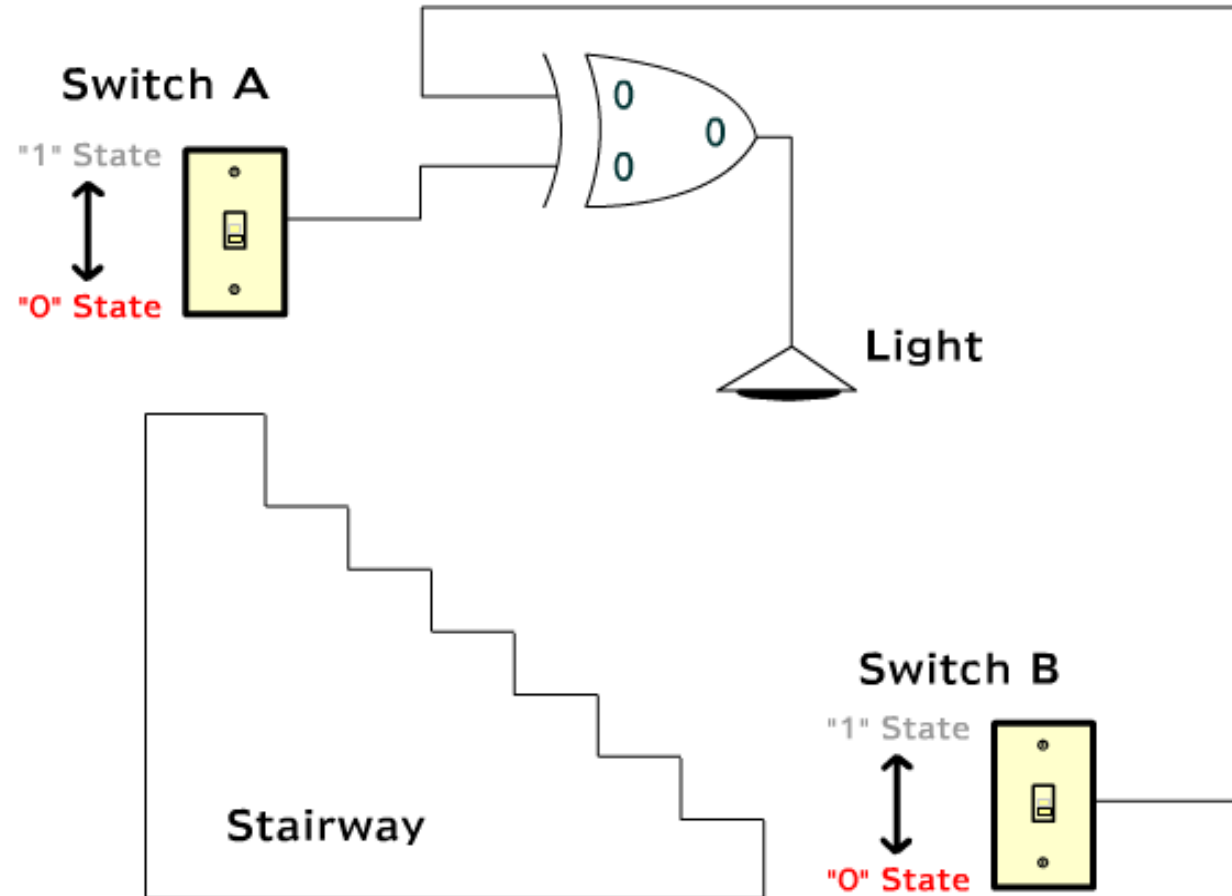
# 5. Basic Logic Operation

## Complex Operations

An Ex-OR gate can be used to turn the overhead light on or off from either the top or the bottom of the stairway.

♣ XOR Gate

Equivalent Circuit:



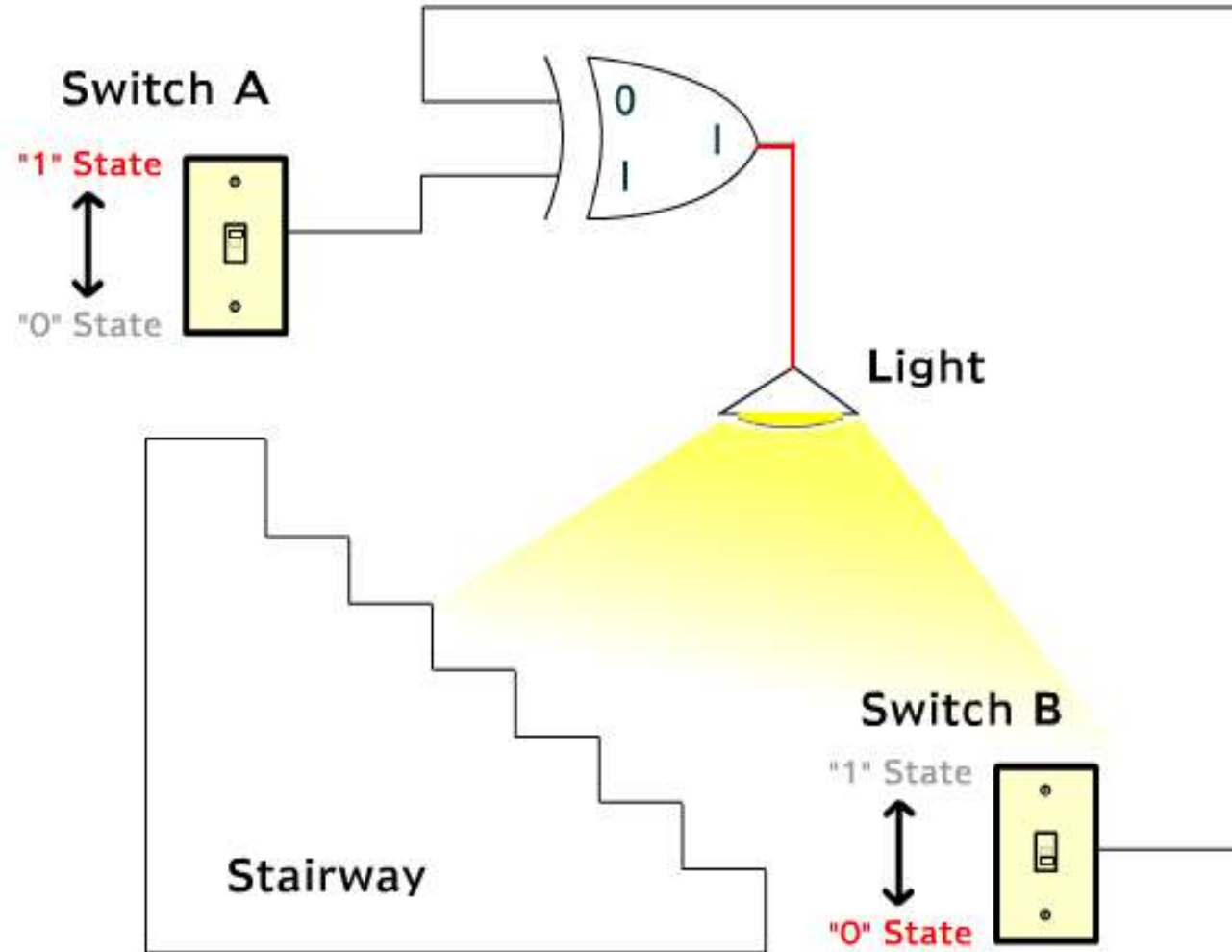


# 5. Basic Logic Operation

## Complex Operator

♣ XOR Gate

Equivalent Circuit:

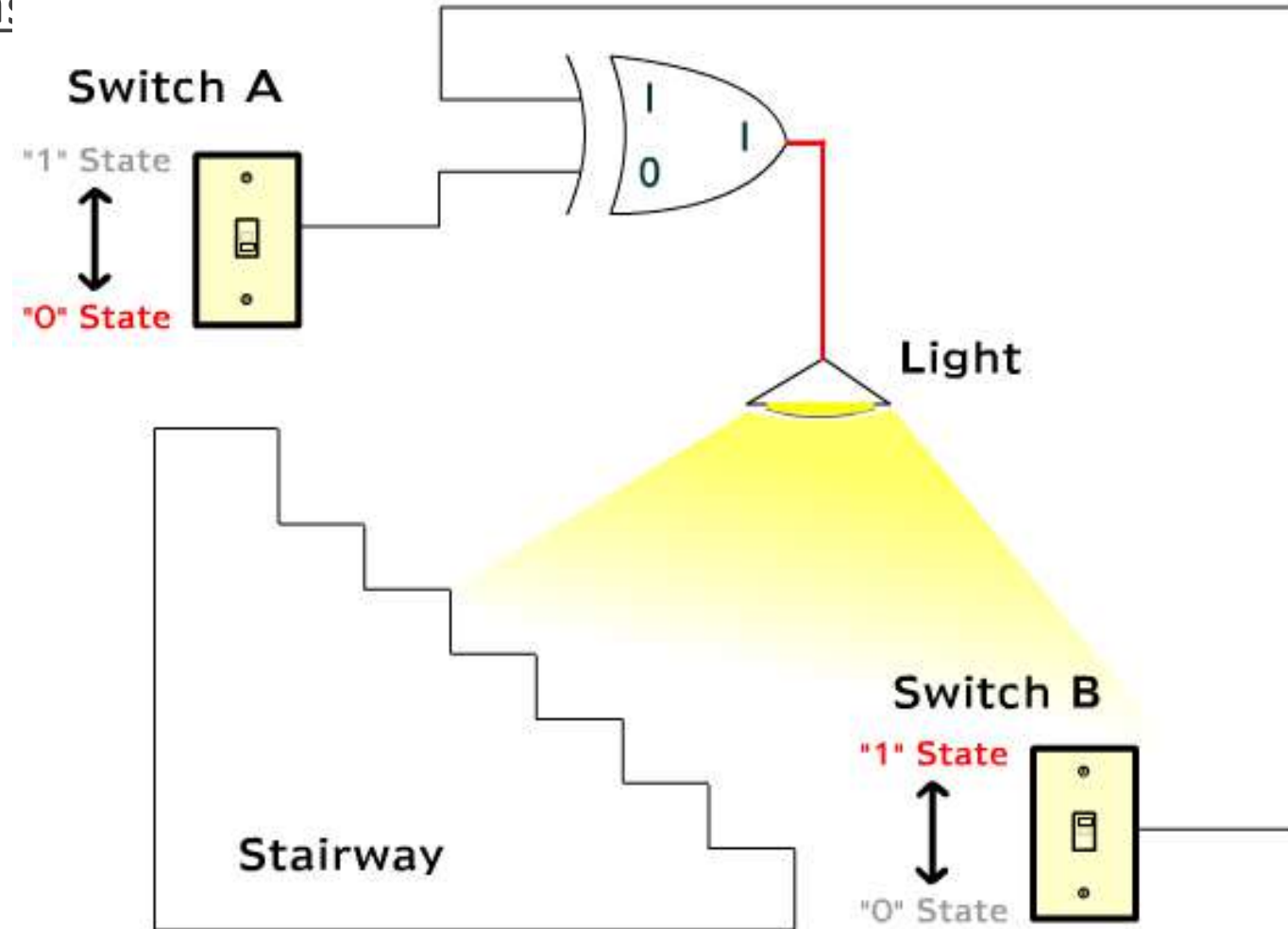


# 5. Basic Logic Operation

## Complex Operation:

♣ XOR Gate

Equivalent Circuit:

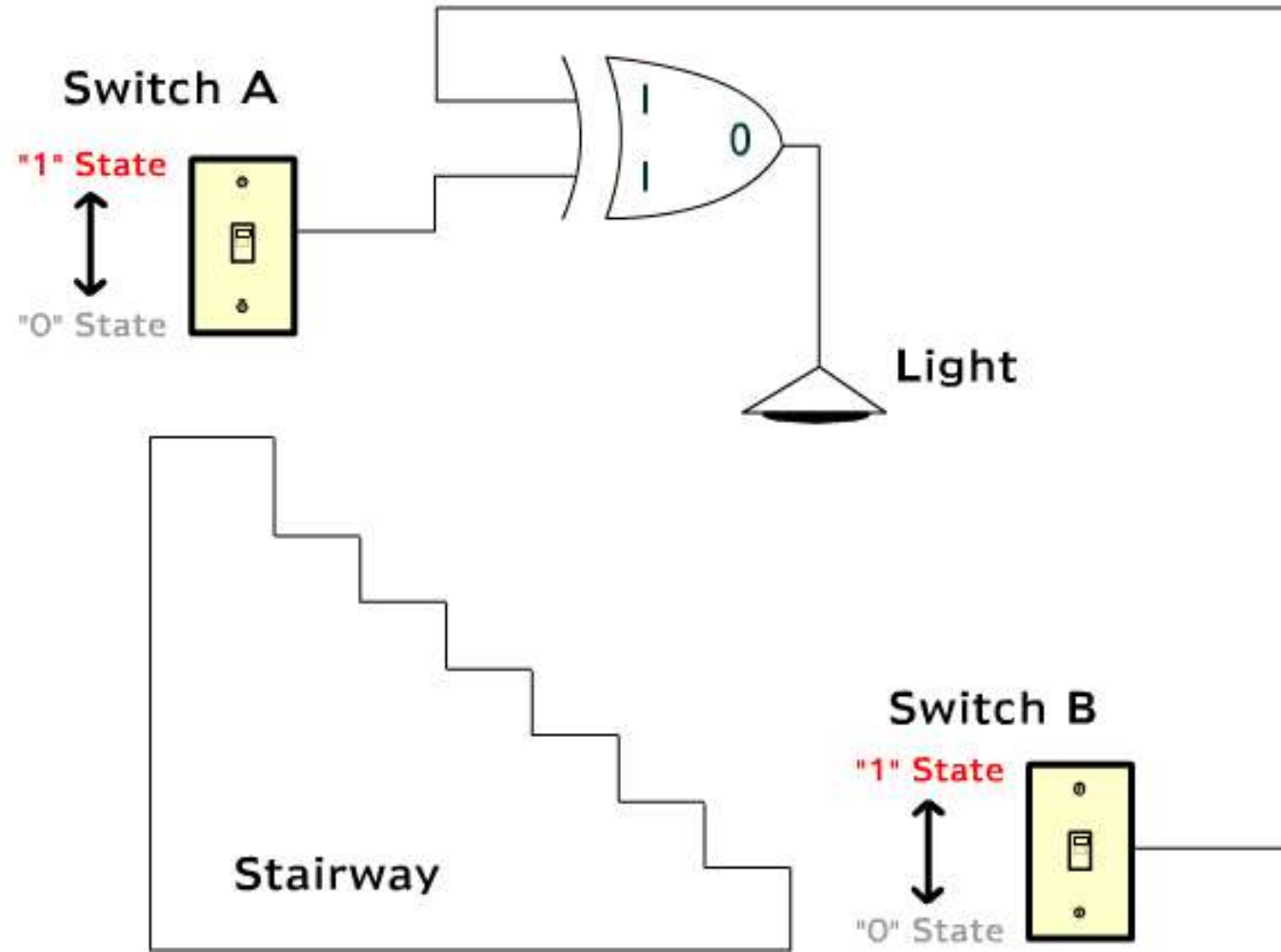


# 5. Basic Logic Operation

## Complex Operation:

♣ XOR Gate

Equivalent Circuit:

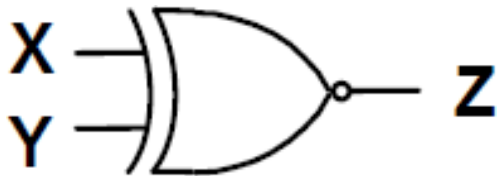


# 5. Basic Logic Operation

## Complex Operations

---

- ♣ XNOR Gate: The 'Exclusive-NOR' gate circuit does the opposite to the XOR gate. It will give a low output when its two inputs are different. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.



Boolean Equation:  $Z = \overline{A \oplus B}$

XNOR works as a Equality Detector, because  $Z=1$  for  $X=Y$ .

Truth Table:

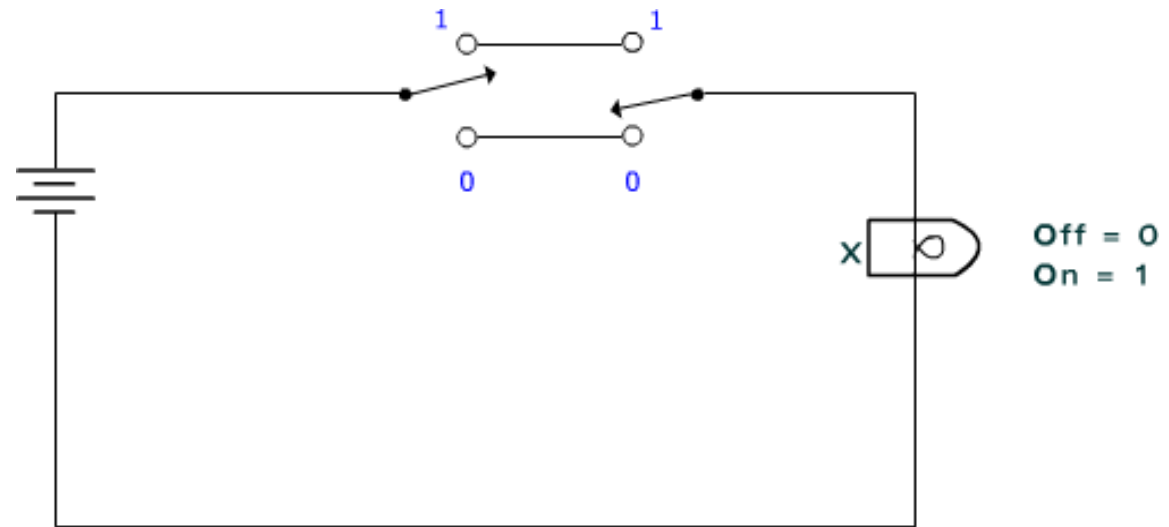
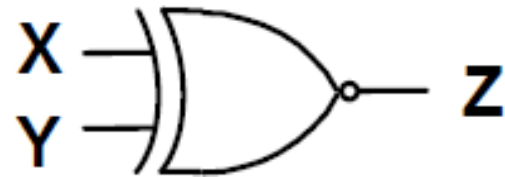
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

# 5. Basic Logic Operation

## Complex Operations

---

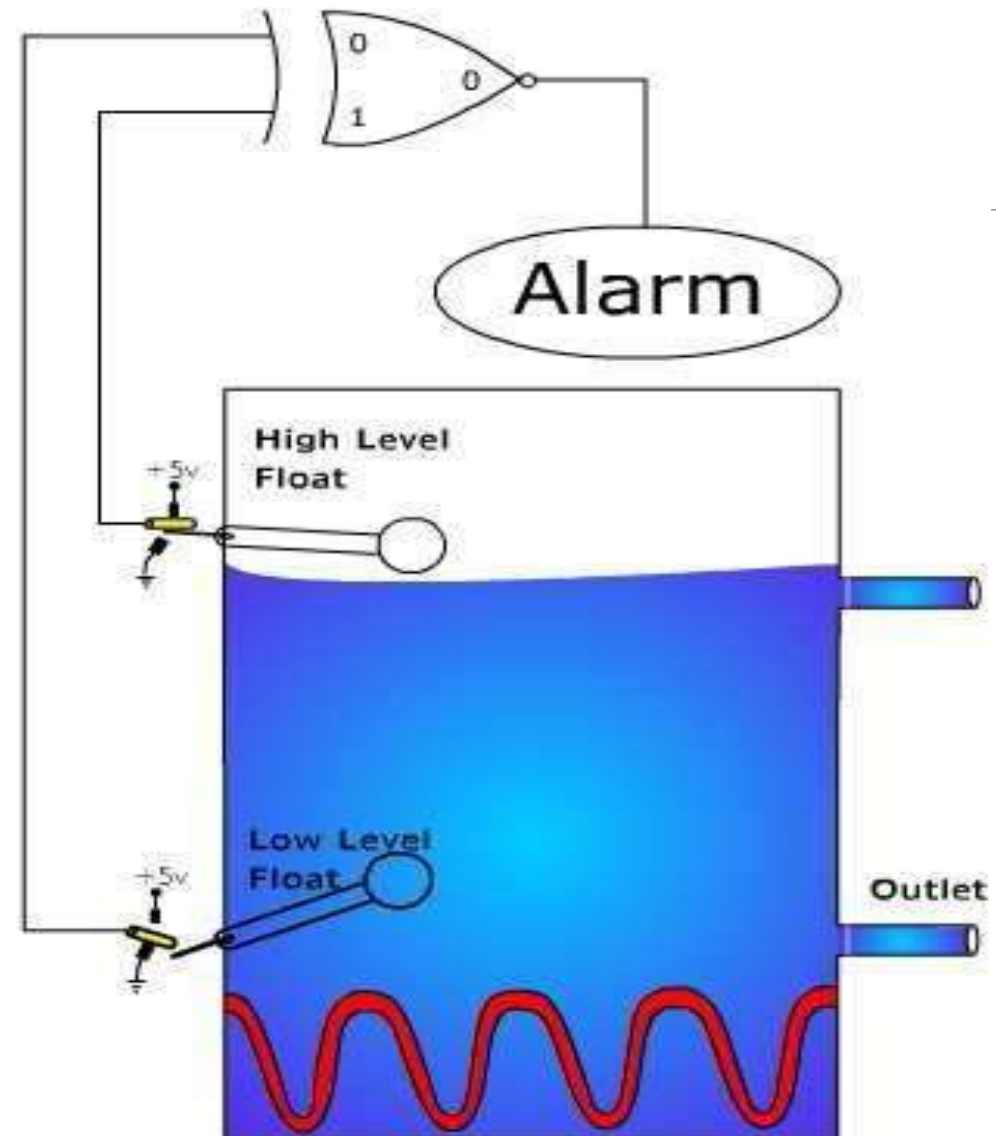
♣ XNOR Gate Equivalent Circuit:



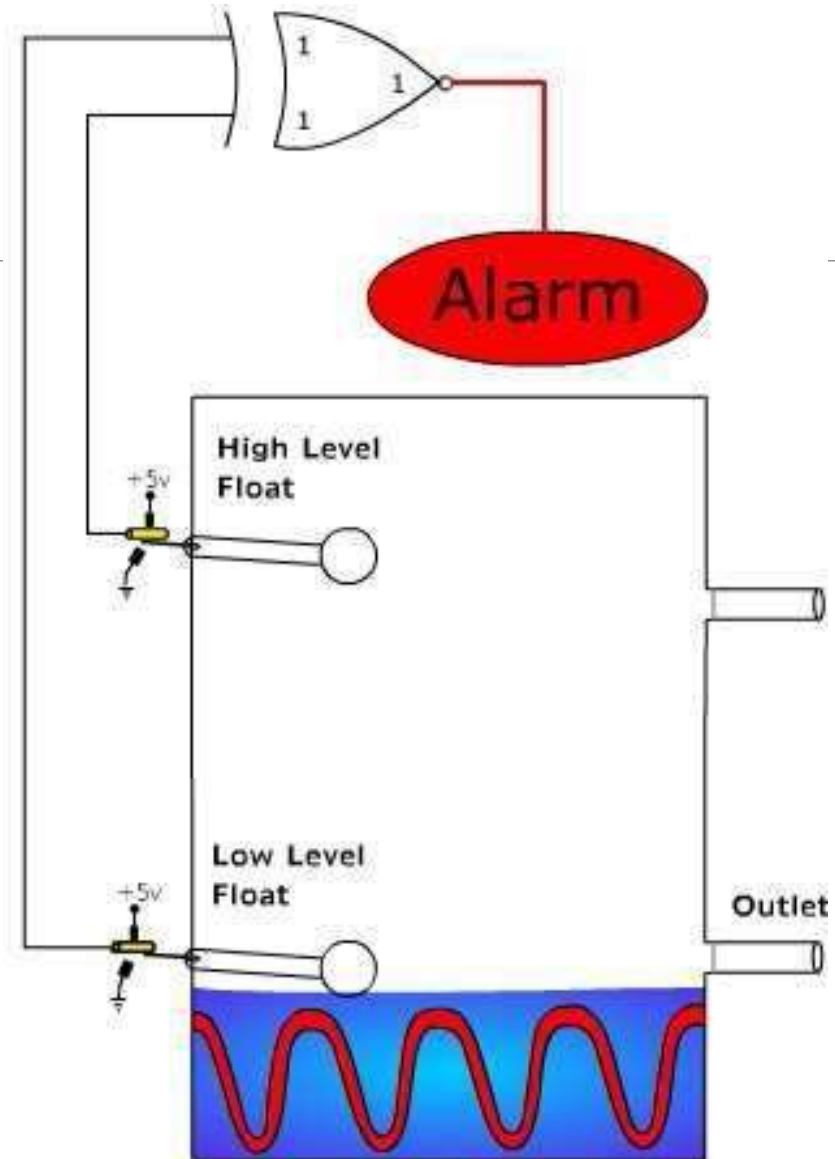
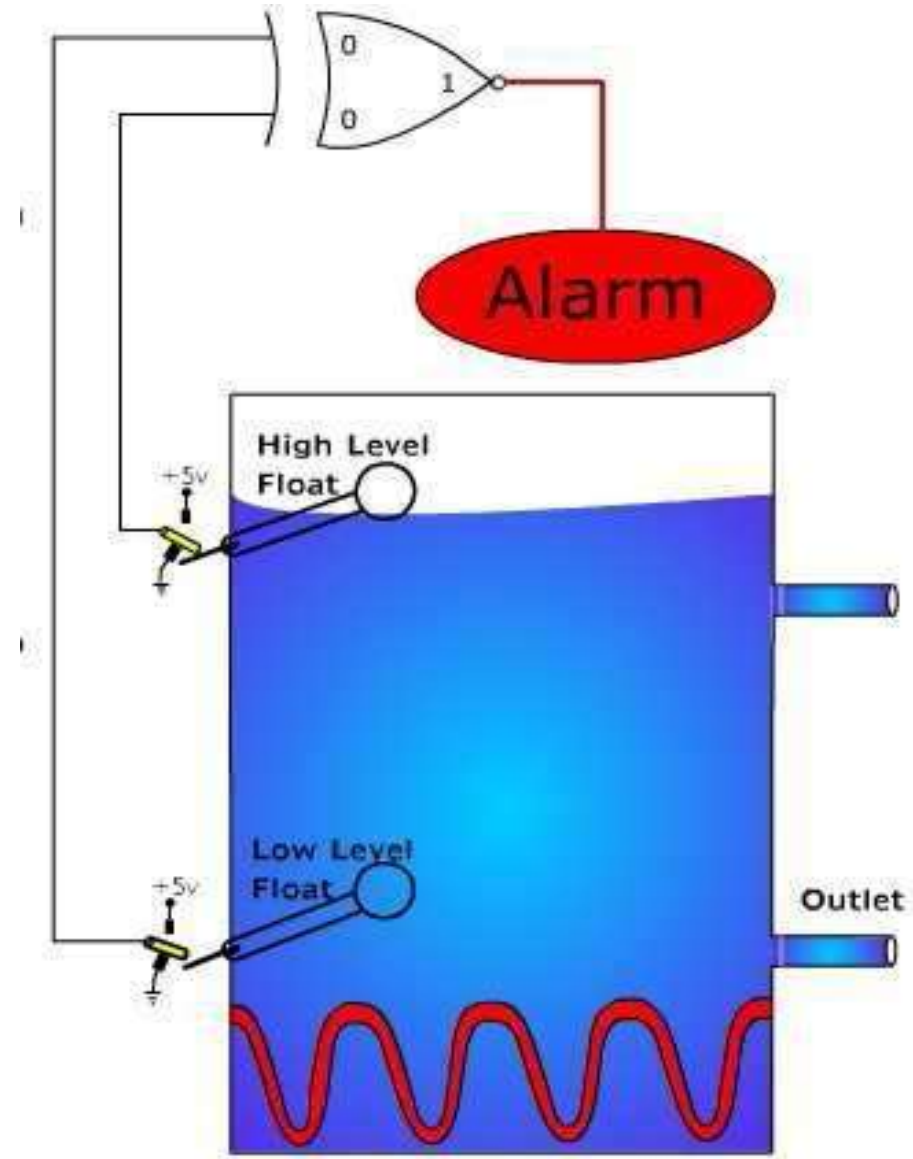
## 5. Complex Operations

The diagram shows a heat exchanger tank that is used to warm water as it passes through. If the level becomes too high, the water will overflow. If it becomes too low, the heating element will burn out when it is not covered with water.

To alert someone if the water level gets too high or too low, float switches are placed at two locations inside the tank. Whenever a float is pushed upward when in contact with water, it will cause the float's lever to break contact with an armature, which causes a logic 0 to be applied to the Ex-NOR gate. Whenever the float is not in contact with water, the float lowers and causes the armature to connect to a +5 volt contact, which causes a logic "1" state to be applied to the input of the Ex-NOR gate. The gate output produces a "1" state and turns the alarm on when +5 volts are simultaneously applied to its inputs, or when a 0 volt ground is simultaneously applied to its inputs. Whenever the water level is between the float switches, a logic "1" is applied to the Ex-NOR gate from the high level float, and a logic "0" is applied from the low level float. Opposite states applied to the Ex-NOR gate will cause its output to be a logic "0" and not activate the alarm.



## 5. Complex Operations



# 5. Basic Logic Operation

## Simplification of Logic Circuits

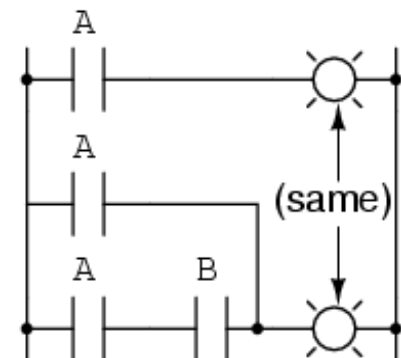
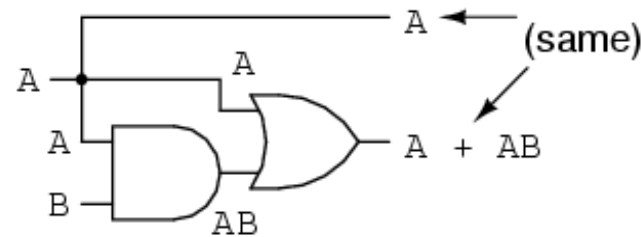
- ❖ The previous Boolean identities and properties are used to simplify complex digital circuits in order to obtain more direct ways of circuit implementation.
- ❖ The circuit will have the same function but with fewer components, and consequently more liability at a lower cost of production.
- ❖ Like in normal Algebra, the use of Theorems defines the way to simplify digital circuits using Boolean Algebra.

$$\mathbf{A + AB = A}$$

$$A + AB = A$$

$$A + \overline{A}B = A + B$$

$$(A + B)(A + C) = A + BC$$



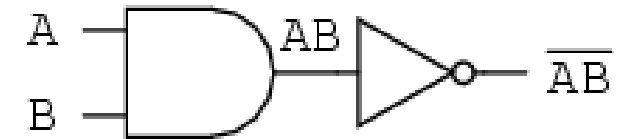
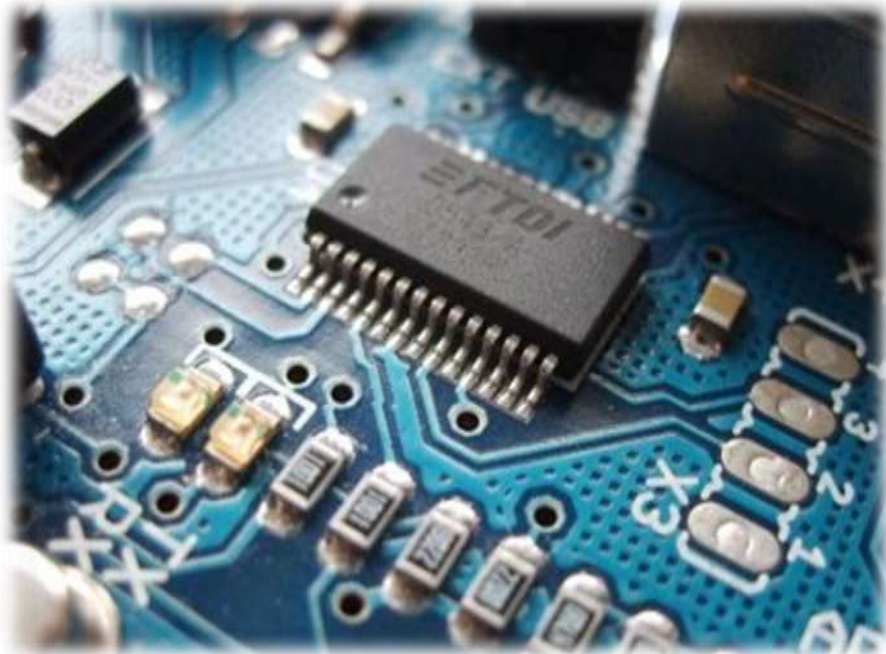


# 5. Basic Logic Operation

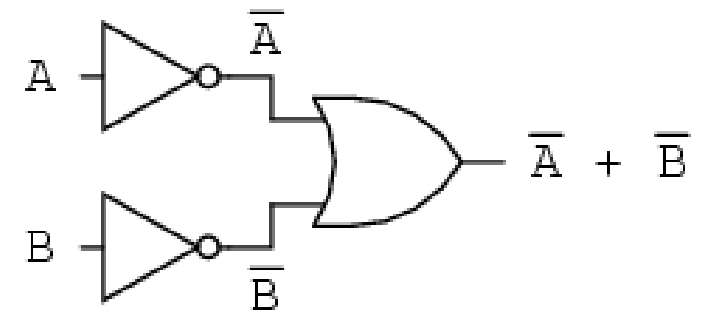
## Simplification of Logic Circuits

### ♣ DeMorgan's Theorems

DeMorgan was a mathematician that developed Simplification of Logic Circuits in Boolean algebra.



*... is equivalent to ...*

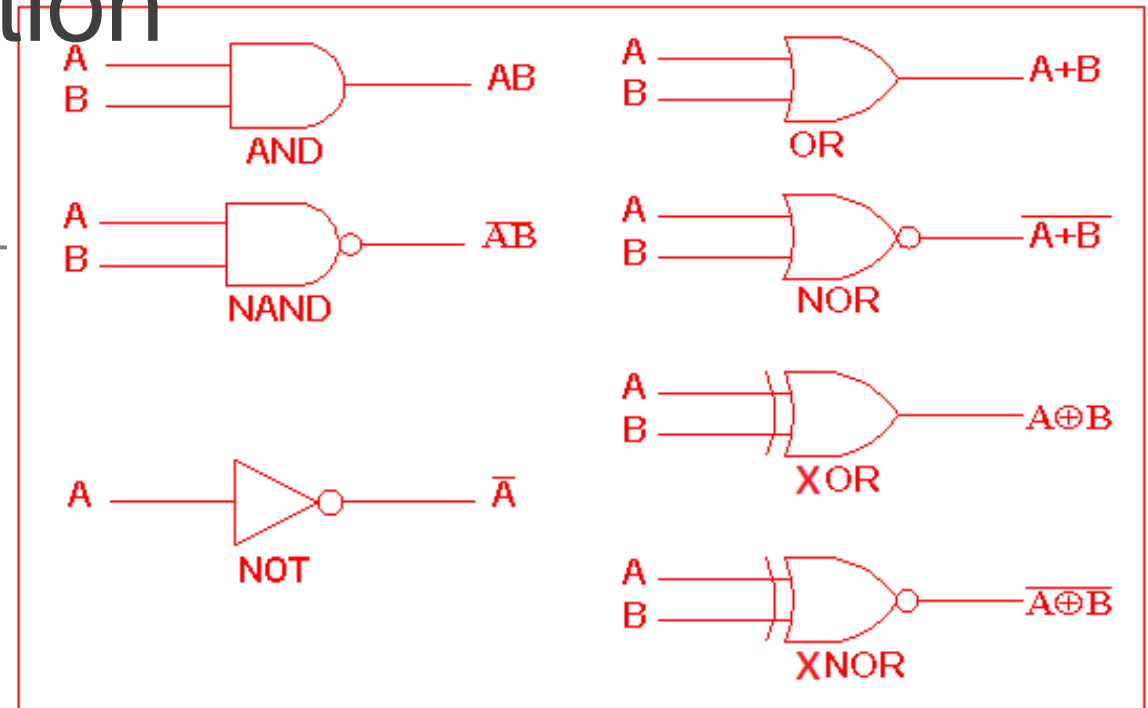


$$\overline{AB} = \overline{A} + \overline{B}$$

# 5. Basic Logic Operation

## Summary of Logic Gates

- ♣ Logic Gates Symbols
- ♣ Logic Gates Truth Table



NOT gate	
A	$\overline{A}$
0	1
1	0

INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	XOR	XNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

# DIGITAL ELECTRONICS

## Section 6 Integrated Circuits Families

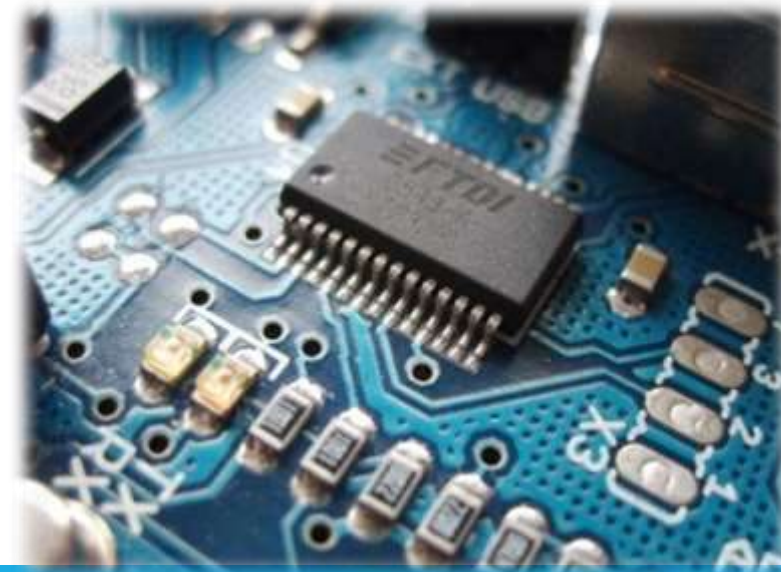


# 6. Integrated Circuits Families

## Integrated Circuits

---

- ♣ An integrated circuit (IC, a chip, or a microchip) is a set of electronic circuits on one small plate of semiconductor material, normally silicon.
- ♣ This can be made much smaller than a discrete circuit made from independent components.
- ♣ Integrated circuits are used in virtually all electronic equipment today and have revolutionized the world of electronics.

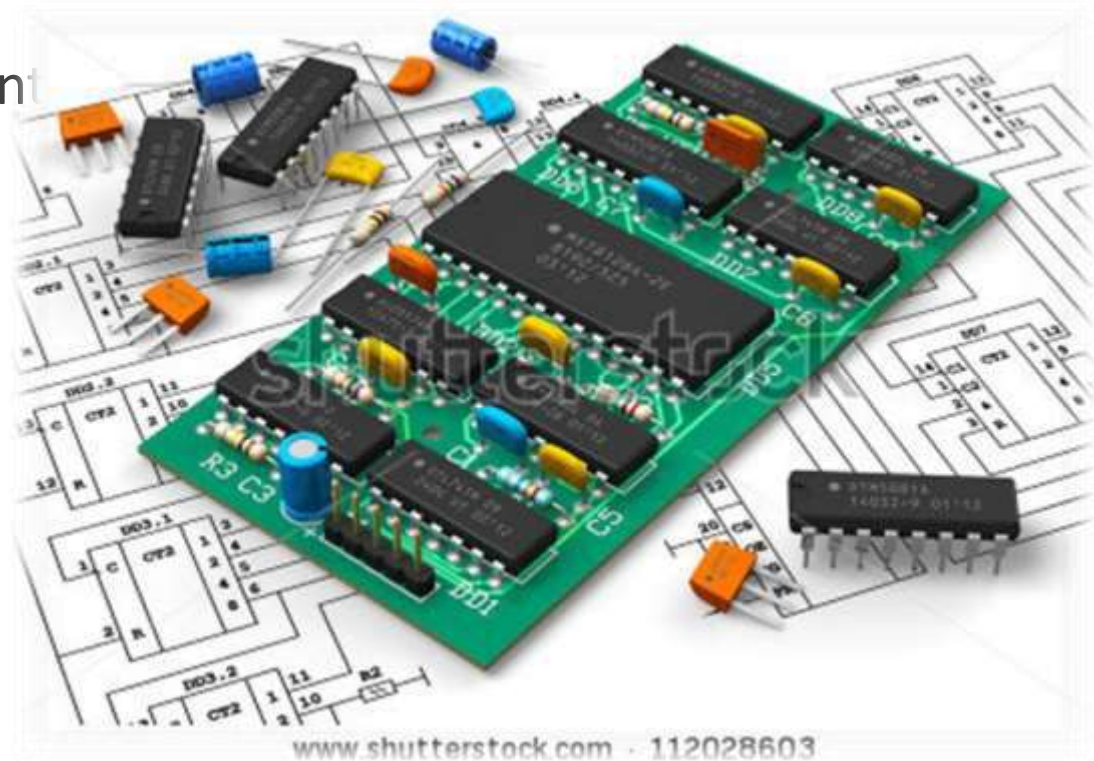


# 6. Integrated Circuits Families

## Advantages of ICs

---

- ♣ Smaller components
- ♣ Less volume and weight of the equipment
- ♣ Reduction of power consumption
- ♣ Cost reduction
- ♣ More reliable

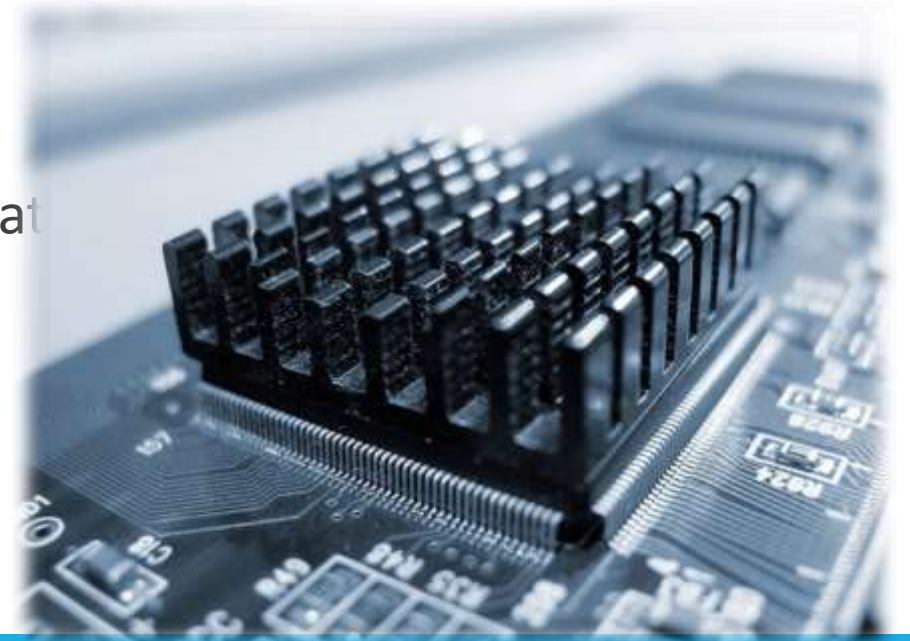


# 6. Integrated Circuits Families

## Integration Scales

---

- ♣ ICs can be made very compact, having up to several billion transistors and other electronic components in an area the size of a fingernail.
- ♣ The width of each conducting line in a circuit (the line width) can be made smaller and smaller as the technology advances, in 2008 it dropped below 100 nanometers and in 2013 it is expected to be in the teens of nanometers.
- ♣ The electronic circuit industry has been going towards the integration of the biggest number of components in a single chip (Integrat



# 6. Integrated Circuits Families

## Classification of Integrated Circuits

The classification of Integrated Circuits is done accordingly to the density of integration:

- ♣ SSI (Small Scale Integration) – Elementary logic functions, up until 100 components and implement around 10 logic gates.
- ♣ MSI (Medium Scale Integration) – Encoders, multiplexers, counters, etc. Up until 100 and 1000 components and implement around 100 logic gates.
- ♣ LSI (Large Scale Integration) – Memories, microprocessors, calculators, etc. Up until 1000n and 10000 components and implement around 1000 logic gates.
- ♣ VLSI (Very Large Scale Integration) – Over 10000 components in a single chip, technology that has between 20 and 30 years.
- ♣ SLSI (Super-Large Scale Integration) – Allows between 50000 and 100000 components in a single chip.

# 6. Integrated Circuits Families

## Logic Families

---

There are several Families of logic integrated circuits, from which we can point out:  
TTL, CMOS, PMOS/NMOS and ECL

The most important parameters on the datasheets for every family are:

- ♣ Power supply voltage,
- ♣ Maximum operating temperature,
- ♣ Fan-out (how many logic gate inputs of the same family we can connect to the output of a gate on the chip),
- ♣ Noise Immunity (the amount of noise that can withstand without changing the operating conditions),
- ♣ Propagation Time (time between the entry of an input and outgoing of the output),
- ♣ Power Dissipation (usually referring to a single gate or function),



# 6. Integrated Circuits Families

## Logic Families

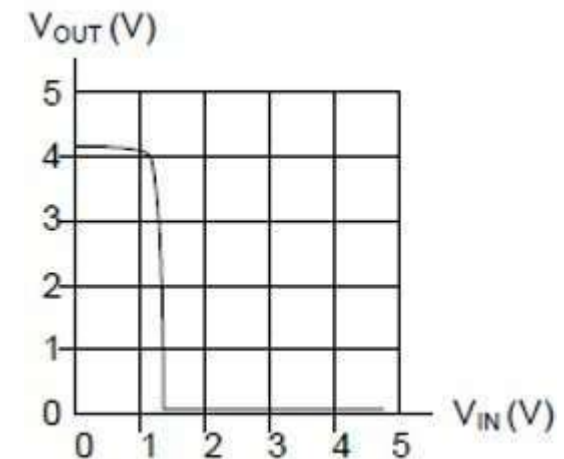
- ♣ Each family has its own advantages and disadvantages. We can say that ideally they should allow: Huge density of integration,  
Fast commutation,  
Low power consumption,  
Maximum immunity to noise and temperature variations,  
Low cost,
- ♣ The choice of one particular logic family over other depends on the project. More than one family may be used depending on the application.
- ♣ Nowadays the most commonly used are TTL and CMOS.
- ♣ Power consumption and propagation time have the indirect proportion. If the power consumption decreases so the propagation time increases and same to the invert.

# 6. Integrated Circuits Families

## Logic Families – TTL

---

- ♣ TTL means Transistor-Transistor Logic,
- ♣ The first integrated circuit chip was manufactured in 1963, called “Sylvania Universal High-Level Logic family (SUHL)”. The manufactured components were used in the control of military missiles,
- ♣ The TTL technology became popular with the 5400 series for military applications in 1964,
- ♣ The transition zone for the TTL family is very close to 1,2V,



# 6. Integrated Circuits Families

## Logic Families – TTL

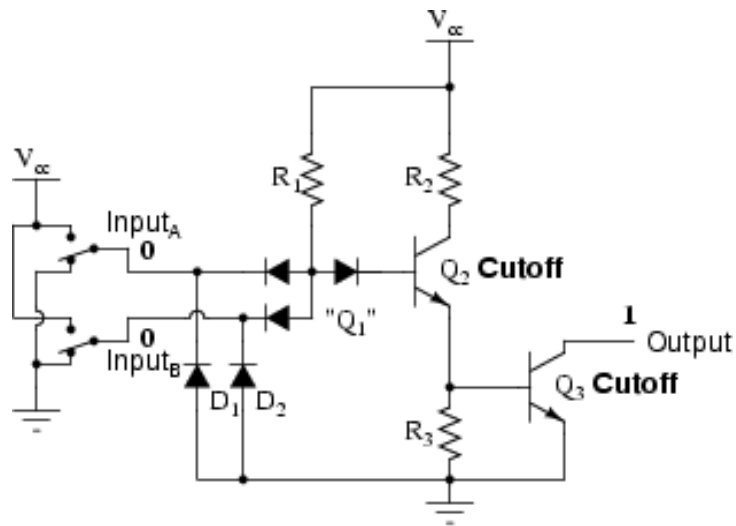
---

- ♣ Main generic characteristics are:
  - Supply Voltage between 4,5V and 5,5V
  - Temperature of Operation between 0°C and 70°C
  - Transition zone between 1V and 1,5V
  - Average Propagation Time is 10ns
  - Power Dissipation of 10mW per gate

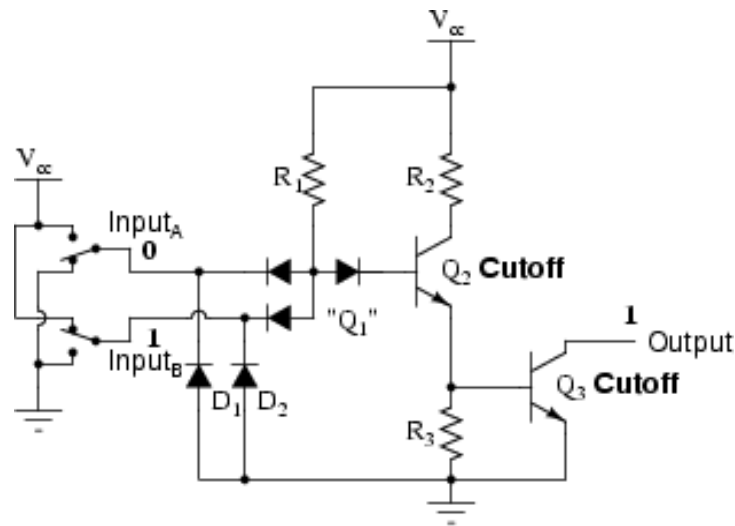
# 6. Integrated Circuits Families

## Logic Families – TTL

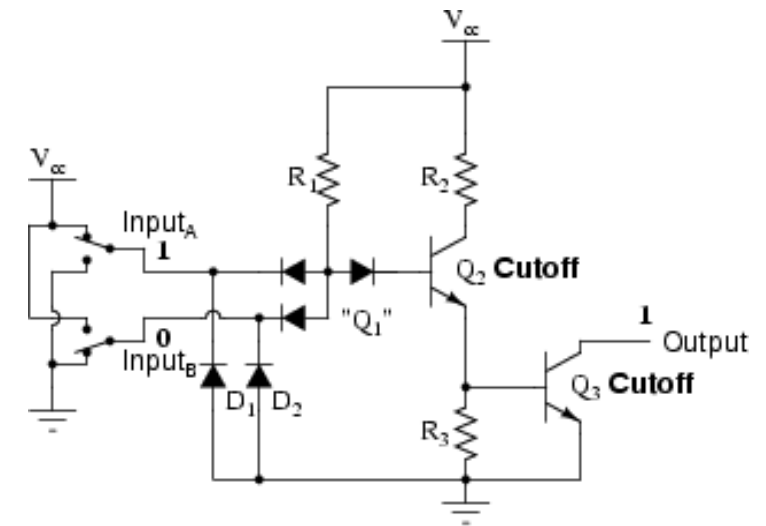
♣ NAND Gate: If either of the inputs are grounded, the logic gate will work as a NAND gate and the output goes "high". The following series of illustrations shows this for three input states (00, 01, and 10):



Input<sub>A</sub> = 0  
Input<sub>B</sub> = 0  
Output = 1



Input<sub>A</sub> = 0  
Input<sub>B</sub> = 1  
Output = 1

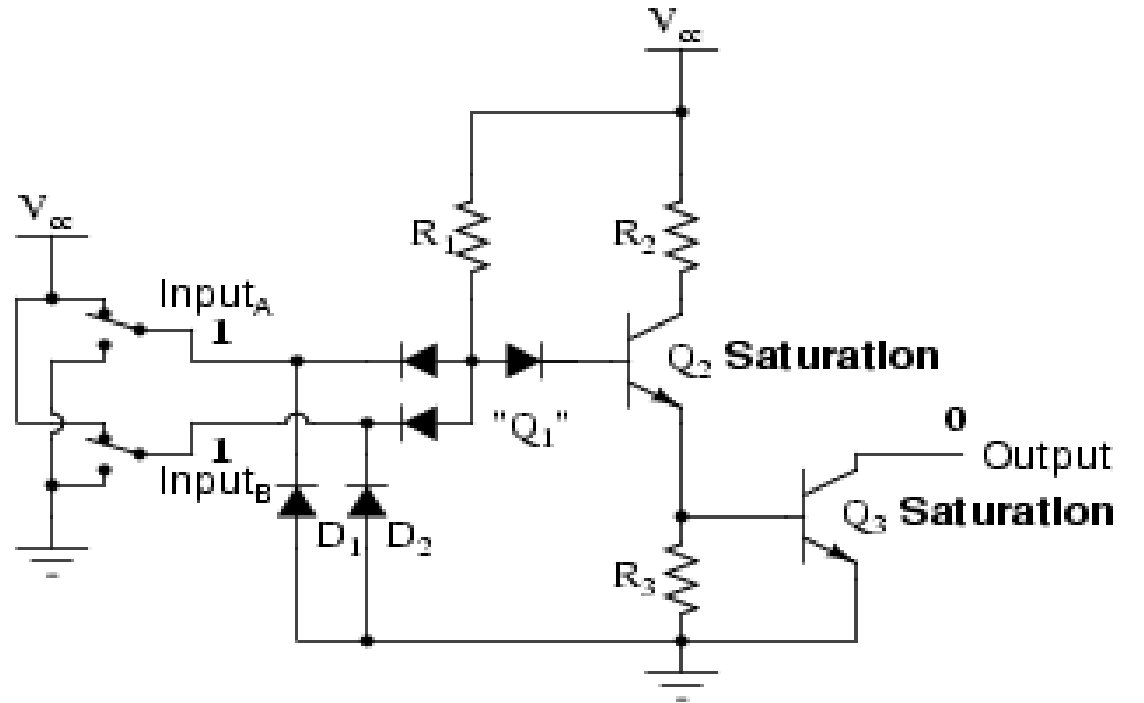


Input<sub>A</sub> = 1  
Input<sub>B</sub> = 0  
Output = 1

# 6. Integrated Circuits Families

## Logic Families – TTL

- ♣ NAND Gate when both inputs are "high" (1), the output is guaranteed to be floating ("high")



Input<sub>A</sub> = **1**

Input<sub>B</sub> = **1**

Output = **0**

# 6. Integrated Circuits Families

## Logic Families – TTL

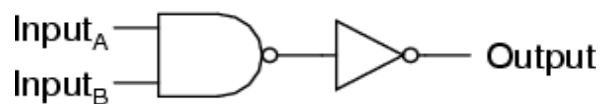
- ♣ NAND to AND: to create an AND function using TTL circuitry, we need to increase the complexity of the NAND circuit by adding an inverter stage to the output:

*AND gate*

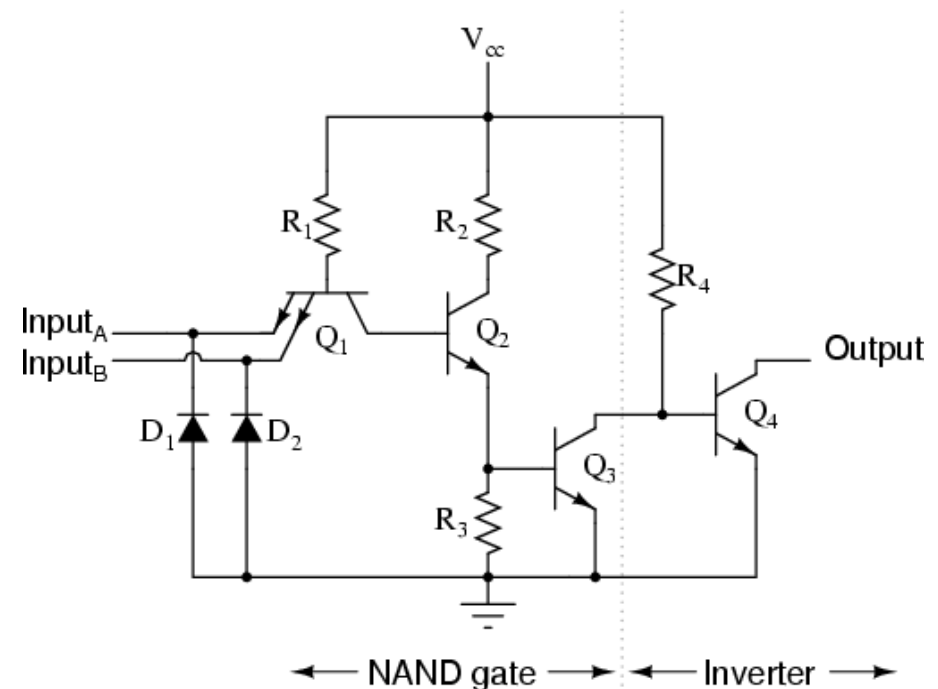


A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

*Equivalent circuit*



*AND gate with open-collector output*



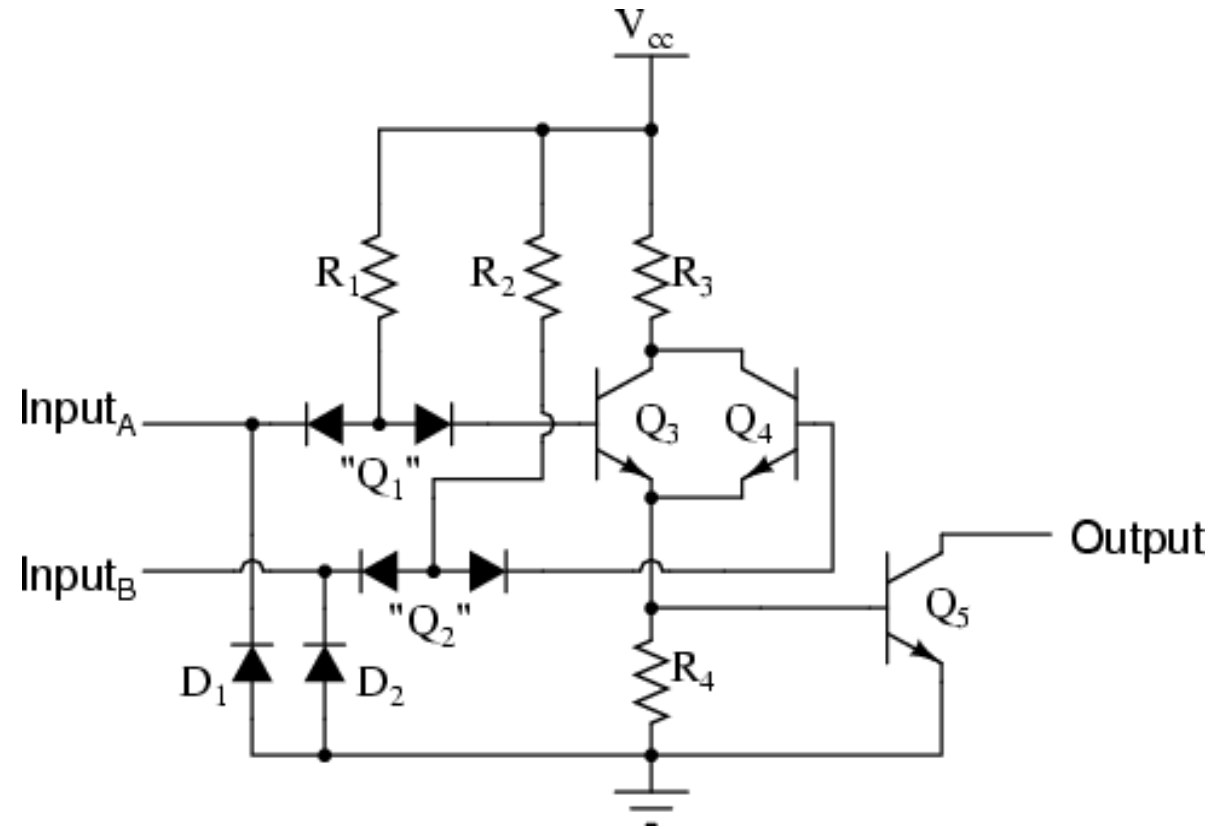
# 6. Integrated Circuits Families

## Logic Families – TTL

♣ NOR Gate: if either of the inputs are high, the output will be low and when both inputs are low so the output will be high.



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

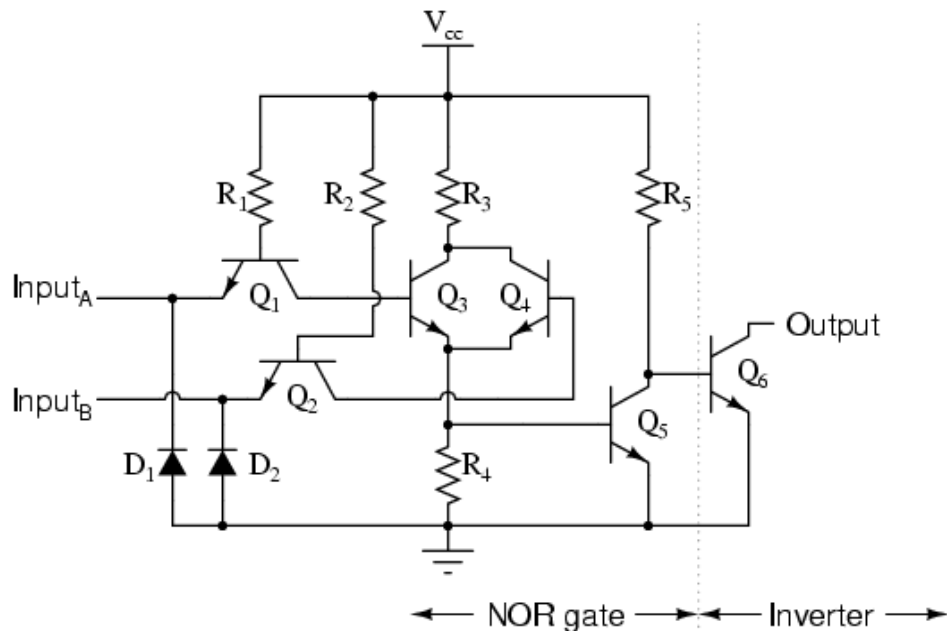


# 6. Integrated Circuits Families

## Logic Families – TTL

❁ NOR to OR: In order to turn this NOR gate circuit into an OR gate, we would have to invert the output logic level with another transistor stage, just like we did with the NAND-to-AND gate example:

*OR gate with open-collector output*

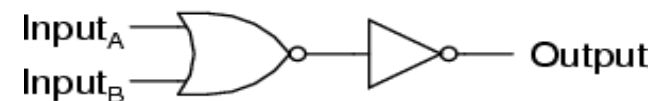


*OR gate*



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

*Equivalent circuit*





# 6. Integrated Circuits Families

## Logic Families – CMOS

---

- ♣ CMOS technology is known as Complementary - Symmetry metal – Oxide – Semiconductor.
- ♣ The words "complementary - symmetry" refers to the fact that the circuit has a pair of complementary MOSFET transistors.
- ♣ This technology exists in most of the integrated circuits.
- ♣ CMOS technology first used appears in microprocessors, micro controllers and memories.
- ♣ CMOS technology is also used in a big variety of analogic circuits like image sensors, data converters and communications equipments.
- ♣ Two of the most important CMOS characteristics are better immunity to noise and low power consumption.
- ♣ CMOS devices don't produce as much heat as the other logic families.
- ♣ CMOS also allows a bigger density of logic function in a single chip.

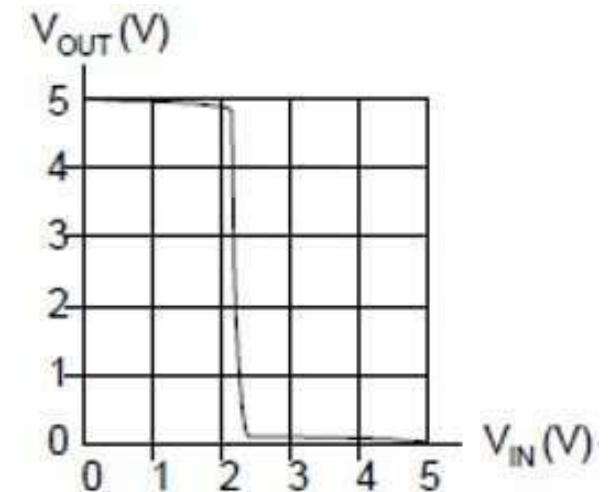
# 6. Integrated Circuits Families

## Logic Families – CMOS

---

♣ Most significant characteristics are:

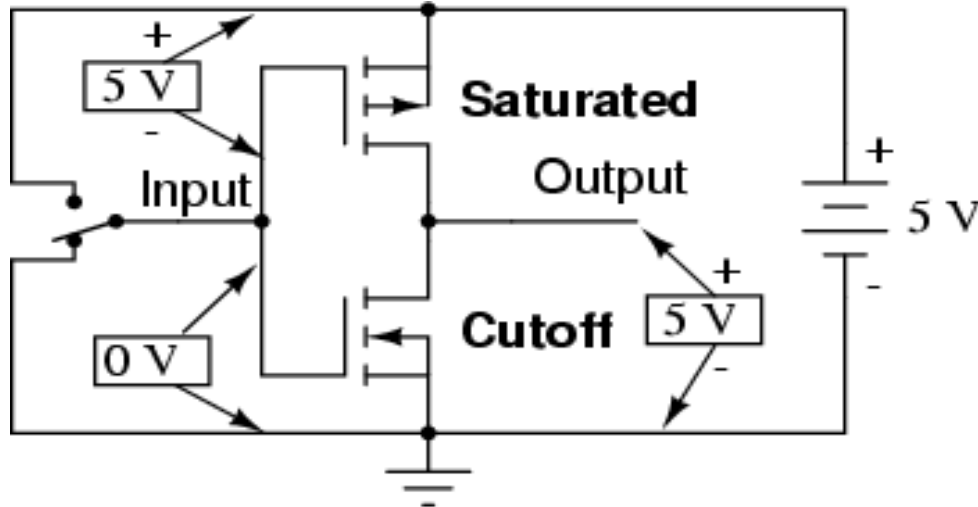
- Supply voltage variable between 3V and 15V,
- Working temperature range until 85°C,
- Bigger immunity to noise,
- Propagation time varies inversely to supply voltage, is 125ns for 5V and 45ns for 15V,
- Power dissipation is 10nW per gate,
- Transition zone for CMOS technology is very near to the middle, very far from the noise zone, unlike TTL technology,



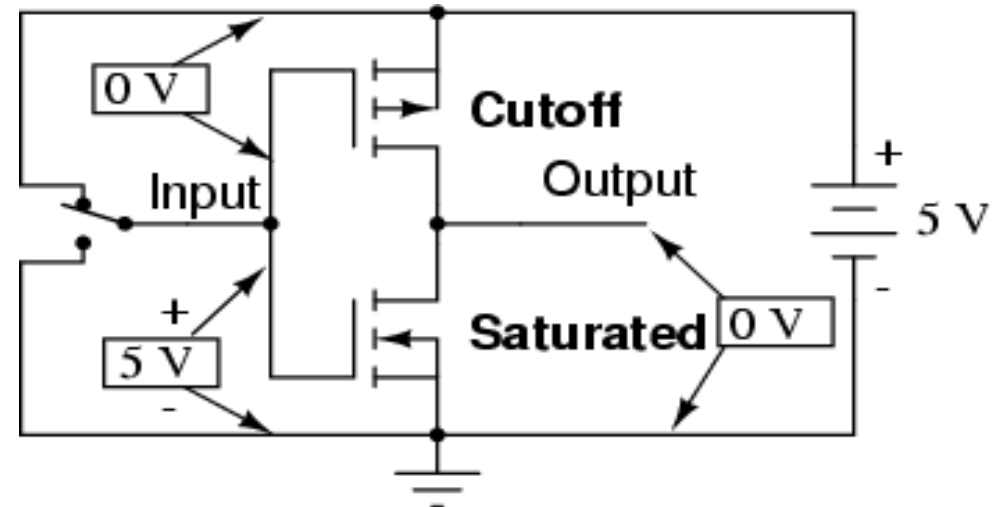
# 6. Integrated Circuits Families

## Logic Families – CMOS

♣ NOT Gate: This circuit exhibits the behavior of an inverter, or NOT gate.



Input = "low" (0)  
Output = "high" (1)

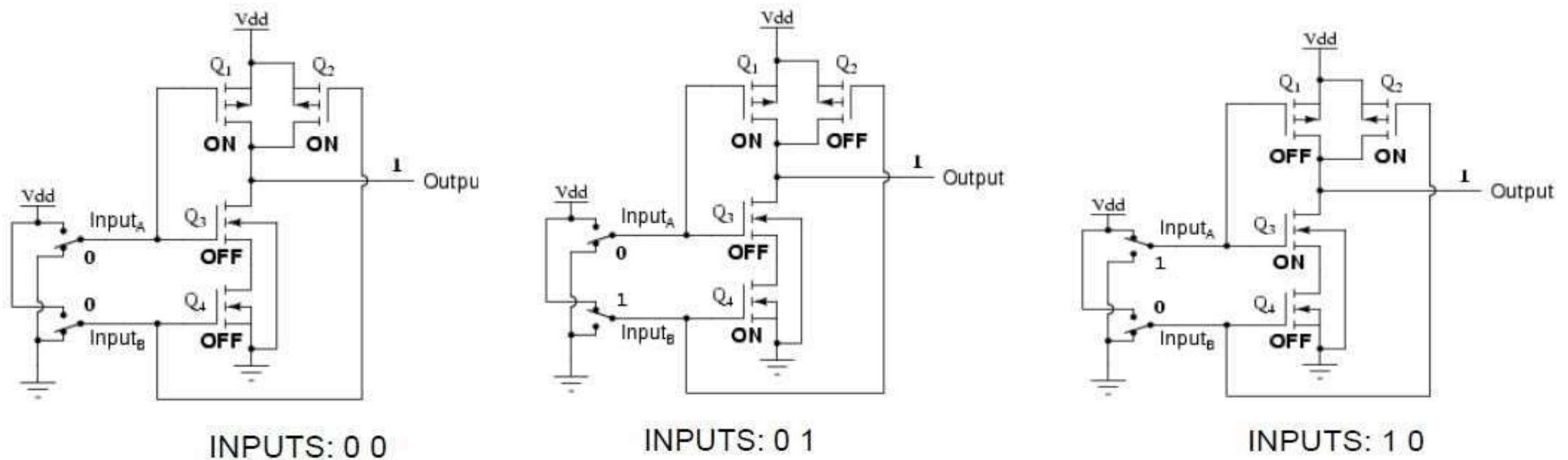


Input = "high" (1)  
Output = "low" (0)

# 6. Integrated Circuits Families

## Logic Families – CMOS

- ❖ NAND Gate: the logic gate will work as a NAND gate and the output goes "high". The following series of illustrations shows this for three input states (00, 01, and 10):

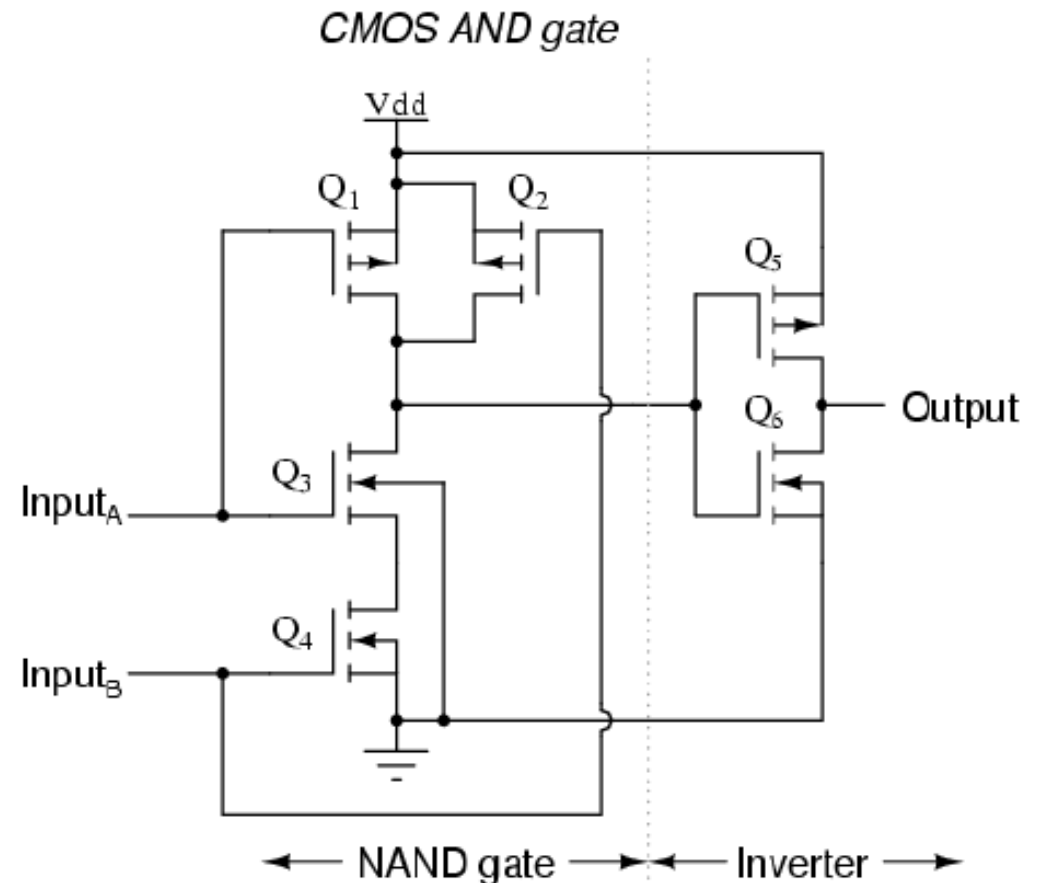


# 6. Integrated Circuits Families

## Logic Families – CMOS

- ♣ NAND to AND: As with the TTL NAND gate, the CMOS NAND gate circuit may be used as a starting point for the design of a CMOS AND gate. All the

signal:

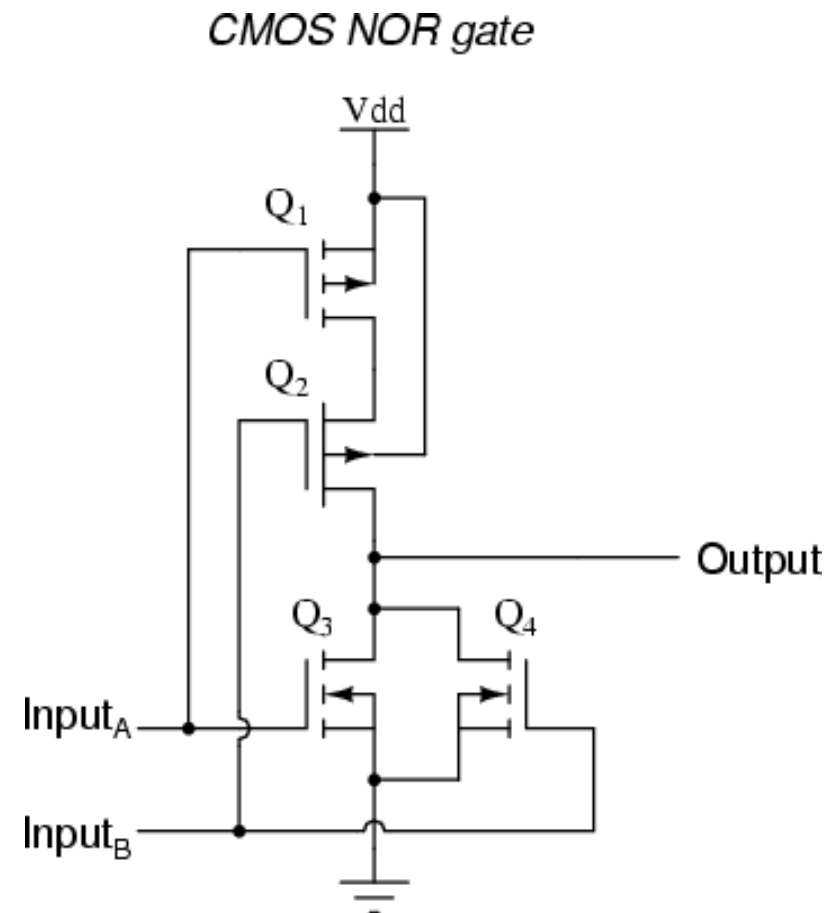


# 6. Integrated Circuits Families

## Logic Families – CMOS

---

♣ NOR Gate: if either of the inputs are high, the output will be low and when both inputs are low so the output will be high.

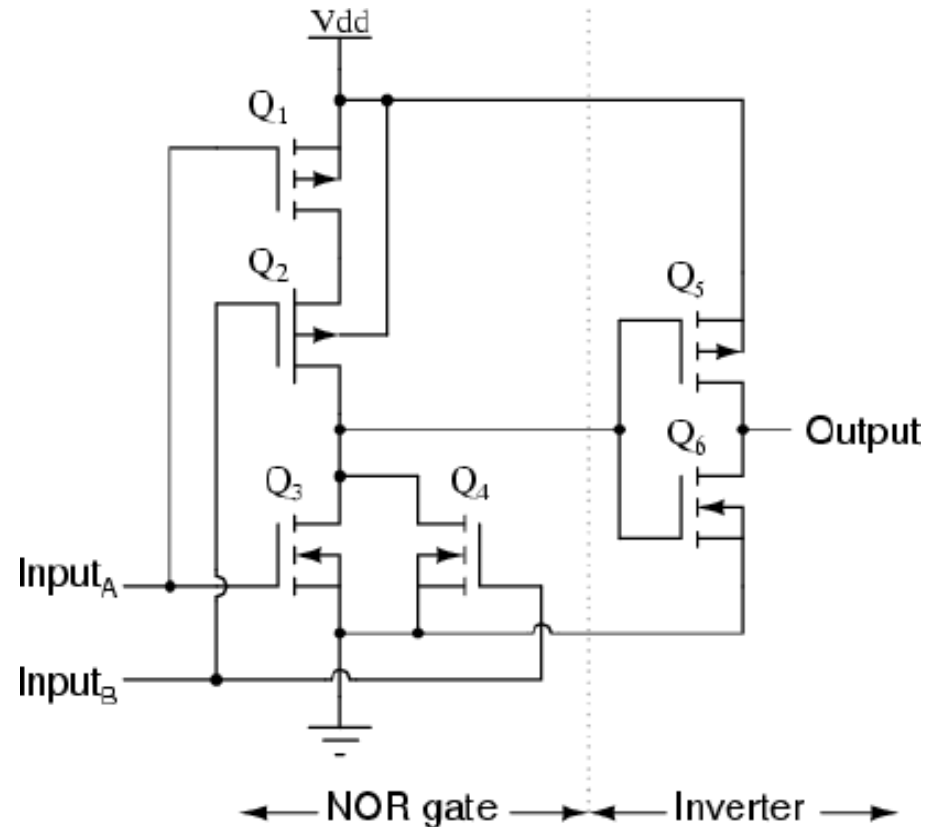


# 6. Integrated Circuits Families

## Logic Families – CMOS

---

- ♣ NOR to OR: The OR function may be built up from the basic NOR gate with the addition of an inverter stage on the output:



# 6. Integrated Circuits Families

## TTL Versus CMOS

---

♣ Since it appears that any gate possible to construct using TTL technology can be duplicated in CMOS,

why do these two "families" of logic design still coexist? The answer is that both TTL and CMOS have

their own unique advantages:

♣ CMOS has lower power consumption.

♣ TTL dissipates less power.

♣ CMOS has a much wider allowable range of power supply voltages than TTL.

♣ TTL is faster than CMOS.



# 6. Integrated Circuits Families

## Logic Families – NMOS, PMOS and ECL

---

- ♣ nMOS: Uses MOSFET transistors to implement logic circuits. Is also known as “pull-down” between the output and the power source’s negative pole.
- ♣ pMOS: Uses MOSFET transistors to implement logic circuits. Is also known as “pull-up” between the output and the power source’s positive pole. Its characteristics are similar to nMOS technology.
- ♣ ECL: Originally called “*current steering logic*” and was used in IBM7090 and IBM7094 computers.

# DIGITAL ELECTRONICS

## Section 7

### COMBINATIONAL CIRCUITS



# 7. Combinational Circuits

## INTRODUCTION TO COMBINATIONAL CIRCUITS

---

- ♣ The term "combinational" comes to us from mathematics. With combinational logic, the circuit produces the same output regardless of the order the inputs are changed.
- ♣ There are circuits which depend on when the inputs change, these circuits are called sequential logic.
- ♣ Practical circuits will have a mix of combinational and sequential logic.
- ♣ With sequential logic making sure everything happens in order.
- ♣ With combinational logic performing functions like arithmetic, logic, or conversion.
- ♣ We have already used combinational circuits. Each logic gate discussed previously is a combinational logic function.

# 7. Combinational Circuits

## KINDS OF COMBINATIONAL CIRCUITS

---

- ♣ Half-Adder
- ♣ Full-Adder
- ♣ Half- Subtractor
- ♣ Full-Subtractor
- ♣ Decoder
- ♣ Encoder
- ♣ Multiplexer
- ♣ Demultiplexer
- ♣ Comparator

# 7. Combinational Circuits

## Half-Adder

---

♣ The Half-Adder logic circuit sums 2 numbers of 1 bit.

♣ We can quickly calculate what the answers should be:

$$0 + 0 = 0 \quad 0 + 1 = 1 \quad 1 + 0 = 1 \quad 1 + 1 = 1$$

♣ So we will need two inputs (a and b) and two outputs. The first output will be called  $\Sigma$  because it represents the sum, and the second output will be called C out because it represents the carry out.

♣ The truth table is:

A	B	SUM ( $\Sigma$ )	CARRY ( $C_{out}$ )
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

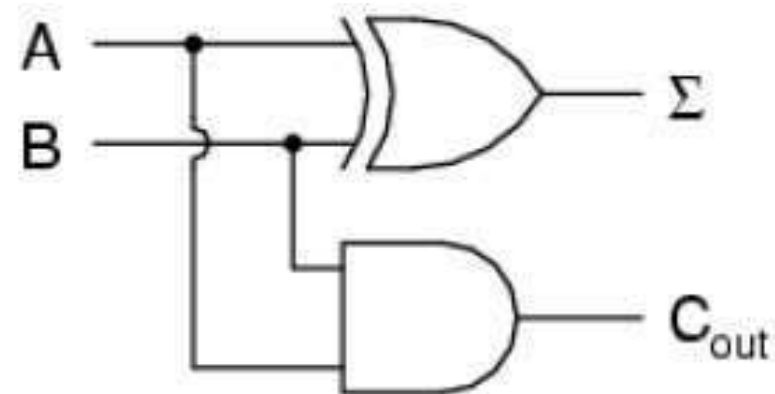
# 7. Combinational Circuits

## Half-Adder

---

♣ Simplifying Boolean equations or making a Karnaugh map will produce the same circuit shown below, but start by looking at the results. The  $\Sigma$  column is our familiar XOR gate, while the C out column is the AND gate. This device is called a half-adder.

A	B	SUM ( $\Sigma$ )	CARRY ( $C_{out}$ )
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



♣ By the minterms we get:

$$\text{SUM} = \bar{A}B + A\bar{B} = A \oplus B$$

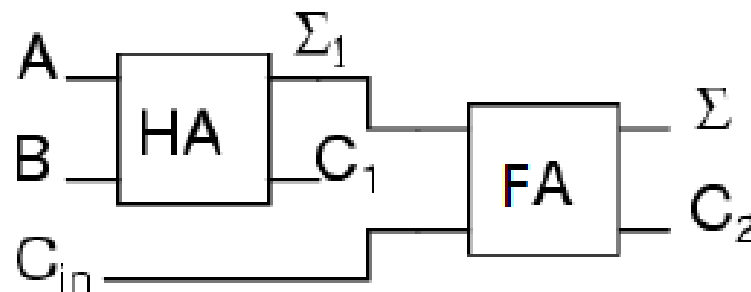
$$C_{out} = A.B$$

# 7. Combinational Circuits

## Full-Adder

- ♣ The Full-Adder logic circuit sums 3 single-bit binary numbers;
- ♣ Looking at a two binary digit sum shows what we need to extend addition to multiple binary digits:
- ♣ The adder needs three inputs; a, b, and the carry from the previous sum, and we can use our two-input adder to build a three input adder.
- ♣  $\Sigma$  is the easy part. Normal arithmetic tells us that if  $\Sigma = a + b + C_{in}$  and  $\Sigma_1 = a + b$ , then  $\Sigma = \Sigma_1 + C_{in}$ .

$$\begin{array}{r} 1\ 1 \\ 1\ 1 \\ +\ 1\ 1 \\ \hline 1\ 1\ 0 \end{array}$$



# 7. Combinational Circuits

## Half-Subtractor

---

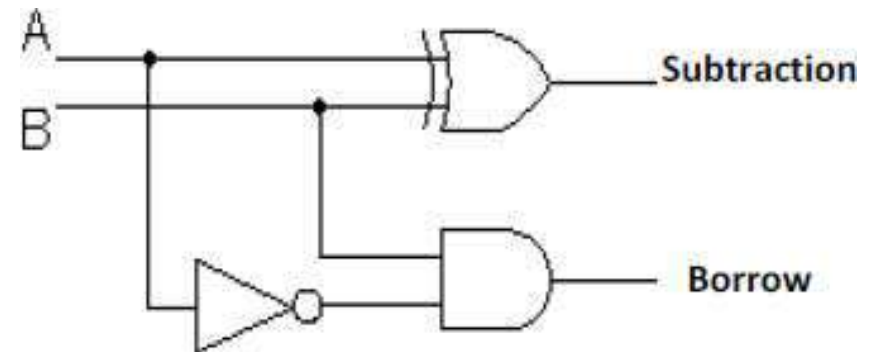
- ♣ The Half-Subtractor logic circuit subtracts 2 numbers of 1 bit;
- ♣ Let's see how we can build a half-subtractor from its truth table:

A	B	Subtraction	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

By the minterms we get:

$$\text{SUM} = AB + A\bar{B} = A \oplus B$$

$$C_{\text{out}} = \bar{A} \cdot B$$



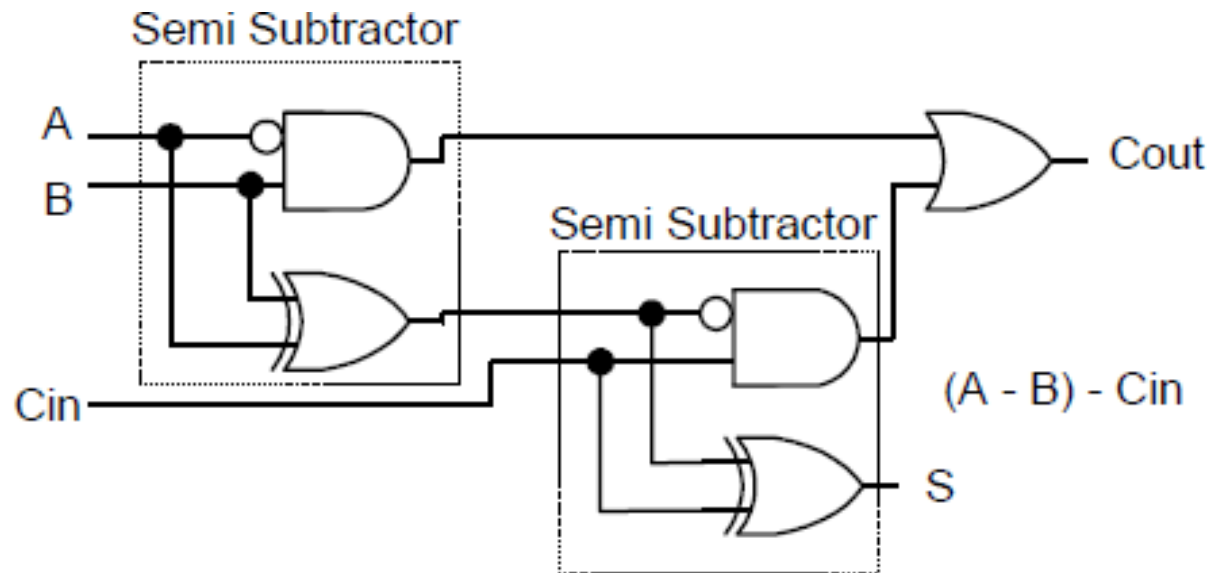


# 7. Combinational Circuits

## Full-Subtractor

---

- ♣ The full-subtractor subtracts two numbers of one bit and uses the borrow;
- ♣ It is built using two half-subtractors and has 3 inputs and 2 outputs, one of the inputs is the previous C in.

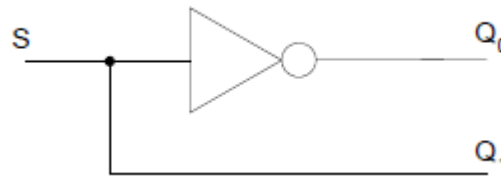


# 7. Combinational Circuits

## Decoder

---

- ♣ A decoder is a circuit that changes a code into a set of signals.
- ♣ A decoder selects one of its outputs depending on the binary combination on the input.
- ♣ The number of outputs is  $2^n$ , with the  $n$  being the number of bits.
- ♣ The most simple decoder has only one bit for selection and is formed only by an inverter (also called a line decoder).
- ♣ One bit input =  $2^1$  outputs.
- ♣ A typical application of a line decoder circuit is to select among multiple devices. A circuit needing to select among sixteen devices could have sixteen control lines to select which device should "listen". With a decoder only four control lines are needed.

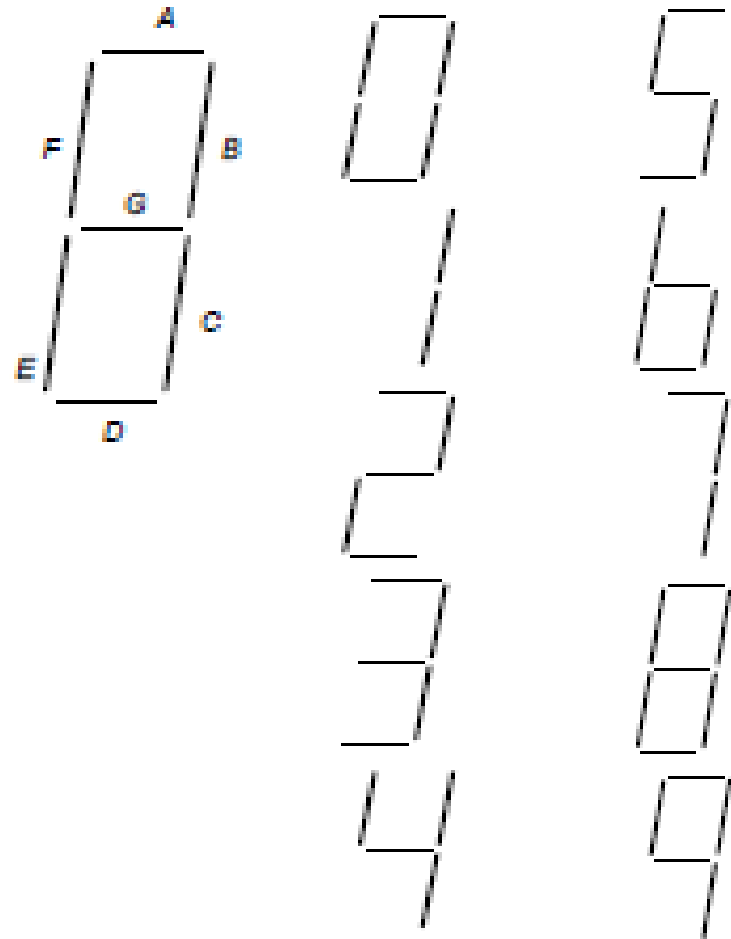


Decoder 1 bit selection		
S	Q <sub>0</sub>	Q <sub>1</sub>
0	1	0
1	0	1

# 7. Combinational Circuits

## Decoder – 7-digit display

BCD Inputs				LED Outputs						
D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



# 7. Combinational Circuits

## Encoder

- ♣ An encoder is a circuit that changes a set of signals into a code (opposite of the decoder).
- ♣ It's a combinational circuit with  $2^n$  inputs and n outputs (n is the number of bits), and when only one of the inputs is activated the outputs represent the order number of the activated input.

- ♣ To make a 2-to-1 line encoder truth table by reversing the 1-to-2 decoder truth table

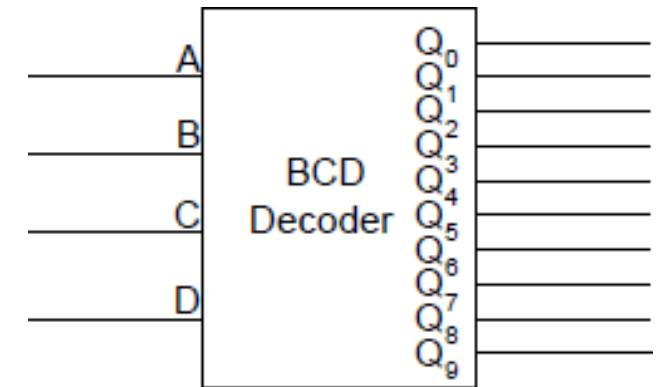
$D_1$	$D_0$	A
0	0	
0	1	0
1	0	1
1	1	

- ♣ One question we need to answer is what to do with those other inputs? Do we ignore them? Do we have them generate an additional error output? In many circuits this problem is solved by adding sequential logic in order to know not just what input is active but also which order the inputs became active.

# 7. Combinational Circuits

## Encoder – 7-digit display

BCD Encoder													
Outputs				Inputs									
D	C	B	A	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1



# 7. Combinational Circuits

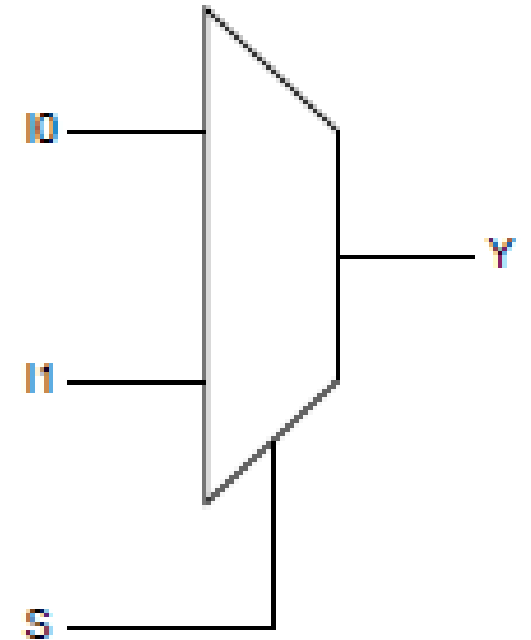
## Multiplexer

---

- ♣ A multiplexer, abbreviated **mux**, is a device that has multiple inputs and one output.
- ♣ The function of a multiplexer is to transmit through an output the information on one of many inputs.
- ♣ Its constituted by N lines of entry, one output and  $2^n$  selection lines.
- ♣ The schematic symbol for multiplexers is:

When  $S = 0$ , input  $I_0$  is connected to output  $Y$

When  $S = 1$ , input  $I_1$  is connected to output  $Y$

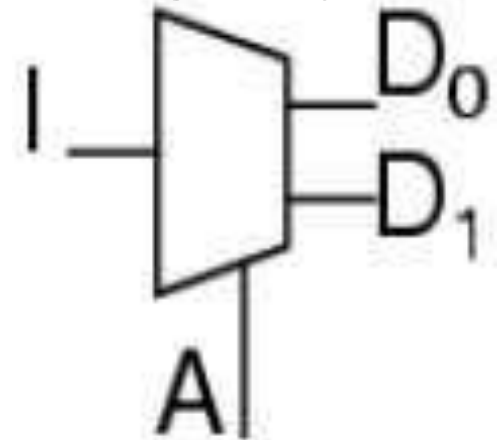


# 7. Combinational Circuits

## Demultiplexer

---

- ♣ A Demultiplexer, abbreviated **demux**, is a device that has one input and more than one output.
- ♣ It is used when a circuit wishes to send a signal to one of many devices. This description sounds similar to the description given for a decoder, but a decoder is used to select among many devices while a demultiplexer is used to send a signal among many devices.
- ♣ Its constituted by one input, N outputs and n selection lines.
- ♣ The schematic symbol for demultiplexer is:



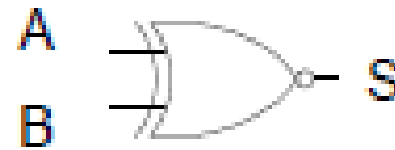
# 7. Combinational Circuits

## Comparator

---

♣ Comparators are combinational circuits that detect whether two words of  $n$  bits are equal, and if they are equal which one is higher or lower.

♣ The XNOR gate is an elementary comparator:



A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

♣ Considering the numbers 01 as “A” and 10 as “B”, we can immediately say that  $B > A$ .

♣ One example of comparator usage is a secure door in each we have to insert a code to enter.

♣ If we insert the wrong code, the bits won't match and the door will not be unlocked.

♣ If we push the keys in the correct order, the expected result will be matched and the door will unlocked.



# 7. Combinational Circuits

## Sequential Circuits

---

- ♣ In sequential logic circuits the output depends upon previous values of the input signals as well as their present-time values. This means that the previous output number (before the new input combination) is also taken in account.
- ♣ There are several types of sequential circuits:
  - Flip-Flop RS/ Flip-Flop JK/ Latch D/ RS Master Slave/ JK Master Slave/ Flip-Flop D/ Flip-Flop T
- ♣ The elementary sequential logic circuit is known as Flip-Flop.