# Chapter 8

# Graphical Language

## Introduction

- Machine languages are the first generation of programming languages.
- A machine language is a machine-dependent, low-level language that uses binary code to interact with a specific computer system. A machine-dependent language works only on a specific computer system and its components.
- A Machine Independent language is one that can run on any machine. An example of this would be Java.
- Machine Independent language can take the compiled code for any given machine and run it on the machine we are attempting to run it on. A TRULY machine-independent language would produce exactly the same output no matter which computer it was run on.

## Graphics Software

- Graphics software refers to a program or collection of programs that enable a person to manipulate images or models visually on a computer.
- There are two broad classifications for computer-graphics software: special-purpose packages and general programming packages.

1. **Special Purpose Graphics package**

   - Special-purpose packages are designed for nonprogrammers who want to generate pictures, graphs, or charts
   - in some application area without worrying about the graphics procedures that might be needed to produce such displays. The interface to a special-purpose package is typically a set of menus that allow users to communicate with the programs in their own terms. Examples of such applications include artists' painting programs and various architectural, business, medical, and engineering CAD systems.
   - There are many graphics software available in market; they can be categories as:

     1) **Paint program:** Paint program works with bit map images.

     2) **Photo manipulation program:** It works with bit map images and is widely used to edit digitized photographs.

     3) **Computer Aided Design program:** CAD software is used in technical design fields to create models of objects that will be built or manufactured. CAD software allows user to design objects in 3 dimensions and can produce 3-D frames and solid models.

     4) **3-D Modelling Programs:** It is used to create visual effects. 3-D modeling program works by creating objects like surface, solid, polygon etc.

     5) **Animation:** Computer are used to create animation for use in various fields, including games, and movies composing to allow game makers and film makers to add characters and objects to scenes that did not originally contain them.

2. **General Programming Package**

   - A general graphics programming package provides a library of graphics functions that can be used in a programming language such as C, C++, Java, or Fortran. Basic functions in a typical graphics library include those for specifying picture components (straight lines, polygons, spheres, and other objects), setting color values, selecting views of a scene, and applying rotations or other transformations. Some examples of general graphics programming packages are GL (Graphics Library), OpenGL, VRML (Virtual-Reality Modeling Language), Java 2D, and Java 3D.

- A set of graphics functions is often called a computer-graphics application programming interface (CG API) because the library provides a software interface between a programming language (such as C++) and the hardware.
- So when we write an application program in C++, the graphics routine allows us to construct and display a picture on an output device.

## Software Standard: Need of machine independent graphics language

- The primary goal of standardized graphics software is portability.
- When packages are designed with standard graphics functions, software can be moved easily from one hardware system to another and used in different implementations and applications.
- Without standards, programs designed for one hardware system often cannot be transferred to another system without extensive rewriting of the programs.

### General Kernel System (GKS)

- The Graphical Kernel System (GKS) was the first ISO standard for low-level computer graphics, introduced in 1977.
- The main purpose of GKS is the production and manipulation of 2D pictures in a way that does not depend on the system of graphical device used.
- In GKS, pictures are constructed from a number of basic building blocks.
- These basic building blocks are called as primitives.
- Some of the GKS primitives includes.
  - **i.** Polyline – draws sequence of connected lines
  - **ii.** Polymaker – marks a sequence of points with same symbol
  - **iii.** Fill Area – displays a specified area
  - **iv.** Text – draws a string of characters
  - **v.** Cell Array – displays an image composed of a variety of colors or gray scales.

### PHIGS (Programmer's Hierarchical Interactive Graphics System)

- PHIGS, short for the Programmer's Hierarchical Interactive Graphics System, is basically a library of about 400 functions that allow the user to display and interact with 2-D and 3-D graphics.
- It is an international standard, being created by the International Organization for Standardization (ISO).
- PHIGS hides hardware-dependent details from the user; so, for example, it allows an application draw on a plotter the same way it draws on a computer screen.
- PHIGS provides a set of familiar graphics objects called primitives, each with attributes that control its location, orientation, color, and appearance.
- Some of the PHIGS Primitives includes
  - **i.** Polyline: which draws a sequence of connected line segments;
  - **ii.** Polymarker: which marks a sequence of points with a symbol;
  - **iii.** Fill area: which defines the boundary of an area to be displayed;
  - **iv.** Fill area set: which defines the boundaries of a set of areas to be displayed as one;
  - **v.** Text: which draws a sequence of characters;
  - **vi.** Annotation text: which draws a sequence of characters to annotate a drawing;
  - **vii.** Cell array: which displays an image;

## Over View of Graphical File Formats

1. JPEG
   - Joint Photographic Experts Group) is a lossy compression method.
   - JPEG-compressed images are usually stored in the JFIF (JPEG File Interchange Format) file format.
   - The JPEG/JFIF filename extension is JPG or JPEG.

- Nearly every digital camera can save images in the JPEG/JFIF format, which supports eight-bit grayscale images and 24-bit color images (eight bits each for red, green, and blue).
- JPEG applies lossy compression to images, which can result in a significant reduction of the file size.

2. TIFF
   - The TIFF (Tagged Image File Format) format is a flexible format that normally saves eight bits or sixteen bits per color (red, green, blue) for 24-bit and 48-bit totals, respectively, usually using either the TIFF or TIF filename extension.
   - TIFF image format is not widely supported by web browsers.

3. GIF
   - GIF (Graphics Interchange Format) is in normal use limited to an 8-bit palette, or 256 colors (while 24-bit color depth is technically possible).
   - GIF is most suitable for storing graphics with few colors, such as simple diagrams, shapes, logos, and cartoon style images, as it uses LZW lossless compression, which is more effective when large areas have a single color, and less effective for photographic images.

4. BMP
   - The BMP file format (Windows bitmap) handles graphic files within the Microsoft Windows OS.
   - Typically, BMP files are uncompressed, and therefore large and lossless; their advantage is their simple structure and wide acceptance in Windows programs.

5. PNG
   - The PNG (Portable Network Graphics) file format was created as a free, open-source alternative to GIF.
   - The PNG file format supports eight-bit paletted images (with optional transparency for all palette colors) and 24-bit truecolor (16 million colors) or 48-bit truecolor with and without alpha channel - while GIF supports only 256 colors and a single transparent color.

## Data structure in Computer Graphics

- Triangle mesh: A triangle mesh is a type of polygon mesh in computer graphics. It comprises a set of triangles (typically in three dimensions) that are connected by their common edges or corners. The data structure representing the mesh provides support for two basic operations, inserting triangles and removing triangles.
- Quad-edge: A quad-edge data structure is a computer representation of the topology of a two-dimensional or three-dimensional map, that is, a graph drawn on a (closed) surface. It represents simultaneously both the map, its dual and mirror image
- Polygon Mesh: A polygon mesh is a collection of vertices, edges and faces that defines the shape of a polyhedral object in 3D computer graphics and solid modeling. The faces usually consist of triangles (triangle mesh), quadrilaterals, or other simple convex polygons, since this simplifies rendering, but may also be composed of more general concave polygons, or polygons with holes.
- Octree: An octree is a tree data structure in which each internal node has exactly eight children. Octrees are most often used to partition a three dimensional space by recursively subdividing it

## Open GL

- OpenGL is an API (Application programming interface) used for drawing 2D and 3D graphics.
- OpenGL is not a programming language.
- OpenGL is platform independent.
- An OpenGL application is typically written in C or C++.
- What OpenGL does allow you to do is draw attractive, realistic 2D and 3D graphics with minimal effort.
- OpenGL is a basic library of functions provided for specifying 2D and 3D graphics primitives(vertices), attributes(colors), geometric transformations, viewing transformations, and many other operations.

- OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn).
- We need the following sets of libraries in programming OpenGL:
    1. **Core OpenGL (GL)**: consists of hundreds of functions, which begin with a prefix "gl e.g., glColor, glVertex, glTranslate, glRotate etc.). The Core OpenGL models an object via a set of geometric primitives, such as point, line, and polygon.
    2. **OpenGL Utility Library (GLU):** built on-top of the core OpenGL to provide important utilities and more building models (such as quadratic surfaces). GLU functions start with a prefix "glu" (e.g., gluPerspective(), gluOrtho2D() etc.)
    3. **OpenGL Utilities Toolkit (GLUT):** provides support to interact with the Operating System (such as creating a window, handling key and mouse inputs); and more building models (such as sphere and torus). GLUT functions start with a prefix of "glut" (e.g., glutCreatewindow(), glutInitWindowSize(500,500)).

## Callback function

- GLUT supports a number of callbacks to respond to event like window repaint event, mouse clicked event, window resized event etc.
- A callback function is a function which the library (GLUT) calls when it needs to know how to process something.
- GLUT provides many callback functions, some of them are given below.
    1. When glut gets a window repaint event it uses **the glutDisplayFunc()** callback routine to find out what to do with a window repaint event.
    2. When glut gets a Key Down event it uses the **glutKeyboardFunc()** callback routine to find out what to do with a Key Down event.
    3. When glut gets a Mouse Clicked event it uses the **glutMouseFunc()** callback routine to find out what to do with a Mouse Button pressed event.
    4. When glut gets a window resized or moved event it uses **glutReshape()** callback routine to find out what to do with a window resized or moved event.

```
Void main(int argc, char** argv)
{
    ...
    glutDisplayFunc(display);
    ...
}
```

void display() – the function you provide. It contains all the OpenGL drawing function calls and will be called when pixels in the window need to be refreshed.

In the above example, display() is the callback function.

## Color Command

1. glClearColor(float Red, float Green, float Blue, float alpha)
    - The first three arguments in this function set the red, green, and blue component colors for the background color for the display window.
    - For example, If, 1.0, is set to each of these component, we get a White background. Similarly,0.0 is set then we get black background. We can vary these components color to get various background color.
    - The fourth parameter in the glClearColor function is called the alpha value for the specified color
    - Alpha values can be used to determine the resulting color for two overlapping objects.
    - An alpha value of 0.0 indicates a totally transparent object, and an alpha value of 1.0 indicates an opaque object.

      Example: glClearColor(1.0f, 1.0f, 1.0f, 0.0f);

2. glClear(GL_COLOR_BUFFER_BIT);
   - Although the glClearColor command assigns a color to the display window, it does not put the displaywindowon the screen. To get the assigned window color displayed, we need to invoke the glClear() OpenGL function
   - The argument GL COLOR BUFFER BIT is an OpenGL symbolic constant specifying that it is the bit values in the color buffer (refresh buffer) that are to be set to the values indicated in the glClearColor function.

3. glColor3f(float Red, float Green, float Blue);
   - This command set the color for object to be drawn. The suffix 3f on the glColor function indicates that we are specifying the three RGB color components using floating-point (f) values. This function requires that the values of red, green and blue be in the range from 0.0 to 1.0
   - Example:
     glColor3f(1.0f,1.0f,0.0f); // yellow color

## Header File

- In all of our graphics programs, we will need to include following the header file for the OpenGL core library.

**#include <windows.h>** // header file for accessing WGL library WGL or Wiggle is an API between OpenGL and the windowing system interface of Windows
**#include <GL/gl.h>**
**#include <GL/glu.h>**

- However, if we use GLUT to handle the window-managing operations, we do not need to include gl.h and glu.h because GLUT ensures that these will be included correctly. Thus, we can replace the header files for OpenGL and GLU with

**#include <GL/glut.h>**

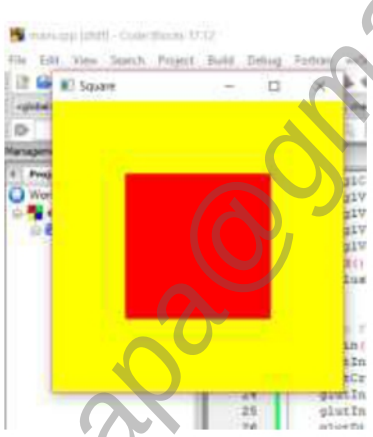
## OpenGL code to Draw a Quadrilateral

```
#include <windows.h> //windows specific header file
#include <GL/glut.h>

void display()
{
        glClearColor(1.0f, 1.0f, 0.0f, 0.0f);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0f, 0.0f, 0.0f);

        glBegin(GL_QUADS);
                glVertex2f(-0.5f, -0.5f);
                glVertex2f( 0.5f, -0.5f);
                glVertex2f( 0.5f,  0.5f);
                glVertex2f(-0.5f,  0.5f);
        glEnd();
        glFlush();
}

int main(int argc, char** argv)          ///argc=argument count and argv=argument vector
{
        glutInit(&argc, argv);
        glutCreateWindow("Square");
        glutInitWindowSize(320, 320);
        glutInitWindowPosition(350, 50);
        glutDisplayFunc(display); //Register callback handler for window re-paint event
        glutMainLoop();
        return 0;
}
```

In the above program

1. The function **display** () is the handler for window-repaint event. So this is **Call back** when the window first appears and whenever the window needs to be re-painted
2. The function glClearColor(1.0f, 1.0f, 0.0f, 0.0f); sets background color to yellow i.e. Red=1.0, Green=1.0 and Blue=0.0 and opacity=0.0 i.e. totally transparent
3. The function glClear(GL_COLOR_BUFFER_BIT) displays the previously assigned background window color.
4. The function glColor3f(1.0f, 0.0f, 0.0f); sets the red, green and blue component of the object to be drawn to 1.0, 0.0 and 0.0 respectively i.e. RED.
5. The function glBegin(GL_QUADS) and glEnd() is used to define set of vertices for drawing different primitive shape like quadrilateral, triangle, point, line, polygon etc.
6. The function glVertex2f(-0.5f, -0.5f); sets the X and Y coordinate of required vertex.
7. The function glFlush(); starts rendering of the object.
8. The function glutInit(&argc, argv); initialize the GLUT
9. The function glutCreateWindow("Square") creates a given window with given title.
10. The function glutInitWindowSize(320, 320); sets the window's initial width & height
11. The function glutInitWindowPosition(350, 50); sets the position the window's initial top-left corner
12. The function glutDisplayFunc(display); registers display **callback handler** for window re-paint
13. The function glutMainLoop(); puts the program into an infinite loop that checks for input from devices suchas a mouse or keyboard.

Output:

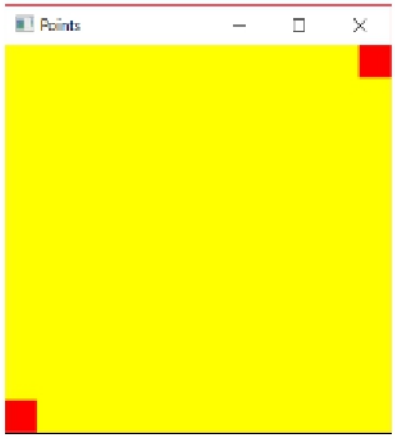



## OpenGL code to Draw a Point/Pixel

```
#include <windows.h> //windows specific header file
#include <GL/glut.h>

void display()
{
        glClearColor(1.0f, 1.0f, 0.0f, 0.0f);
        glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(50.0f); // point size
        glColor3f(1.0f, 0.0f, 0.0f);

        glBegin(GL_POINTS);
            glVertex2f(1.0f,1.0f);//upper-right corner
            glVertex2f(-1.0f,-1.0f);//lower-left corner


        glEnd();
        glFlush();
}

int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutCreateWindow("Points");
        glutInitWindowSize(320, 320);
        glutInitWindowPosition(350, 50);
        glutDisplayFunc(display);
```

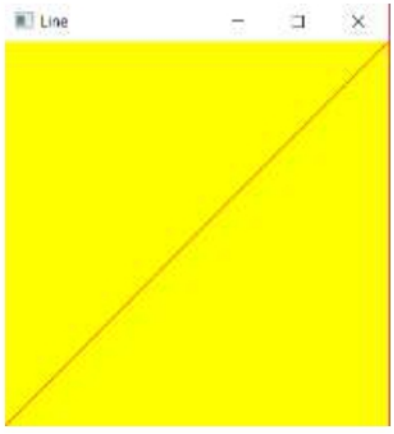
```
        glutMainLoop();
        return 0;
    }
```

Output:




## OpenGL code to Draw a Line

```
#include <windows.h> //windows specific header file
#include <GL/glut.h>

void display()
{
    glClearColor(1.0f, 1.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);

    glBegin(GL_LINES);
        glVertex2f(-1.0f,-1.0f);//lower-left corner
        glVertex2f(1.0f,1.0f);//upper-left cornor
    glEnd();
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutCreateWindow("Line");
    glutInitWindowSize(320, 320);
    glutInitWindowPosition(350, 50);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```



## OpenGL code to Draw a Polygon

```
#include <windows.h> //windows specific header file
#include <GL/glut.h>

void display()
{
    glClearColor(1.0f, 1.0f, 0.0f, 0.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);
    glBegin(GL_POLYGON);
        glVertex2f(-0.5f, 0.5f);//first vertex
        glVertex2f(0.5f, 0.5f);//second vertex
        glVertex2f(1.0f, 0.0f);//third vertex
        glVertex2f(0.5f, -0.5f);//fourth vertex
        glVertex2f(-0.5f ,-0.5f);//fifth vertex
        glVertex2f(-1.0f ,0.0f);//sixth vertex
    glEnd();
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutCreateWindow("Polygon");
    glutInitWindowSize(320, 320);
    glutInitWindowPosition(350, 50);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Output: