OpenGL

What is OpenGL?

- OpenGL is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics.

- The API is typically used to interact with a graphics processing unit, to achieve hardware-accelerated rendering.

Introduction

- Silicon Graphics, Inc. (SGI) began developing OpenGL in 1991 and released it on June 30, 1992.

- OpenGL is a hardware-independent, operating system independent API.

- The well-specified OpenGL standard has language bindings for C, C++, Fortran, Ada, and Java.

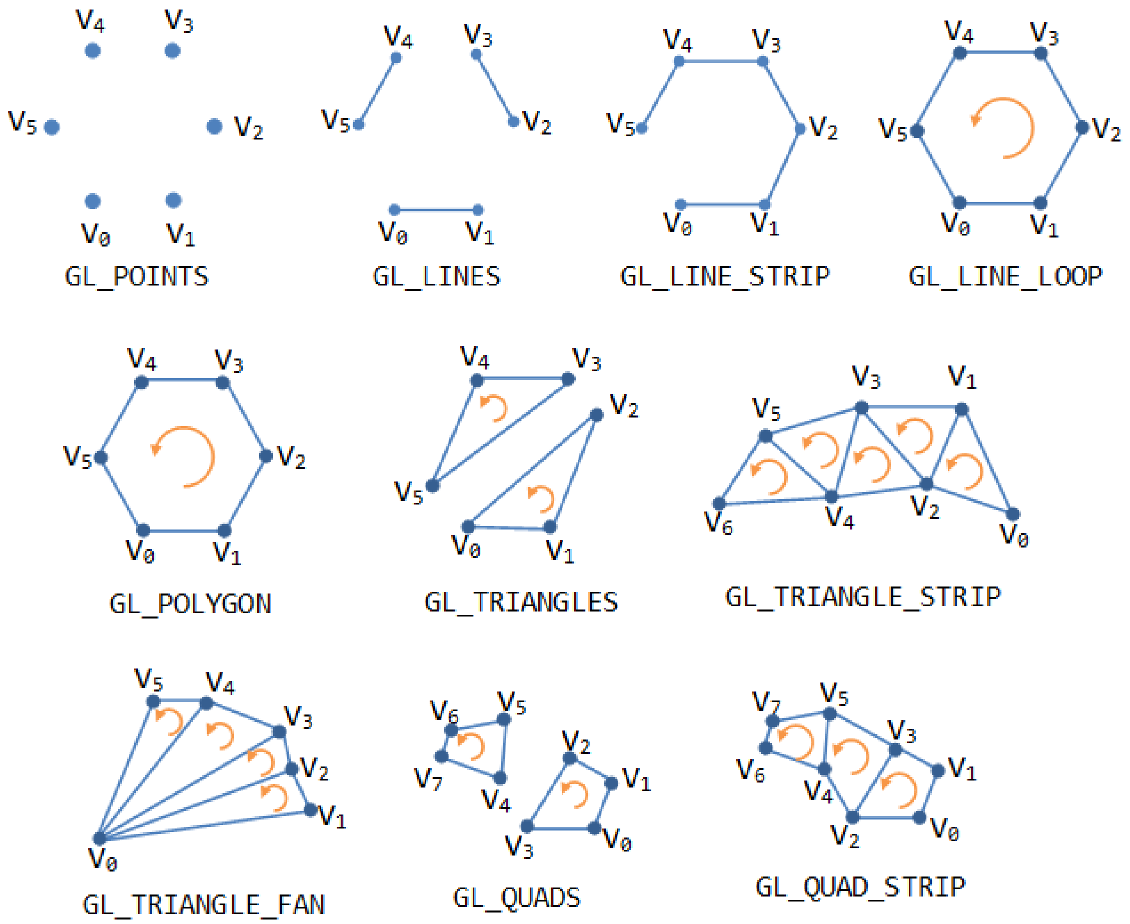
- OpenGL does not provide direct support for complex geometrical shapes, such as cubes or spheres. These must be built up from supported primitives.

OpenGL Primitives

- In OpenGL, an object is made up of geometric primitives such as triangle, quadrilateral, line segment and point.

- A primitive is made up of one or more vertices.

OpenGL Primitives



OpenGL Primitives

Programming OpenGL in C/C++

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). 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 qradric surfaces). GLU functions start with a prefix "glu" (e.g., gluLookAt, gluPerspective)

Programming OpenGL in C/C++

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, glutMouseFunc).

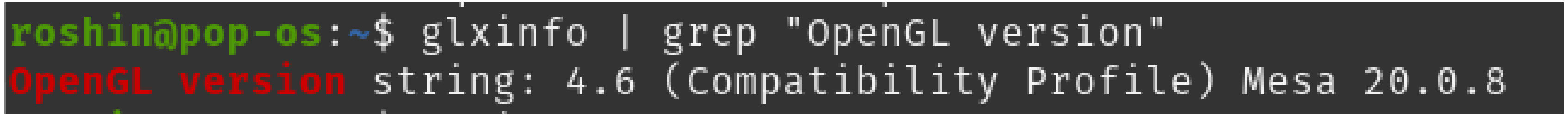
4. **OpenGL Extension Wrangler Library (GLEW**): GLEW is a cross-platform open-source C/C++ extension loading library. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform.

How to install OpenGL in Linux?

Open a terminal and execute the following commands

```
sudo apt-get update                     // To get information on the newest version of package and their dependencies
sudo apt-get install freeglut3          // Provides simple windowing API and I/O operations
sudo apt-get install freeglut3-dev      // header files for freeglut3
sudo apt-get install binutils-gold      // A linker for ELF files. Faster than GNU Linker.
sudo apt-get install g++ cmake          //Software tool for managing the build process of software
sudo apt-get install libglew-dev  //For determining which OpenGL extensions are supported on the platform
sudo apt-get install g++                //GNU C++ compiler
sudo apt-get install mesa-common-dev    //mesa is an OpenGL compatible 3D graphics library
sudo apt-get install build-essential    // All the packages needed to compile a debian package
sudo apt-get install libglew1.5-dev libglm-dev   //(glm) C++ mathematical library for graphics program.
sudo apt-get install mesa-utils         //provides several basic GL utilities. Ex: glxinfo, glxgears, etc.
```

Check the installation



```
roshin@pop-os:~$ glxinfo | grep "OpenGL version"
OpenGL version string: 4.6 (Compatibility Profile) Mesa 20.0.8
```

Sample codes

First OpenGL code

```
#include<GL/freeglut.h>
#include<GL/gl.h>

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - First window demo");
    glutMainLoop();
    return 0;
}
```

Initializations

glutInit

```
void glutInit(int *argcp, char **argv);
```

`glutInit` initializes the GLUT library and negotiate a session with the window system.

```
#include<GL/freeglut.h>
#include<GL/gl.h>
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - First window demo");
    glutMainLoop();
    return 0; }
```

glutInitDisplayMode

```
void glutInitDisplayMode(unsigned int mode);
```

*This can be used to select the features we

would want a window to have.

*It can be the color system we are using,

the frame buffers needed etc.

Ex:- `glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE)`

```
#include<GL/freeglut.h>
#include<GL/gl.h>
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|
GLUT_RGBA);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - First
window demo");
    glutMainLoop();
    return 0; }
```

glutInitWindowSize

```
void glutInitWindowSize(int width, int height);
```

The intent of the initial window position

and size values is to provide a suggestion

to the window system for a window's initial size.

```
#include<GL/freeglut.h>
#include<GL/gl.h>
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE| GLUT_RGBA);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - First window demo");
    glutMainLoop();
    return 0; }
```

glutInitWindowPosition

```
void glutInitWindowPosition(int x, int y);
```

glutInitWindowSize set the initial window position.

```
#include<GL/freeglut.h>
#include<GL/gl.h>
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE| GLUT_RGBA);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - First window demo");
    glutMainLoop();
    return 0; }
```

glutCreateWindow

```
void glutCreateWindow(char *name);
```

The parameter will be used to set the window name.

`glutCreateWindow` creates a top-level window.

```
#include<GL/freeglut.h>
#include<GL/gl.h>
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE| GLUT_RGBA);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - First window demo");
    glutMainLoop();
```

glutMainLoop

```
void glutMainLoop(void);
```

`glutMainLoop` enters the GLUT event processing loop.

Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

```
#include<GL/freeglut.h>
#include<GL/gl.h>
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - First window demo");
    glutMainLoop();
    return 0; }
```

Compilation

```
g++ main.c -lglut -lGL -lGLEW -lGLU -o OpenGLExample

            /*    -lglut: Link with glut

                   -lGL: Link with GL

                        -lGLEW: Link with GLEW

                        -lGLU: Link with GLU

                   */
```

Run the executable file

```
./OpenGLExample
```

Drawing in the window

Draw in the window

```
#include<GL/freeglut.h>
#include<GL/gl.h>

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGBA);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - First window demo");

    glBegin(GL_POLYGON);
       glVertex2f(-0.5,-0.5);
       glVertex2f(-0.5,0.5);
       glVertex2f(0.5,0.5);
       glVertex2f(0.5,-0.5);
    glEnd();

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

glBegin, glEnd

glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives. glBegin accepts a single argument that specifies in which of ten ways the vertices are interpreted.

glColor3f

Sets the current color of the pen

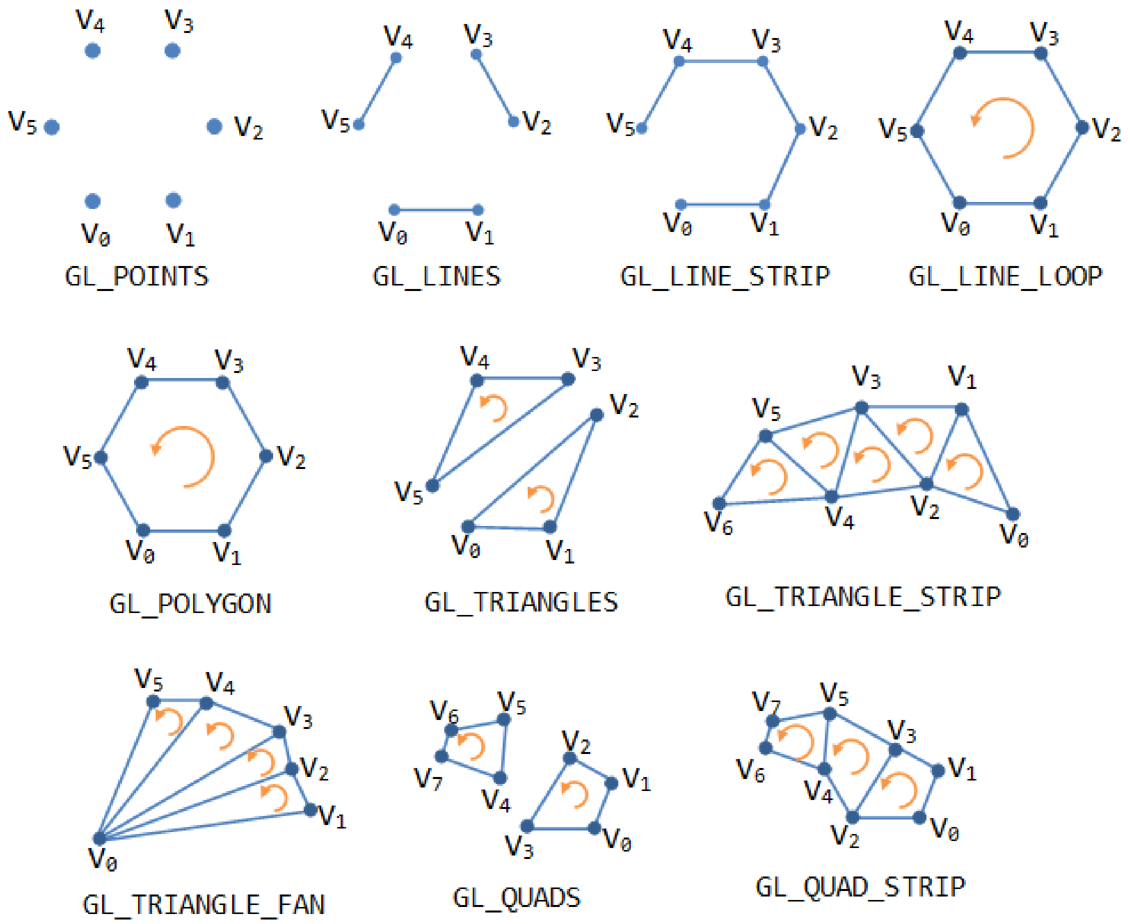
Syntax

```
void glColor3f(GLfloat red, GLfloat green, GLfloat blue);
```

Usage

```
glColor3f(1.0, 1.0, 1.0);
```

OpenGL Primitives



OpenGL Primitives

1. Display a set of 7 clearly visibile points : 4 points just near the corners of the OpenGL window and three points distributed far apart from each other in the interior of the window.
2. Display 3 line segments of different length and different color.
3. Display a polygon of 6 vertices with at least two reflex vertces and color the whole polygon
   with Red color.
4. Display a polygon of 8 vertices with at least two reflex vertces and color the polygon boundary with Red color and polygon interior with green color.

References

[1].*https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi4ydrc7LHsAhU94HMBHViVDxcQFjADegQIBRAC&url=https%3A%2F%2Fwww.cs.utexas.edu%2Fusers%2Ffussell%2Fcourses%2Fcs354%2Fhandouts%2FAddison.Wesley.OpenGL.Programming.Guide.8th.Edition.Mar.2013.ISBN.0321773039.pdf&usg=AOvVaw0eSL-A754ij_wV_sq03JvS*

[2].*OpenGL Architecture Review Board, OpenGL Reference Manual: The Official Reference Document for OpenGL, Release 1, Addison-Wesley, Reading, Massachusetts, 1992 (ISBN 0-201-63276-4).*

[3].*http://www.cs.toronto.edu/~kyros/courses/418/Notes/Visibility.pdf*

[4].*https://www.youtube.com/watch?v=pQcC2CqReSA https://flylib.com/books/en/2.789.1.32/1/*

[5].*http://www.opengl-tutorial.org/beginners-tutorials/tutorial-1-opening-a-window/*

Thank you