



Lecture 5

My first Open GL source code

What is OpenGL

- GL: stands for Graphic Library
- Software interface for rendering purposes for 2D or 3D geometric data objects.

Required

- Knowledge in C\C++
- Dev C++ : a freeware for C/C++
<http://www.bloodshed.net/devcpp.html>
- Glut Dev Pack

Setting OpenGL Dev Pack

1. To use Open GL install the correct version of glut DevPack go to <http://www.nigels.com/glt/devpak>: download glut.3.7.6+.DevPak to a local folder
2. In Dev C++ open the package manager: Tools→Package Manager
3. Install the glut package: Package→Install Package
4. Browse to the local location where the package was downloaded to follow the installation wizard
5. The glut package should now appear in the main pane
6. Exit package manager

Setting Project for OpenGL

1. To compile, build and run using Dev C++: make a new project: File → New → Project and choose empty project and C++ project is selected
2. Create the new project in the folder with the source file
3. Add the .cpp file to the project: Project → add to project: select the .cpp file
4. Define the linker options: Project → Project options
5. In the Parameters tab window add the following line in the Linker pane: *-lglut32 -lglu32 -lopengl32 -lwinmm -lgdi32*
6. click ok
7. Compile and run the project:
8. Execute → Compile & run

Various Pieces

- **gl**: The basic libraries.
- **glu**: Advanced implementation of OpenGL. Higher level function calls/commands. i.e. gluSphere.
- **glut**: Generic platform independent library.

Reference/Source Code

Nehe's OpenGL Tutorials

<http://nehe.gamedev.net/>

GLUT Programming Interface:

The OpenGL Utility ToolKit (GLUT) Programming Interface

GL Programming Interface

<http://www.glprogramming.com/blue/> Blue Book

Hello world.cpp

```
#include <gl/glut.h>

void display( )
{
    // nothing to see here, please move along
}

int main(int argc, char* argv[])
{

    glutInit( &argc, argv );
    glutInitDisplayMode( GLUT_SINGLE | GLUT_RGB );
    glutInitWindowPosition( 50, 50 );
    glutInitWindowSize( 200, 200 );
    glutCreateWindow( "Hello World" );
    glutDisplayFunc( display );
    glutMainLoop( );

    return 0;
}
```

Functions

void glutInit()

to initialize the GLUT library, the variables should be the same in main()

void glutInitDisplayMode(),

void glutInitWindowSize(),

void glutWindowPosition()

define the type of window, its size, and its position (detail in OpenGL Utility Toolkit)

Functions cont

```
int glutCreateWindow("Hello World")
```

creates a window on the screen with the given title. This function returns an integer that can be used to refer to the window in multiwindow environments

```
void glutMainLoop()
```

causes the program to enter an event-processing loop. Should be in the end in the main function

Display Function

```
void display( )
{
    // Clear the background before drawing
    glClear( GL_COLOR_BUFFER_BIT );

    // Actual code to do the drawing
    glBegin( GL_POLYGON );
        glVertex2f( -0.5, -0.5);
        glVertex2f( -0.5, 0.5);
        glVertex2f( 0.5, 0.5);
        glVertex2f( 0.5, -0.5);
    glEnd();

    // Ensure all drawing commands are executed
    glFlush( );
}
```

Procedure to draw a rectangle

Clearing the window

```
glClear(GL_COLOR_BUFFER_BIT);
```

Drawing the rectangle

Must starts with glBegin and ends with glEnd. Drawing the rectangle must be in between them

Flushing GL buffers

Force to render or display the rectangle on the windows using glFlush functions

```
glFlush();
```

What is entity, geometry and attributes

- ◆ Entity: lines, polygon, triangle
- ◆ Geometry: coordinates of the vertices
- ◆ Attribute: color, thickness

Data Types

character	C-language type	OpenGL type definition
b	signed char	GLbyte
s	short	GLshort
i	int	GLint, GLsizei
f	float	GLfloat, GLclampf
d	double	GLdouble, GLclampd
ub	unsigned char	GLubyte, GLboolean
us	unsigned short	GLushort
ui	unsigned int	GLuint, GLenum, GLbitfield
	void	GLvoid

Coordinate System in OpenGL

```
void glVertex{234}{sifd}(TYPE x, TYPE y, ...)  
void glVertex{234}{sifd}v(TYPE *v)
```

*specifies the location in space of the vertex with type of coordinate.
If v presence, the coordinate must be declared as array*

```
glVertex2f(2.0, 2.0);  
glVertex2fv(p); // p[0] = 2.0, p[1] = 2.0
```

The 4th coord is homogenous specifically used during transformation

CS in OpenGL and Window

`glutInitWindowSize()` sets the windows size in pixels.

CS in OpenGL is according to Cartesian CS in plane XY. By default the origin is in the center of the window and bottom left is (-1,-1) and top right is (1,1).

200 x 200 pixel and 500 x 500 pixel will have the same bottom left (-1,-1) and top right (1,1)

`gluOrtho2D` sets the new bottom left and top right point.

`gluOrtho2D(GLdouble minx, GLdouble maxx, GLdouble miny, GLdouble maxy)` will set the new clipping area. This is place after `glutDisplayFunc()` function

```
glutDisplayFunc( display );  
gluOrtho2D(-2.0, 2.0, -2.0, 2.0);
```

Color in OpenGL

```
void glColor3{bsifd ubusui}{v} (TYPE r, TYPE g, TYPE b);  
void glColor4{bsifd ubusui}{v} (TYPE r, TYPE g, TYPE b, TYPE a);
```

a is alpha, to measure opacity; 1.0 is opaque and 0.0 is transparent

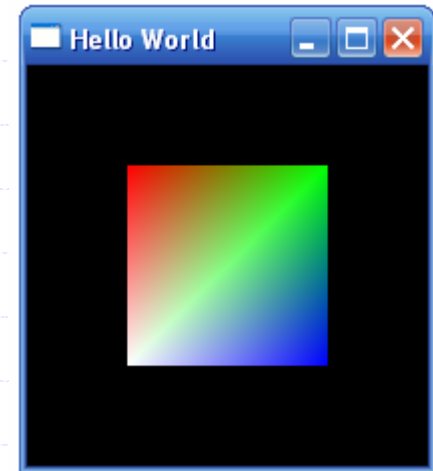
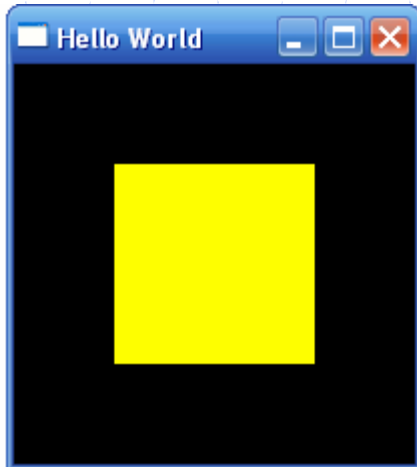
(1.0, 1.0, 1.0): white

(1.0, 0.0, 0.0): bright red

Coloring the polygon

```
glColor3f(1.0,1.0,0.0);
```

```
// Actual code to do the drawing  
glBegin( GL_POLYGON );  
    glVertex2f( -0.5, -0.5);  
    glVertex2f( -0.5, 0.5);  
    glVertex2f( 0.5, 0.5);  
    glVertex2f( 0.5, -0.5);  
glEnd();
```



```
glBegin( GL_POLYGON );  
    glColor3f(1.0,1.0,1.0);  
    glVertex2f( -0.5, -0.5);  
    glColor3f(1.0,0.0,0.0);  
    glVertex2f( -0.5, 0.5);  
    glColor3f(0.0,1.0,0.0);  
    glVertex2f( 0.5, 0.5);  
    glColor3f(0.0,0.0,1.0);  
    glVertex2f( 0.5, -0.5);  
glEnd();
```

Thickness Attribute

Point: the size of the point

```
void glPointSize(GLfloat pixel_size )
```

```
    glPointSize( 5.0 );
```

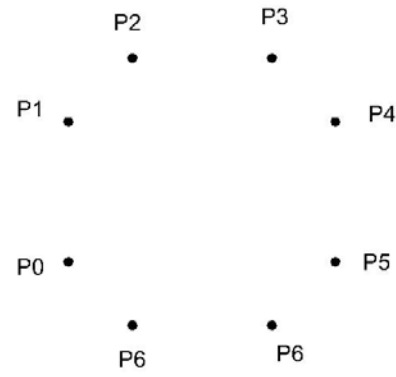
*this function must be before glBegin

Line: the width of the line

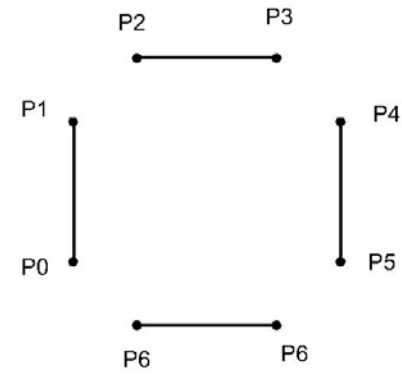
```
void glLineWidth(GLfloat pixel_width)
```

```
    glLineWidth( 5.0 );
```

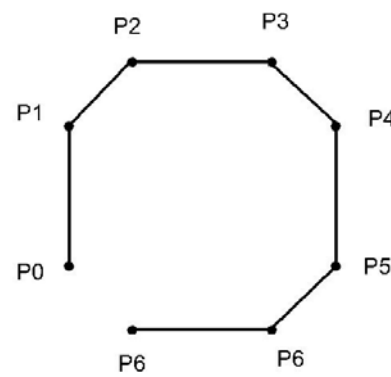
Primitive Points and Lines



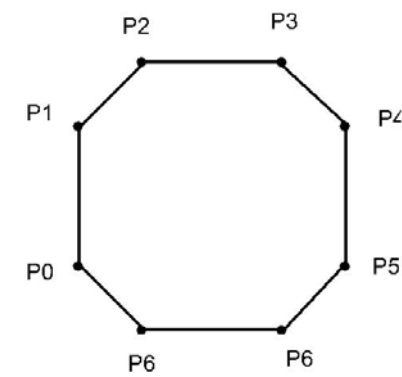
GL_POINTS



GL_LINES

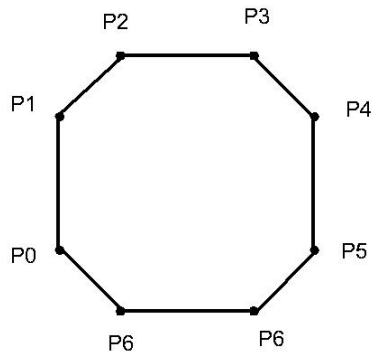


GL_LINE_STRIP

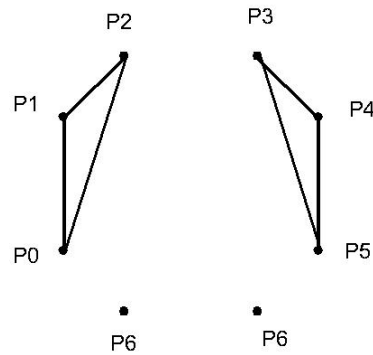


GL_LINE_LOOP

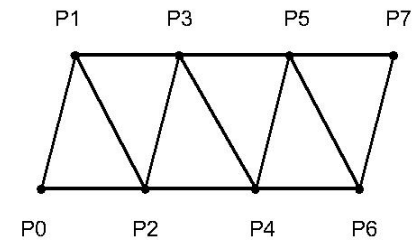
Filled Primitives



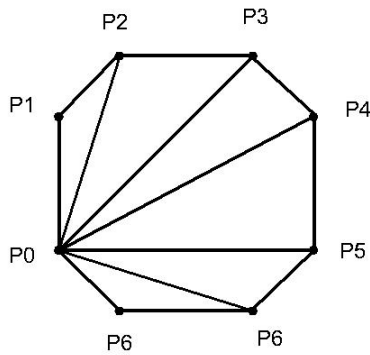
GL_POLYGON



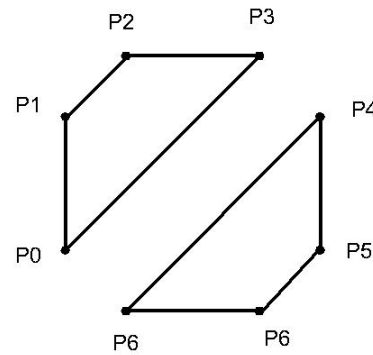
GL_TRIANGLES



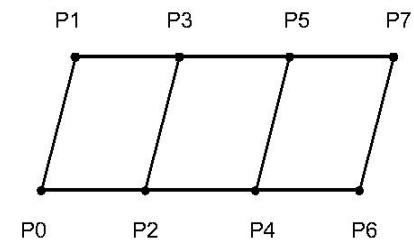
GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP

Window related functions

void glutDestroyWindow(int id)

Destroys top-level window id

void glutSetWindow(int id)

Set the current to the window with identifier id

**int glutCreateSubWindow(int parent, int x, int y,
int width, int height)**

Create subwindows to specified id windows

void glutPostWindowRedisplay(int winid)

Post a redisplay for window winid

Example

Circle can be drawn using connected lines. The lines are drawn using the following formula

$$x = r \cos (\theta) \quad y = r \sin (\theta)$$

As the angle gets smaller, the connected lines will form a circle. Therefore, the task is to draw a circle using this angle $\theta = 10^\circ$.

