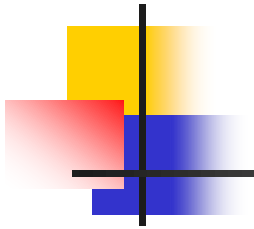
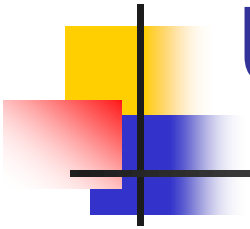


Tutorial 1
for the Course Project of
CS4182/CS5182 Computer Graphics



Useful Resources

- **Refence links**

  http://www.opengl.org/

  http://dindinx.net/OpenGL/Introduction/

  http://nehe.gamedev.net/lesson.asp?index=01

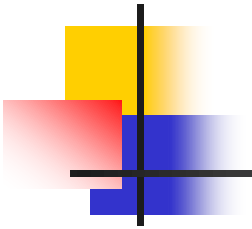
  http://www.yakergong.net/nehe/ (Chinese version)

  http://www.cse.msu.edu/~cse872/

  http://www.swiftless.com/tutorials/opengl/opengltuts.html

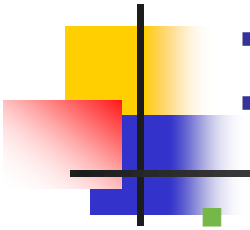
- **Reference Book**

  OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2, OpenGL Architecture Review Board, Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis, Addison-Wesley Professional, 2005



Outline

1. **Introduction to OpenGL**
2. Start OpenGL Program
3. OpenGL Functions
   - Primitives & Primitive Attributes
   - Light & Shading
   - Viewing: Camera & Projection
   - Transformations
   - Texture Mapping
   - Interaction: Keyboard, Mouse and Menus
4. Assignment



Introduction to OpenGL

- *What is **OpenGL**?*

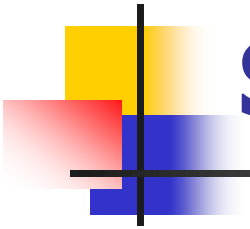
  OpenGL (Open Graphics Library) is the premier environment for developing portable, interactive 2D and 3D graphics applications.

- *What is **GLUT**?*

  The OpenGL Utility Toolkit (GLUT) is a library used to simplify the process of creating an OpenGL window as well as the process of capturing keyboard and mouse input.

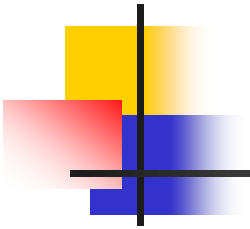
- *Why OpenGL?*

  - The industry's most widely used and supported 2D and 3D graphics application programming interface (API). (The two most commonly used graphics APIs are OpenGL and Microsoft's **DirectX**.)
  - Platform independent.
  - Easy to use



Some OpenGL Demos

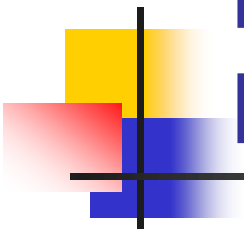
- Demo1 - Bouncing Box
- Demo2 - Super Stars
- Demo3 - Halloween



Outline

1. Introduction to OpenGL
2. **Start OpenGL Program**
3. OpenGL Functions
   - Primitives & Primitive Attributes
   - Light & Shading
   - Viewing: Camera & Projection
   - Transformations
   - Texture Mapping
   - Interaction: Keyboard, Mouse and Menus
4. Assignment

Basic Structure of OpenGL Programs



An OpenGL program usually contains the following functions:

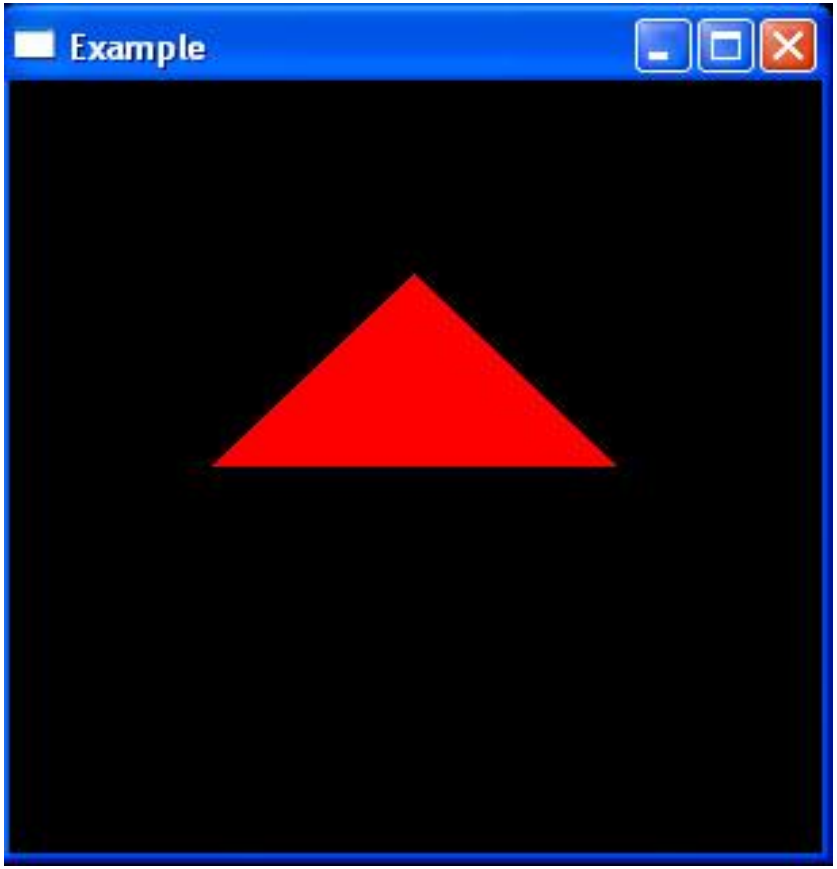
- init ():
  - Sets the state variables
- main ():
  - Open one or more windows
  - Defines callback functions
  - Enter event loop
- Callback Functions:
  - Display function
  - Input functions
  - …

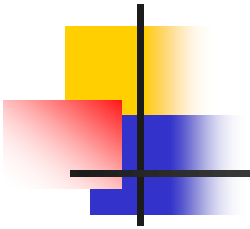
Example 1- OpenGLProgramStructure

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

void init()
{
        glClearColor(0.0, 0.0, 0.0, 0.0); // background color
        glColor3f(1.0, 0, 0);  // foreground color
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
}
void mydisplay()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glBegin(GL_POLYGON);
                glVertex2f(0, 0.5);
                glVertex2f(0.5, 0);
                glVertex2f(-0.5, 0);
        glEnd();
        glFlush();
}
void main (int argc, char ** argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutCreateWindow("Example"); // open a window named "Example"
        glutDisplayFunc(mydisplay);  // display callback
        init(); // set OpenGL state
        glutMainLoop(); // enter event loop
}
```

Example 1- OpenGLProgramStructure

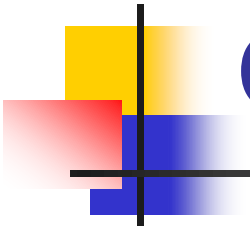
Running result:





Outline

1. Introduction to OpenGL
2. Start OpenGL Program
3. **OpenGL Functions**
   - **Primitives & Primitive Attributes**
   - Light & Shading
   - Viewing: Camera & Projection
   - Transformations
   - Texture Mapping
   - Interaction: Keyboard, Mouse and Menus
4. Assignment



OpenGL Functions

- There are **seven** major groups of functions:
  - Primitive Functions
  - Attribute Functions
  - Viewing Functions
  - Transformation Functions
  - Input Functions
  - Control Functions
  - Query Functions

Index of some OpenGL functions: http://dindinx.net/OpenGL/Introduction/

Primitive Functions –Primitives

- Basic graphs include: simple points, lines, shapes, polygons, etc.
- Each primitive is made up of vertices.
- To create primitives, you need to make a call to the function ***glBegin()***. It accepts one parameter indicating what primitives are going to be created.
- When all vertices have been specified, the function ***glEnd()*** should be called at the end.

```
glBegin (……);
      …… // vertices
glEnd();
```

Primitive Functions –Primitives

- **Points**: each vertex specifies a point.

```
glBegin(GL_POINTS);
    glVertex2f(-0.3, 0.5);
glEnd();
```

- **Lines**: Each pair of vertices creates a line.

```
glBegin(GL_LINES);
    glVertex2f(0.5, 0.5);
    glVertex2f(0, 0.3);
glEnd();
```

- **Polygons**: vertices are connected in sequence to form an polygon.
  - Edges cannot cross
  - All vertices are in the same plane
  - Convex polygon, not concave

```
glBegin(GL_POLYGON);
    glVertex2f(-0.3, 0.5);
    glVertex2f(0, 0.8);
    glVertex2f(0.3, 0.5);
    glVertex2f(0.15, 0.1);
    glVertex2f(-0.15, 0.1);
glEnd();
```

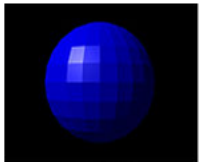
Primitive Functions -3D Primitives

- **Polygons**

```
glBegin(GL_POLYGON);
    glVertex3f(-0.3,0,0.2 );
    glVertex3f(0.3, 0, -0.2);
    glVertex3f(0, 0.7, 0);
glEnd();
```

- **Cube**
  - Function: draw a cube with the size of a×a×a and its center locates at the origin of the coordinate.

glutSolidCube (double a);

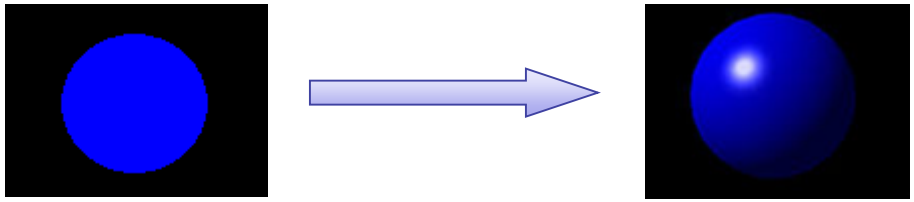


- **Sphere**
  - Function: draw a sphere with its center locates at the origin of the coordinate.

glutSolidSphere (double radius, int slices, int stacks);

  - radius: radius of the sphere, slices: number of subdivisions around Z axi, stacks: number of subdivisions along Z axis
- Others: cone, teapot , etc.

Atribute Functions

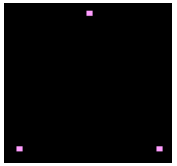


- Material properties
- Light sources
- Location of viewer
- Surface orientation

Attribute Functions -Size

- **Point Size**
  - Get the current size of points:

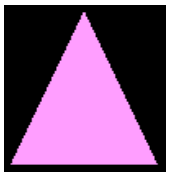
```
float size=0;
glGetFloatv(GL_POINT_SIZE,&size);
```



  - Specify the point size—the diameter of the point:

```
glPointSize(2.5);
glBegin(GL_POINTS);
    glVertex2f(-0.2,0.3);
glEnd();
```

- **Line Width**
  - glGetFloatv(GL_LINE_WIDTH, &width);
  - glLineWidth(width)



- **Polygon Mode**: Polygons can either be rendered as points, lines or filled.
  - glGetIntegerv(GL_POLYGON_MODE, mode);
  - glPolygonMode(GL_FRONT_AND_BACK, GL_FILL/ GL_POINT/GL_LINE));

Attribute Functions -Color

- **Color**
  - Background:

    glClearColor (float red, float green, float blue, float alpha)

    - red, green, blue: the RGB value of the color
    - alpha: transparence (how transparent a color is)
  - Foreground:

    glColor3f (float red, float green, float blue)
  - Clean the whole scene: it is usually called before creating a new scene

    glClear (GL_COLOR_BUFFER_BIT);

Attribute Functions -Material Properties

In OpenGL, the function ***glMaterial()*** is used to define the material parameters for the lighting model.

glMaterialfv (Glenum face, Glenum name, float* para)

- **face:** specify which face or faces are being defined.
  (take value from "GL_FRONT ", "GL_BACK ", "GL_FRONT_AND_BACK");
- **name:** specify which property is being defined.
  (take value from GL_AMBIENT,/ GL_DIFFUSE / GL_SPECULAR / GL_AMBIENT_AND_DIFFUSE / …);
- **para:** specify the parameters of the property

Example:

```
GLfloat pl_ambient[]={0.05,0.05,0.05,1};
GLfloat pl_diffuse[]={0.8,0.8,0.8,1};
GLfloat pl_specular[]={0.6,0.6,0.6,1};
glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,pl_ambient);
glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,pl_diffuse);
glMaterialfv(GL_FRONT_AND_BACK,GL_SPECULAR,pl_specular);
```

Attribute Functions -Light Properties

- Position of light:

```
GLfloat LightPosition[]= {float* para1, float* para2, float* para3, float* para4}
```

- Properties:
  - Ambient light, diffuse reflection, and specular reflection.
  - In OpenGL, the function ***glLight()*** is used to describe the light source

```
glLightfv (Glenum light, Glenum property_name, float* para)
```

    - light: specify a light, should be in the form of 'GL_LIGHT*i*', where i ranges from 0 to GL_MAX_LIGHTS-1.
    - property_name: specify which property is being defined. (GL_POSITION / GL_AMBIENT / GL_SPECULAR / GL_SPOT_DIRECTION / …)
    - para: specify the parameters of the property

- Enable the light:

```
glEnable (LIGHTING); // start using light sources
```

```
glEnable (light_name); // let the specified light be useable
```

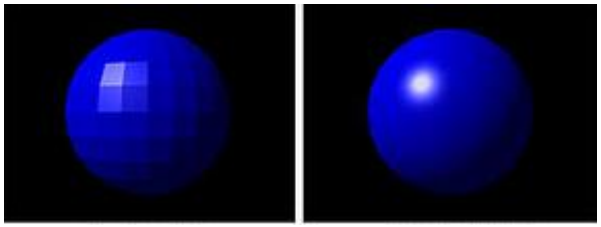
Attribute Functions -Light Properties

- Example

```
//light source
GLfloat light_position[]={0.0, 1.0, 0.0, 0.0};
GLfloat ambient[]={0.1, 0.1, 0.1, 1.0};
GLfloat diffuse[]={1.0, 1.0, 1.0, 1.0};
GLfloat specular[]={1.0,0.6,0.6,1.0};

glLightfv(GL_LIGHT1,GL_POSITION,light_position);
glLightfv(GL_LIGHT1,GL_AMBIENT,ambient);
glLightfv(GL_LIGHT1,GL_DIFFUSE,diffuse);
glLightfv(GL_LIGHT1,GL_SPECULAR,specular);

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT1);
```

Attribute Functions -Shading

- ***Shading***: a method in computer graphics to simulate the differing effects of light and color across the surface of an object.
- Two shading modes:
  - **Flat shading**: the color of the first vertex determines the fill color of the slice
  - **Smooth shading**: colors are interpolated between the vertices where each vertex can be given a different dolor. It is the default shading mode in OpenGL.

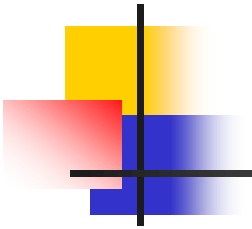


- Shading mode is specified by:
  - `glShadeModel(GL_SMOOTH);` or `glShadeModel(GL_FLAT);`

Example 2 – Tea Pot

```
#include <GL/glut.h>
void Init () { //setting up the environment
    GLfloat light_position[]={0,50,-100,1};
    GLfloat ambient[]={0.2,0.2,0.2,1};
    GLfloat diffuse[]={0.8,0.8,0.8,1};
    GLfloat specular[]={1,0.6,0.6,1};
    glLightfv(GL_LIGHT1,GL_POSITION,light_position);
    glLightfv(GL_LIGHT1,GL_AMBIENT,ambient);
    glLightfv(GL_LIGHT1,GL_DIFFUSE,diffuse);
    glLightfv(GL_LIGHT1,GL_SPECULAR,specular);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT1);

    glClearColor(0.3,0.4,0.8,0.1);
    glShadeModel(GL_SMOOTH);
}

void Display () { //create objects
    glClear(GL_COLOR_BUFFER_BIT);

    GLfloat tp_ambient[]={0.05,0.05,0.05,1};
    GLfloat tp_diffuse[]={0.7,0.3,1,1};
    GLfloat tp_specular[]={0.6,0.6,0.6,1};
    glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,tp_ambient);
    glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,tp_diffuse);
    glMaterialfv(GL_FRONT_AND_BACK,GL_SPECULAR,tp_specular);

    glutSolidTeapot(0.2);

    glFlush();
}

void main (int argc, char** argv) {
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

    glutInitWindowSize (800,600);
    glutInitWindowPosition (150,100);
    glutCreateWindow ("Example");

    glutDisplayFunc(Display);
    glutMainLoop();
}
```

Example 2 – Tea Pot

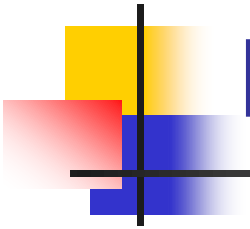


Outline

1. Introduction to OpenGL
2. Start OpenGL Program
3. OpenGL Functions
   - Primitives & Primitive Attributes
   - Light & Shading
   - Viewing: Camera & Projection
   - Transformations
   - Texture Mapping
   - Interaction: Keyboard, Mouse and Menus
4. **Assignment**

Assignment Program

- Now you can run the program, the running result is a 3D Room:



Requirements

Basic requirements:

- Creating new objects
- Manipulating objects
- Manipulating camera
- Window resolution
- Adding autonomous objects

Advanced requirements:

- Extend the program into an application
- …

Note: For the requirement details, please refer to the assignment description document.