OpenGL Introduction

- OpenGL Reference:
  - OpenGL Programming Guide (Addison-Wesley) is available online. Web site -> Projects -> OpenGL
- OpenGL program examples:
  - Course web site -> Projects -> OpenGL
- Compiling:

  (-lglut) -lGLu -lGL - lXmu -lXext -lX11 -lm

  Example makefile is found on the course web site.

  Also, need the **libglut.a** file.

OpenGL Introduction

- Include files:

`#include <GL/gl.h>`	OpenGL library
`#include <GL/glu.h>`	OpenGL utility library
`#include <GL/glx.h>`	OpenGL ext. to X-Windows
`#include <GL/glut.h>`	OpenGL Utility Toolkit

(`glut.h` also includes `gl.h, glu.h and glx.h`)

OpenGL Command Syntax

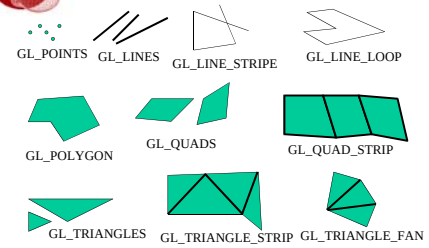
- Uses the prefix **gl**
- Capital letters for each word making up the command name. E.g. **glClearColor()**
- The prefix **glu** is used when the OpenGL command is from the utility library.
- The prefix **glX** is used when the OpenGL command is from the X-window system.
- Constants begin with **GL_** . And is all written in capital letters. E.g. **GL_COLOR_BUFFER_BIT.**
- Root names are **glColor()** and **glVertex().**

Drawing Geometric Primitives

- **glVertex{2,3,4}{s,i,f,d}[v]()** can be used to create a set of points, lines or a filled polygon.
- Example:

```
glBegin(GL_POLYGON);
    glVertex2f(0.0,0.0);
    glVertex2f(0.0,3.0);
    glVertex2f(4.0,3.0);
    glVertex2f(6.0,1.5);
    glVertex2f(4.0,0.0);
glEnd();
```

OpenGL Primitives



Specifying Colors

glColor3f(1.0,0.0,0.0); (Red)

Is the same as

float red[3] = {1.0,0.0,0.0};

glColor3fv(red);

v means pointer to a vector

Clearing Buffers

glClearColor(0.0,0.0,0.0,0.0); (RGB mode)
glClearDepth(0.0);
glClear(**GL_COLOR_BUFFER_BIT** | **GL_DEPTH_BUFFER_BIT**);

Forcing Completion of Drawing

glFlush(); makes sure the OpenGL commands are executed in finite time.

Window Management using GLUT

- **glutInitDisplayMode()** - tells whether to create RGB or color index window; a single- or double buffered window can be specified; a depth buffer can also be specified.
- **glutInitWindowSize()** - indicates the windows' size in pixels.
- **glutInitWindowPosition()** - tells where to position a window on the screen.
- **glutCreateWindow()** - opens a window on the screen and returns a unique identifier for the window.

OpenGL Example

```
void InitGraphics (void)
{
   /*Always open a display mode first*/
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

   glutInitWindowSize(1000,800);
   glutInitWindowPosition(0,0);  /* Can be omitted, then you
                                    can place the window with
                                    your mouse */
   glutCreateWindow("Assignment 1");
}
```

Display Callback using GLUT

glutDisplayFunc() - specifies the function called whenever the contents of the window need to be redrawn.

```
Main()
   :
   glutDisplayFunc(display);
   glutMainLoop();  /* Starting event processing*/
       (The last thing to do in Main)
```

Handling Input Events using GLUT

glutReshapeFunc() - what actions to be taken when the window is resized, moved or exposed.

glutKeyboardFunc() - links a key with a routine that is invoked when a key is pressed. **glutSpecialFunc()** - F1, F2, F3 etc.

glutMouseFunc() - links a mouse button with a routine that is invoked when the mouse button is pressed/released.

glutMotionFunc() - is called when the mouse moves within the window while one or more buttons are pressed. Also, **glutPassiveMotionFunc()** (no buttons pressed).

More GLUT

glutPostRedisplay() - it will call the function registered with glutDisplayFunc.

glutSwapBuffers() - one buffer is displayed while one is being redrawn. Animation, rotation. glutInitDisplayMode(**GLUT_DOUBLE**| …..) Should be called at the end of the function registered with glutDisplayFunc.

OpenGL Example

```
Main()
   InitGraphics();
   glClearColor(0.0,0.0,0.0,0.0);
   glClear(GL_COLOR_BUFFER_BIT);
   glutReshapeFunc(SetView);
   glutKeyboardFunc(Keyboard);
   glutMouseFunc(MouseEvent);
   glutMotionFunc(MoveMouse);
   glutDisplayFunc(display);
   glutMainLoop();
```

Attributes of Output Primitives

glPointSize(size); Default is 1.0.
glLineWidth(width); Default is 1.0.
Default line type attribute is solid.
gLineStripple(repeatfactor,pattern); Default pattern 0xFFFF.
For example, **glLineStripple**(1,0x3F07); ex. lines.c
glEnable(GL_LINE_STRIPPLE); Turn off: **glDisable**
glPolygonStripple(filpattern); ex. polys.c
glEnable(GL_POLYGON_STRIPPLE);

Fonts

glutBitmapCharacter(font, string);

GLUT_BITMAP_HELVETICA_18

GLUT_BITMAP_HELVETICA_12

GLUT_BITMAP_TIMES_ROMAN_24

GLUT_BITMAP_9_BY_15

Pop-Up Menus

int glutCreateMenu(void (*func)(int value));

void glutAddMenuEntry(char *name, int value);

void glutAddSubMenu(char *EntryName, int MenuIndex);

void glutAttachMenu(int button);

void glutDetachMenu(int button);

Pop-Up Menu Example

```
void InitMenu()   /*Called in main*/
   int c_menu, p_menu, f_menu
   c_menu = glutCreateMenu(color_menu);
         glutAddMenuEntry("Red",1);
         glutAddMenuEntry("Green",2);
   p_menu = glutCreateMenu(pixel_menu);
         glutAddMenuEntry("Increase Pixel",1);
         glutAddMenuEntry("Decrease Pixel",2);
   glutCreateMenu(right_menu);
         glutAddMenuEntry("quit",1);
         glutAddMenuEntry("clear",2);
   glutAttachMenu(GLUT_RIGHT_BUTTON);
```

Pop-Up Menu Example

```
glutCreateMenu(middle_menu);
glutAddSubMenu ("Colors",c_menu);
glutAddSubMenu ("Pixel Size",p_menu);
glutAttachMenu(GLUT_MIDDLE_BUTTON);

void color_menu (int id)
   if (id == 1) {
         r = 1.0;
         g = 0.0;
         b = 0.0;
   }
```

OpenGL Color Models

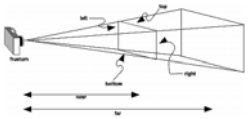
- RGBA color
  - Separate channel for each primary color.
  - Additional channel for alpha (α) used for transparency.
  - 8 bits/channel = 16 million colors.
- Indexed Color
  - Small number of colors accessed by indices into a color lookup table.
  - 8 bit = 256 colors

Hidden Surface Removal

- When setting up your window, specify a dept buffer:
  glutInitDisplayMode(…|…| GLUT_DEPTH);
- When clearing, make sure to:
  glClear(GL_DEPTH_BUFFER_BIT);
- **glEnable(GL_DEPTH_TEST);**
- Set the depth test comparison operation:
  glDepthFunc(GL_LESS); (this is the default)

Perspective Projection

glFrustum(-1.0,1.0,-1.0,1.0,2.5,100.0) - creates a perspective view frustum. Does not require a symmetric viewing volume.



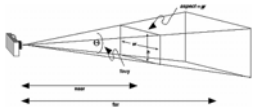
gluLookAt(0,0,3,0,0,0,0,1,0) - creates line of sight.

Perspective Projection

gluPerspective(60.0,w/h,1.0,200.0) -

- 60.0 = angle of the field of view
- w/h = aspect ratio, width divided by height
- 1.0 = zNear ; 200.0 = zFar - distance between viewpoint and clipping planes. Always positive.

Creates frustums that are symmetric in both x- and y- axes along the line of sight. The line of sight points down the negative z- axis (unless rotation or translation is applied).



Normal Vectors

Assigning approximated normal vectors to vertices.

```
glBegin(GL_TRIANGLES);
    glNormal3fv(n0);
    glVertex3fv(v0);
    glNormal3fv(n1);
    glVertex3fv(v1);
    glNormal3fv(n2);
    glVertex3fv(v2);
glEnd();
```

In Gouraud shading you need a normal for each vertex in the polygon.
In flat shading you need one normal representing the polygon.

Display List

glNewList(Polygon_List, GL_COMPILE) - begins the display list.
glEndList() - ends the display list.
glCallList(Polygon_List**)** - executes the display list.

Using display list will optimize:

glRotate()
Matrix operations
Raster bitmaps and images
Lights, material properties and lighting models
Textures
Polygon stripple patterns

Transformations

glMatrixMode(GLenum *mode***)** - Specifies the modelview- or projection- matrix.(GL_MODELVIEW or GL_PROJECTION).
glLoadIdentity() - loads the identity matrix.

Before doing **rotation**, **translation** etc. you need to set :
glMatrixMode(GL_MODELVIEW **);**

Before setting the view volume (Ortho or Perspective) you need to set:
glMatrixMode(GL_ PROJECTION**);**

Manipulating the Matrix Stacks

glPushMatrix() copies current matrix and adds the copy to the top of the stack. "remember where you are"
glPopMatrix() discards the top matrix of the stack. "go back to where you were"

Modeling Transformation

glTranslate{f,d}(-10.0,0.0,0.0**)** - 10 in x direction

glRotate{f,d}(45.0,0.0,0.0,1.0**)** - 45 degrees about x-axis.

glScale{fd}(2.0,-0.5,1.0**)**

Creating Light Sources

glLight{if}v() - creates light source.

```
float   light_ambient[] = {0.0,0.0,0.0,1.0};
float   light_diffuse[] = {1.0,1.0,1.0,1.0};
float   light_specular[] = {1.0,1.0,1.0,1.0};
float   light_position[] = {0.0,1.5,2.5,1.0};
```

glLightfv(GL_LIGHT0,GL_AMBIENT,light_ambient);
glLightfv(GL_LIGHT0,GL_DIFFUSE,light_diffuse);
glLightfv(GL_LIGHT0,GL_SPECULAR,light_specular);
glLightfv(GL_LIGHT0,GL_POSITION,light_position);

Defining Material Properties

glMaterial{if}v() - defines the material property.

```
float mat_ambient[] = {?,?,?,1.0};        float mat2_ambient[] = {?,?,?,1.0};
float mat_diffuse[] = {?,?,?,1.0};        float mat2_diffuse[] = {?,?,?,1.0};
float mat_specular[] = {?,?,?,1.0};       float mat2_specular[] = {?,?,?,1.0};
float mat_shininess[] = {30.0};           float mat2_shininess[] = {30.0};
```

glMaterialfv(GL_FRONT,GL_AMBIENT,mat_ambient);
glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

glMaterialfv(GL_BACK,GL_AMBIENT,mat2_ambient);
glMaterialfv(GL_ BACK,GL_DIFFUSE,mat2_diffuse);
glMaterialfv(GL_ BACK,GL_SPECULAR,mat2_specular);
glMaterialfv(GL_ BACK,GL_SHININESS,mat2_shininess);

Material and Light

glEnable(GL_LIGHTNING**)** - enables the lightning.
glEnable(GL_LIGHT0**)** - enables the light source.

Should be turned of in wire frame mode.
glDisable(GL_LIGHTNING**)**
glDisable(GL_LIGHT0**)**

Inside/outside coloring: (GL_FALSE is default)
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);