EECS 104
Fundamentals of Computer Graphics

OpenGL

Introduction to OpenGL

- Graphics API
- Window system independent
- Operating system independent
- Geometric and image primitives
OpenGL as a Renderer

- **Geometric primitives**
  - points, lines, triangles, quadrilaterals, polygons

- **Image Primitives**
  - images and bitmaps
  - separate pipeline for images and geometry

- **Rendering depends on state**
  - Colors
  - Materials
  - Lights
Related APIs

- **AGL, GLX, WGL**
  - glue between OpenGL and windowing systems

- **GLU (OpenGL Utility Library)**
  - part of OpenGL
  - NURBS, tessellators, quadric shapes, etc.

- **GLUT (OpenGL Utility Toolkit)**
  - portable windowing API
  - not officially part of OpenGL
OpenGL and Related APIs

application program

- OpenGL Motif widget or similar
- GLX, AGL or WGL

GLUT

GLU

X, Win32, Mac O/S

GL

software and/or hardware
Preliminaries

- Libraries

- Headers Files
  - #include <GL/gl.h>
  - #include <GL/glu.h>
  - #include <GL/glut.h>
GLUT Basics

• Application Structure

1. Configure and open window
2. Initialize OpenGL state
3. Register input callback functions
   • render
   • resize
   • input: keyboard, mouse, etc.
4. Enter event processing loop
GLUT Display Modes

GLUT_RGBA  Select an RGBA mode window. This is the default if neither GLUT_RGBA nor GLUT_INDEX are specified.
GLUT_RGB   same as GLUT_RGBA.
GLUT_INDEX Select color index window mode. This overrides GLUT_RGBA.
GLUT_SINGLE Select a single buffered window. This is the default.
GLUT_DOUBLE Select a double buffered window. This overrides GLUT_SINGLE.
GLUT_ACCUM  Select a window with an accumulation buffer.
GLUT_ALPHA  Select a window with an alpha component to the color buffer(s).
GLUT_DEPTH  Select a window with a depth buffer.
GLUT_STENCIL Select a window with a stencil buffer.
GLUT_MULTISAMPLE Select a window with multismapling support.
GLUT_STEREO  Select a stereo window.
GLUT_LUMINANCE Select a stereo window with a "luminance" color model.
void main( int argc, char** argv )
{
    int mode = GLUT_RGB|GLUT_DOUBLE;
    glutInitDisplayMode( mode );
    glutCreateWindow( argv[0] );
    init();

    glutDisplayFunc( display );
    glutReshapeFunc( resize );
    glutKeyboardFunc( key );
    glutIdleFunc( idle );
    glutMainLoop();
}
Configure the state machine

```c
void init( void )
{
    glClearColor( 0.0, 0.0, 0.0, 1.0 );
    glClearDepth( 1.0 );

    glEnable( GL_LIGHT0 );
    glEnable( GL_LIGHTING );
    glEnable( GL_DEPTH_TEST );
}
```
GLUT Callback Functions

- **Routine to call when something happens**
  - window resize or redraw
  - user input
  - Animation

- **Register callbacks with GLUT**

  ```c
  glutDisplayFunc( display );
  glutIdleFunc( idle );
  glutKeyboardFunc( keyboard );
  ```
Rendering Callback

- Do all of your drawing here

```c
glutDisplayFunc( display );

void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glBegin( GL_TRIANGLE_STRIP );
    glVertex3fv( v[0] );
    glVertex3fv( v[1] );
    glVertex3fv( v[2] );
    glVertex3fv( v[3] );
    glEnd();
    glutSwapBuffers();
}
```
Idle Callbacks

- Use for animation and continuous update

```c
    glutIdleFunc( idle );

    void idle( void )
    {
        t += dt;
        glutPostRedisplay();
    }
```
User Input Callbacks

- Process user input

    glutKeyboardFunc( keyboard );

    void keyboard( char key, int x, int y )
    {
        switch( key ) {
            case 'q': case 'Q':
                exit( EXIT_SUCCESS );
                break;
            case 'r': case 'R':
                rotate = GL_TRUE;
                break;
        }
    }
Elementary Rendering

- Geometric Primitives
- OpenGL State Machine
- OpenGL Buffers
OpenGL Geometric Primitives

- All geometric primitives are specified by vertices
OpenGL Command Formats

**glVertex3fv( v )**

- **Number of components**
  - 2 - (x,y)
  - 3 - (x,y,z)
  - 4 - (x,y,z,w)

- **Data Type**
  - b - byte
  - ub - unsigned byte
  - s - short
  - us - unsigned short
  - i - int
  - ui - unsigned int
  - f - float
  - d - double

- **Vector**
  - omit "v" for scalar form
  - glVertex2f( x, y )
void drawRhombus( GLfloat color[] )
{
    glBegin( GL_QUADS );
    glColor3fv( color );
    glVertex2f( 0.0, 0.0 );
    glVertex2f( 1.0, 0.0 );
    glVertex2f( 1.5, 1.118 );
    glVertex2f( 0.5, 1.118 );
    glEnd();
}
Specifying Geometric Primitives

- Primitives are specified using
  
  ```c
  glBegin( primType );
  glEnd();
  ```

  - `primType` determines how vertices are combined

  ```c
  GLfloat red, greed, blue;
  GLfloat coords[3];
  glBegin( primType );
  for ( i = 0; i < nVerts; ++i ) {
      glColor3f( red, green, blue );
      glVertex3fv( coords );
  }
  glEnd();
  ```
OpenGL Color Models

- RGBA or Color Index

![Diagram of color models and processes]
Shapes Tutorial

```
glBegin (GL_TRIANGLES_STRIP);
setColor3f (1.00,0.00,1.00);
vertex2f (0.0,25.0);
setColor3f (0.00,1.00,1.00);
vertex2f (50.0,150.0);
setColor3f (0.00,1.00,0.00);
vertex2f (125.0,100.0);
setColor3f (1.00,1.00,0.00);
vertex2f (175.0,200.0);
end();
```
Controlling Rendering Appearance

- From Wireframe to Texture Mapped
OpenGL’s State Machine

- All rendering attributes are encapsulated
  - rendering styles
  - shading
  - lighting
  - texture mapping
Manipulating OpenGL State

- Appearance is controlled by current state
  for each (primitive to render)
  {
    update OpenGL state
    render primitive
  }

- Manipulating vertex attributes is most common way to manipulate state
  
  glColor*() / glIndex*()  
  glNormal*()  
  glTexCoord*()
Controlling current state

- **Setting State**

```c
glPointSize( size );
gLineStipple( repeat, pattern );
gShadeModel( GL_SMOOTH );
```

- **Enabling Features**

```c
gEnable( GL_LIGHTING );
gDisable( GL_TEXTURE_2D );
```
Transformations in OpenGL

- Modeling

- **Viewing**
  - Viewpoint (camera location and orientations)
  - Projection

- Animation

- Map to screen
Camera Analogy

- Real-time photography

Diagram:
- Camera
- Tripod
- Model within a viewing volume
Camera Analogy and Transformations

- **Projection transformations**
  - adjust the lens of the camera

- **Viewing transformations**
  - tripod—define position and orientation of the viewing volume in the world

- **Modeling transformations**
  - moving the model

- **Viewport transformations**
  - enlarge or reduce the physical photograph
Coordinate Systems and Transforms

- **Steps in Forming an Image**
  - specify geometry (world coordinates)
  - specify camera (camera coordinates)
  - project (window coordinates)
  - map to viewport (screen coordinates)

- Each step uses transformations

- Every transformation is equivalent to a change in coordinate systems (frames)
Affine Transformations

- **Want transformations which preserve geometry**
  - lines, polygons, quadrics

- **Affine = line preserving**
  - Rotation, translation, scaling
  - Projection
  - Concatenation (composition)
Homogeneous Coordinates

- each vertex is a column vector
- \( w \) is usually 1.0
- all operations are matrix multiplications
- directions (directed line segments) can be represented with \( w = 0.0 \)

\[
\vec{v} = \begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]