OpenGL Introduction

Introduction

OpenGL

OpenGL is an API for computer graphics.

- Hardware-independent
  - Windowing or getting input is not included in the API
- Low-level
  - Only knows about triangles (kind of, but also: points, lines, polygons)
  - Libraries are often built on top of OpenGL
- State machine
  - The various API calls change the OpenGL state, query some part of that state, or cause OpenGL to use its current state to render something

GLUT (OpenGL Utility Toolkit) or FreeGLUT

API for creating a display window and using keyboard/mouse interactions.

- See RayWindow.cpp to see how these are used for Assignment3

GLU (OpenGL Utility Library)

Provides commonly used functions on top of OpenGL.

- Viewing/Camera functions
  - gluLookAt (Sets the view matrix that transforms model coordinates to world coordinates)
  - gluPerspective (Sets up perspective projection given a field of view and aspect ratio) – (RayWindow::DisplayFunction)
- Higher level shape drawing functions
  - gluCylinder
  - gluSphere

Other

- GLEW - OpenGL Extension Wrangler
  - Helps loading extensions
- GLFW
– Similar to GLUT and FreeGLUT
• Many more

Tutorial 1 - Hello World! (First Window)

• glutInit
  – Primary initialization routine
• glutInitDisplayMode
  – Creates windows with certain types of windows
• glutCreateWindow
  – Creates the window with the provided name
• glutReshapeWindow
  – Changes the size of the window to the width and height provided

• glutDisplayFunc
  – Sets a callback function for the window. GLUT determines when the
display callback should be triggered based on the window’s redisplay
state (can set explicitly by calling glutPostRedisplay).
• glutMainLoop
  – Starts the GLUT event processing loop, all registered callbacks will
now get called.

Tutorial 2 - First Rendering

• glClearColor
  – Establishes what color the window will be cleared to (background
color)
• glClear
  – Actually clears the window (by clearing the framebuffers)
• glGetFloatv
  – Get values from the current OpenGL state
• GL_MODELVIEW_MATRIX, GL_PROJECTION_MATRIX
  – Model coordinates (vertices defined relative to the model/object)
    * Model Matrix
  – World coordinates (vertices defined relative to the center of the
  world)
    * View Matrix
  – Camera coordinates (vertices defined relative to the camera)
    * Projection Matrix
  – Homogeneous coordinates (vertices defined relative to the camera’s
  perspective)
• glBegin & glEnd

2
Define what type of objects to draw
Possible types: GL_POINTS, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS, GL_QUAD_STRIP, and GL_POLYGON

- glColor3f
  - Set the color for a vertex (or any object)
  - All objects after this call will use this color, need to call it again to change
- glVertex3f
  - Vertices of a shape
- glutSwapBuffers
  - Tells the window to use the current buffers in the OpenGL state (calls glFlush)

---

Tutorial 3 - Model View and Projection Matrices

- glMatrixMode
  - Specifies the matrix that will be operated on
- glLoadIdentity
  - Replaces the specified current matrix with the identity matrix
- gluPerspective
  - For setting the perspective transformation (projection matrix)
  - 4 arguments:
    * fovy - field of view (degrees) in y direction
    * aspect - aspect ratio (width/height)
    * zNear - near clipping plane
    * zFar - far clipping plane
- gluLookAt
  - For setting the viewing transformation (view matrix)
  - 3 sets of arguments
    * Eye x,y,z - location of the camera (3D point) - Center/Reference
    * x,y,z - where camera is looking (3D point) - Up x,y,z - rotation of the camera (3D vector)
  - You want to move forward - What is the direction vector (forward vector) of the camera???
  - You want to move right What is the vector pointing to the right of the camera???
- glRotatef
  - Rotates the current object angle degrees around the vector x,y,z
- glTranslatef
  - Translates in the given x,y,z direction
Tutorial 4 - Transformation Stack

- glPushMatrix
  - Pushes the current matrix stack down by one, duplicating the current matrix
- glPopMatrix
  - Pops the current matrix stack, replacing the current matrix with the one below it

Explanation:
- You are applying transformations to the modelview matrix (with glTranslate or glRotate) and then you are in some good reference position. You can push a duplicate matrix onto the stack, perform a transformation on that, and then pop it off and you are back at that reference position.
- This avoids having to either make backwards transformations to get back to your reference position or making the whole original sequence of transformations again.

Tutorial 5, 6 - Call lists

Not used in Assignment 3 anymore

Tutorial 7 - Lighting

- glEnable
  - Enables a state variable
- glEnable(GL_LIGHTING)
  - If not enabled (or glDisable is called) then everything is colored according to the glColor calls made
- glEnable(GL_LIGHT0)
- glLightfv
  - Sets parameters for a light
  - Parameters:
    * light - which light it is
    * pname - which parameter name to change for the light
    * params - the value(s) of the parameters
- glMaterialfv
– Sets parameters for a material (with respect to a light)
– Parameters:
  * face - face being updated (GL_FRONT, GL_BACK, or GL_FRONT_AND_BACK)
  * pname - which material parameter to change
  * params - the value(s) of the parameters
– pname options: (GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_EMISSION, GL_SHININESS, GL_AMBIENT_AND_DIFFUSE, or GL_COLOR_INDEXES)

Tutorial 8 - Shading

Tutorial 9 - Texture

Tutorial 10 - Depth Test

Tutorial 11 - Transparency