15-462 Computer Graphics:
Quick Introduction to OpenGL

By Raphael Mun

Parts of OpenGL

- GL – short for OpenGL
  - The Graphics API we will be using
- GLU – OpenGL Utility Library
  - Helpful Utility Functions
- GLUT – OpenGL Utility Toolkit
  - Wrapper around window-creation and program-flow

Creating a Render Window (GLUT)

- Allocating Display Memory [Size & Type] (functions are basically called in the following order)
  - glutInitDisplayMode(bit-wise flags);
    - Types of buffers you want:
      - GLUT_RGB
      - GLUT_DOUBLE
      - GLUT_DEPTH
      - [GLUT_STENCIL]
      - [GLUT_ACCUM]
  - glutInitWindowSize(width, height);
  - glutInitWindowPosition(x, y);
  - glutCreateWindow(window name);
  - [glutEnterGameMode()] - Full-screen render mode

Event Handling (GLUT)

- Assign Event-handlers by passing in functions that glut will call as needed ("Callback Functions")
  - glutDisplayFunc
  - glutMouseFunc
  - glutMotionFunc
  - glutKeyboardFunc
  - glutIdleFunc

Setting Up The Program

- Main components of rendering in 3D
  - Camera – Size, Field of View, Orientation, Clipping Planes
  - World Objects – Scale, Rotation, Position
  - Resources – Textures, Menus, etc.
- These are loaded or initialized at the beginning of the program, ready for access in memory

The Rendering Process

- Inside your display function, typically the process goes…
  - Clear the Screen
    - **[glClear]**
  - Set Model Transformations
    - **[glScale, glRotate, glTranslate]**
  - Draw Objects
    - **[glBegin, glColor/Normal/Vertex/TexCoord, glEnd, glFlush]**
  - Flip the Front and Back Buffers
    - **[glutSwapBuffers]**

The Rendering Process Cont'd

- Objects drawn are made up of primitive units commonly referred to as "polygons" or "primitives"
  - Set of 3-vertices, minimum # needed to define a plane with a normal vector (*in computer graphics, 'polygons' are almost always triangles in 3D space*)

Object Drawing

- Objects can be drawn in multiple ways:
  - **Triangle-List** (most common and easiest in concept)
    - Each set of 3 vertices define an independent polygon
    - Memory intensive, Not optimized (There is also an indexed variation)
    - Visibility Culling works only in one-direction (Clockwise or Counter)
  - **Triangle-Strip** (useful for connected polygons)
    - Every vertex matched with the last two vertices of the previous polygon define a new polygon
    - Can save a lot of memory in most cases, e.g. heightmaps
    - Alternates between Clockwise and Counter for every polygon
  - **Triangle-Fan** (useful only in certain cases with one central vertex)
    - Every vertex matched with the last vertex of the previous polygon and the **first** vertex define a new polygon
    - Tough to use and limiting, but can be rewarding for some models
    - Culling works in one-direction
  - **Quads** – 4-vertex polygons, unique to OpenGL and usually slower

Applying Transformations

- There are 3 matrix-modes in OpenGL that correspond to the graphics pipeline
  - *GL_PROJECTION* – Camera Transformation
  - *GL_MODELVIEW* – Vertex Transformation
  - *GL_TEXTURE* – Pixel Transformation
- GLU Utility Library provides a some functions to make it easier for camera transformations
  - gluPerspective
  - gluLookAt
- Mainly for ease of programming but these functions can be called anywhere in the pipeline
  - Transformation = *PROJ* x *MODELVIEW* x *VIEWPORT*
  - Mode set with glMatrixMode(mode), and then functions called

Underlying OpenGL Matrix Functions

- Matrix-Level Functions (manage your own matrix arrays)
  - glLoadIdentity
  - glLoadMatrix
  - glMultMatrix
  - glLoadTransposeMatrix
  - glMultTransportMatrix
- Simple Transformation Functions
  - glScalef
  - glRotatef
  - glTranslatef
  - [* - there are multiple versions such as glScalei which takes in integer parameters]

Questions?