CSC 4356
Interactive Computer Graphics
Lecture 5: OpenGL Basics

Jinwei Ye

http://www.csc.lsu.edu/~jye/CSC4356/

Tue & Thu: 10:30 - 11:50am
218 Tureaud Hall

Lecture 5: OpenGL Basics

- Today let’s do some graphics!
- What is OpenGL?
- How to write an OpenGL program?

What is OpenGL?

- Simple API for 2D/3D graphics
  - Cross-platform 3D solution
  - Optimized for graphics card
- Libraries
  - GL (Graphics Library): 2D/3D drawing
  - GLU (GL Utilities) : camera setup and higher-level shape description
  - GLUT (GL Utilities Toolkit): utility functions dealing with windows and user interface

Other Useful Libraries

- GLUT replacements
  - Freeglut: open source and extended alternative to GLUT
  - http://freeglut.sourceforge.net/
  - GLFW: cross-platform windowing/UI toolkit
  - http://www.glfw.org/
- We will use Freeglut for our homework

History of OpenGL

- Silicon Graphics (SGI) revolutionized the graphics workstation by implementing the pipeline in hardware: Iris GL (1982)
- First version of OpenGL: an open source alternative to Iris GL (1992)
- Controlled by an Architectural Review Board (ARB)
  - Oversees changes in the language specification
  - Founder: SGI, Intel, IBM, DEC and Microsoft
  - Editor board of the Redbook

OpenGL Evolution

- OpenGL is young (born in 1992)
  - Cross-platform (Windows, Linux, Mac OS, Mobile OS …)
  - Easy to use and focus on rendering
  - Close to the hardware to get excellent performance
- Changes in the language have been slow
  - OpenGL 2.0 (2004)
  - OpenGL 3.0 (2008)
  - OpenGL 4.0 (2010)
  - Evolution reflects new hardware capacities

OpenGL Versions

- Classic OpenGL (2.x): Scene-based
  - Specify the objects, camera and lighting
  - Everything else handled for you
  - Supported on all non-mobile platforms
- New OpenGL (3.x and higher): shader program
  - Low-level control (vertex/fragment)
  - Shader programs are the primary focus
  - Allow customized graphics effects

OpenGL Versions

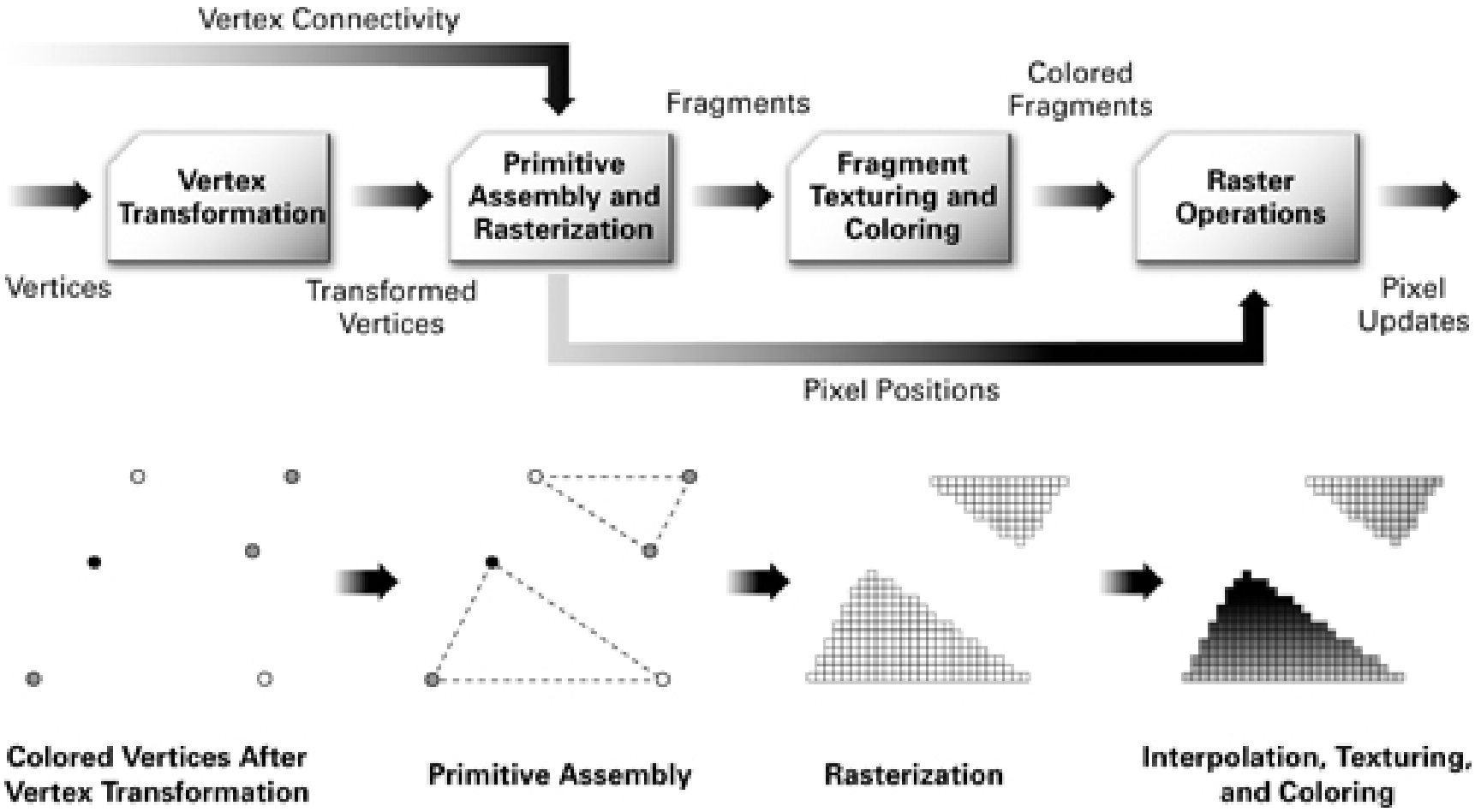
- Classic OpenGL (2.x): Scene-based
  - Specify the c… lighting
  - Everything e…
  - Supported o… forms
- New OpenGL (3.x and higher): shader program
  - Low-level control (vertex/fragment)
  - Shader programs are the primary focus
  - Allow customized graphics effects

Good starting point for learning OpenGL

OpenGL: A State Machine

- State Machine: Remain in the current state until you change it
  - For example: color
- Each state has a default value
- Most Functions manipulate global state
  - Modify global state
  - Query global state
  - Cause something to be rendered

Graphics Rendering pipeline



Images from Nvdia CG tutorial: http://developer.download.nvidia.com/CgTutorial/

Getting Started with OpenGL

- Windows OS: Visual Studio C++
  - https://www.visualstudio.com/vs/community/
- OpenGL is included in the graphics card driver (no installation required)
  - Only handles graphics rendering
- Need to install Freeglut
  - http://freeglut.sourceforge.net/
  - Handles windowing, events, and UI

Install Freeglut

- Download pre-compiled binaries
- Move *.h files to directory:

  `C:\Program Files(x86)\Microsoft SDKs \Windows\v7.0A\Include\gl`
- Move *.lib files to directory:

  `C:\Program Files(x86)\Microsoft SDKs \Windows\v7.0A\Lib`
- Move *.dll to directory:

  `C:\Windows\System32`
  - If you compiles 32 bit program on a 64 bit machine, move to:

  `C:\Windows\SysWOW64`

Create Project

- Create a Win32 Console Application
  - Console application
  - Empty project
- Create source file
  - Add New Item
  - Choose C++ file
- Configure Project Properties

  Project Properties -> Configuration Properties -> Linker -> Input -> Additional Dependencies:

  Opengl32.lib, glu32.lib, freeglut.lib

Simple_OpenGL.c++

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

void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();

glFlush ();
}

void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("hello");
    init ();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

OpenGL Function Conventions

- Many functions have multiple forms:
  - glVertex3f,glVertex2i, glColor3f, glColor3ub, etc.
- Number indicates number of arguments
- Letters indicate type
  - f:float, d:double, i:integer, ub:unsigned byte
- “v” (if present) indicates a pointer argument
  - For example: glVertex3f (x,y,z)
    glVertex3fv(pointer)

Program Structure

- Classic OpenGL programs have the following structure:
  - `main()`
    - Defines the callback functions
    - Opens one or more windows with the required properties
    - Enters event loop
  - `init()`
    - Sets the state variables (viewing, attributes)
  - callback functions
    - Input and display functions
    - For example: the display function

```
void display(void)
```

Simple_OpenGL.c++

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

Include GLUT header glut.h

```
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();

glFlush ();
}

void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("hello");
    init ();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Simple_OpenGL.c++

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

void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();

glFlush ();
}

void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("hello");
    init ();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Display function

Simple_OpenGL.c++

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

void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();

glFlush ();
}

void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("hello");
    init ();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Windows initialization

Simple_OpenGL.c++

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

void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();

glFlush ();
}

void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("hello");
    init ();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Main function

Main Function

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("hello");

    init ();

    glutDisplayFunc(display);

    glutMainLoop();
    return 0;
}
```

Define window properties

Call init() to initialize OpenGL states

Display callback

Enter event loop

GLUT Functions

- `glutInit` allows application to get command line arguments and initializes system
- `glutInitDisplayMode` requests properties for the window (rendering context)
  - RGB color
  - Single Buffering
- `glutWindowSize` specifies size of window in pixels
- `glutWindowPosition` specifies position of the window from top-left corner of the display
- `glutCreateWindow` creates window with OpenGL context
- `glutDisplayFunc` display function callback
- `glutMainLoop` enters infinite event loop

Double Buffering

- Instead of using one color buffer, we can use two
  - Front buffer: one that is displayed but not written to
  - Back buffer: one that is written to but not display
- How to request double buffer?

```
glutInitDisplayMode(GLUT_Double | GLUT_RGB);
```

- Swap buffer at the end of display callback

```
glutSwapBuffers()
```

- Improve rendering efficiency

Callback Function

- Programming interface for event-driven input
- Define a callback function for each type of event that the system recognize
  - glutDisplayFunc, glutMouseFunc, glutKeyboardFunc, etc
- Every OpenGL program must have a glutDisplayFunc

GLUT Event Loop

- Last line of the main function

```
glutMainLoop();
```

- In each pass through the event loop, GLUT
  - Looks at the events in the queue
  - For each event in the queue, GLUT executes the appropriate callback function if one is defined
  - If no callback is defined for the event, the event is ignored

Window Initialization

```
void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

Clear background color and set to black

Initialize viewing mode

These functions are only called once in the main function

Display Callback

```
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
        glVertex3f (0.25, 0.25, 0.0);
        glVertex3f (0.75, 0.25, 0.0);
        glVertex3f (0.75, 0.75, 0.0);
        glVertex3f (0.25, 0.75, 0.0);
    glEnd();

glFlush ();
}
```

Clear color buffer

Specify color for rendering

Draw a polygon

Start drawing!

Specify Geometric Primitives

- In classic OpenGL, primitives are specified using

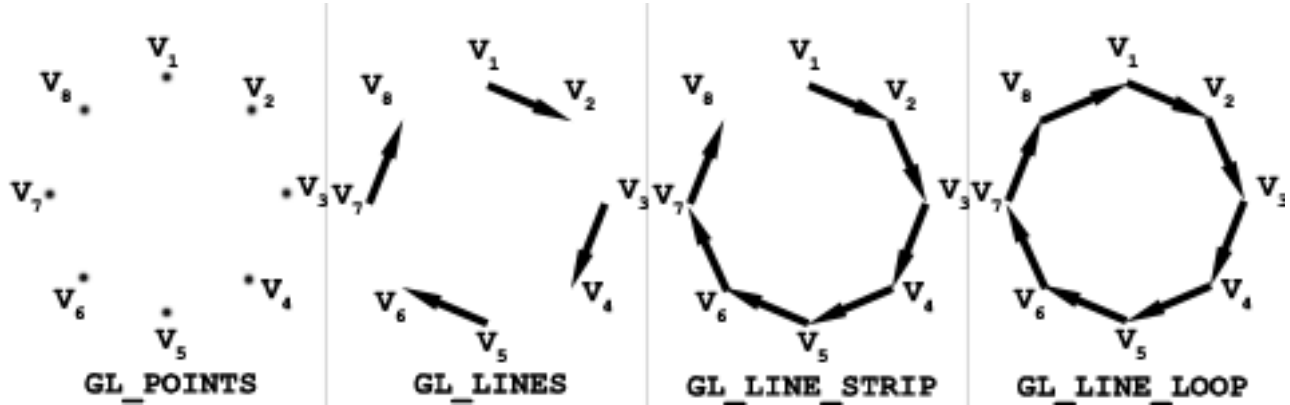
```
glBegin (primitive_type);
glColor3f(red, green, blue);
glVertex3f(x,y,z);
… \\next vertex
glEnd();
```

Vertex color

Vertex Position

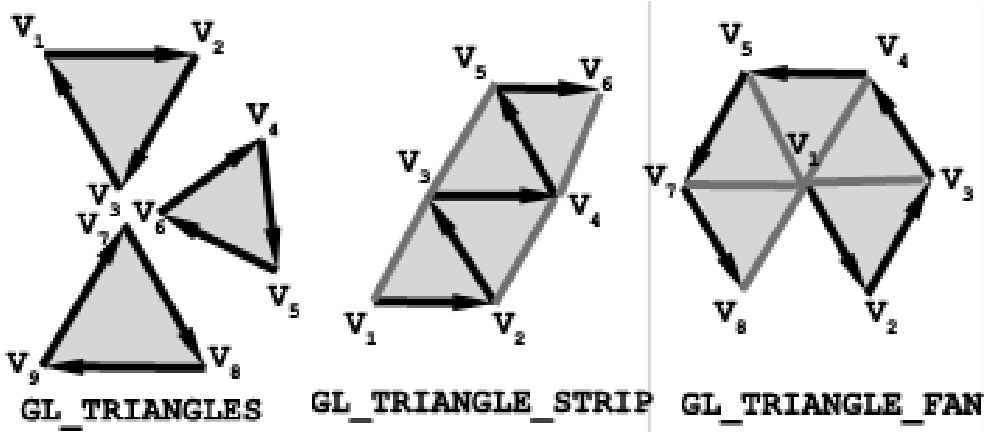
OpenGL primitives (2D)

- `GL_POINTS` (Draw n points)
  - Simply draw each vertex as an individual point
- `GL_LINES` (Draw n/2 lines)
  - Take pairs of vertices and draw lines between them
- `GL_LINE_STRIP` (Draw n-1 lines)
  - Draw lines between the vertices
- `GL_LINE_LOOP` (Draw n lines)
  - Same as above but also connect the first and last vertex



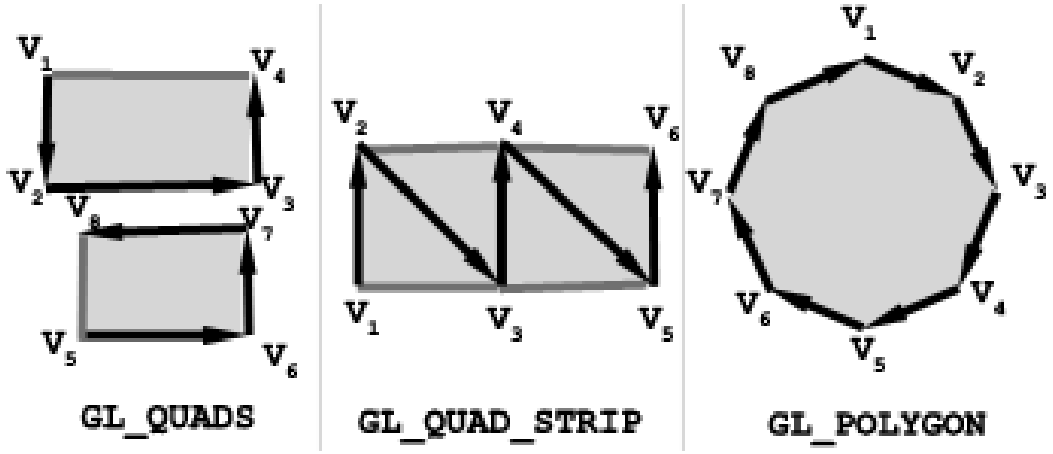
OpenGL primitives (2D)

- `GL_TRIANGLE` (Draw n/3 triangles)
  - Take vertices in triples to draw them as triangles
- `GL_TRIANGLE_STRIP` (Draw n-2 triangles)
  - Triangles are connected (similar to line strip)
- `GL_TRIANGLE_FAN` (Draw n-2 triangles)
  - Triangles radiating about a point



OpenGL primitives (2D)

- `GL_QUADS` (Draw n/4 quads)
  - Take vertices in quadruples to draw them as four-sided polygons
- `GL_QUADS_STRIP` (Draw (n-2)/2 quads)
  - Similar to triangle strip, use adjacent edges for form the next quad
- `GL_POLYGON` (Draw 1 convex polygon)



3D Primitives?

- GLUT provides routines for drawing a few 3D object
  - Cone, Icosahedron, Teapot, Cube, Octahedron, Tetrahedron, Dodecahedron, Sphere, Torus
- You can draw each object as a wireframe or solid shaded object

```
glutWireCube(GLdouble size);
glutSolidCube(GLdouble size);
```

Teapot Example

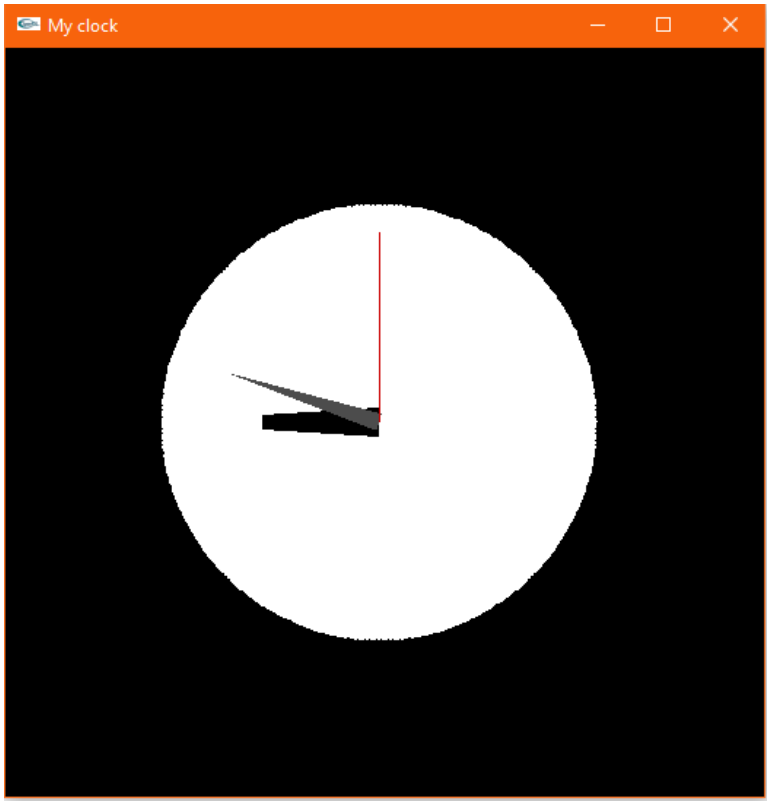
```
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);

    glColor3f (1.0, 1.0, 1.0);
    glutWireTeapot(0.5);


    glFlush ();
}
```

Other Important Functions

- glPushMatrix() / glPopMatrix()
  - Pushes/pops the transformation matrix onto the matrix stack
- glLoadIdentity(), glLoadMatrix(), glMultiMatrix()
  - Pushes the matrix onto the matrix stack
- Use these functions for geometric and viewing transformation

Programming Assignment 1

- Draw a virtual clock with OpenGL
  - Setup OpenGL
  - Draw primitives
  - 2D transformations
- Play with OpenGL!
- Due in Two weeks
  - 9/19 at 11:59pm



Programming Assignment 1

- I'll provide a skeleton code for you to start with
  - Instructions and skeleton code will be posted on our website
- Please also provide a **report** explaining how you implement the functionality with key codes and results
- Your homework will be graded according to both the effect of the program and the report
- TA will run your program using Visual Studio
- Please make sure your code works on another computer!

How to Submit Your Homework?

- Submit to the CSE Linux server classes.csc.lsu.edu
- Everyone will be given a user name & password
  - I'll send you via Email
- Connect to the server via SSH or secure FTP client (such FileZilla on port 22)
  - Use command p_copy
- If you are OFF campus, you must connect to the LSU VPN before you can connect to the server
  - LSU VPN instruction: https://networking.grok.lsu.edu/Article.aspx?articleId=14785
- Your submission will be time stamped on the server
- Detailed instructions will be posted on our course website

Next Time…

- More OpenGL …
- Rasterization
  - How OpenGL draws a line or triangle?