

Lecture slides (AI3501/CT4201/EC4215 – Computer Graphics)

Raster Image

Lecturer: Bochang Moon

Raster Images

- Raster display
 - Show images as rectangular arrays of pixels
 - e.g., computer display or TV
- Raster image
 - 2D array that stores the *pixel value* at each pixel
 - *Pixel value*?
 - Color = [red, green, blue]
- Other ways to define an image?
 - Vector image
 - Store descriptions of shape rather than pixels

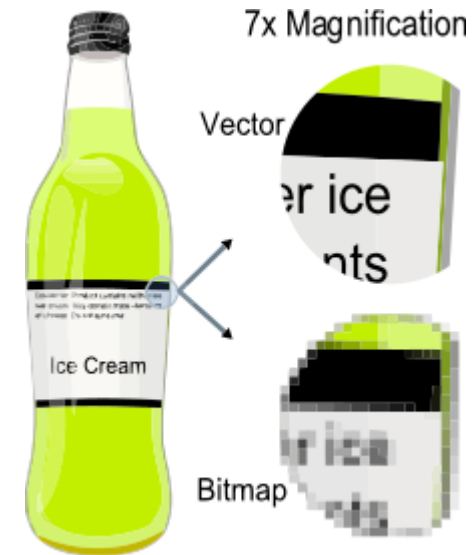


Image from wikipedia.org

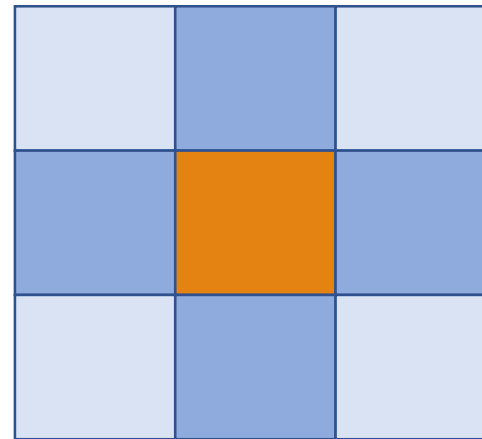
Raster Images

- Approximation of real image (continuous)

- $I(x, y): R \rightarrow V$

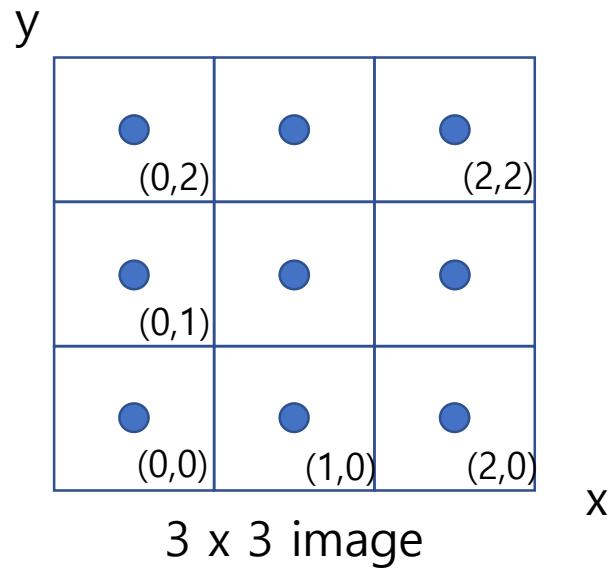
- $R \subset \mathbb{R}^2$: rectangular area

- $V = (\mathbb{R}^+)^3$



3 x 3 image

Raster Images



- Pixel: point sample
 - Local average of colors in the image
- Q. How do we average the colors?

Pixel Values

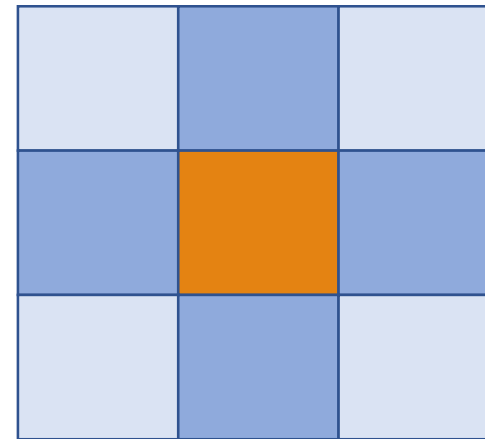
- Pixel values are stored in computer memory
- Pixel formats
 - 1-bit grayscale
 - 8-bit RGB (24 bits total)
 - 16-bit RGB
 - 32-bit RGB
- Q. low vs. high precision?

In this course,

- We will study how to generate raster images from virtual objects



Graphics
Techniques



3 x 3 image

Lecture slides (CT4201/EC4215 – Computer Graphics)

Basic OpenGL

Lecturer: Bochang Moon

Graphics Pipeline

- A series of computer operations to generate images from 3D objects
- Hardware pipeline
 - Real-time rendering (e.g., games)
 - APIs like OpenGL and DirectX
- Software pipeline
 - High-quality but offline rendering (e.g., animated films)
 - APIs like RenderMan
- Scope of this course?

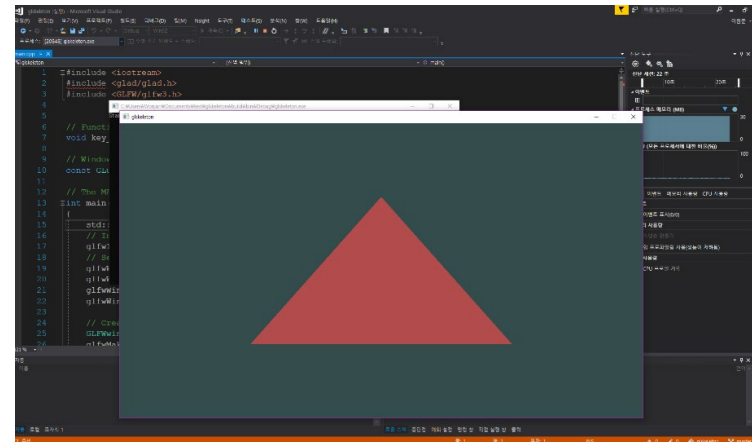
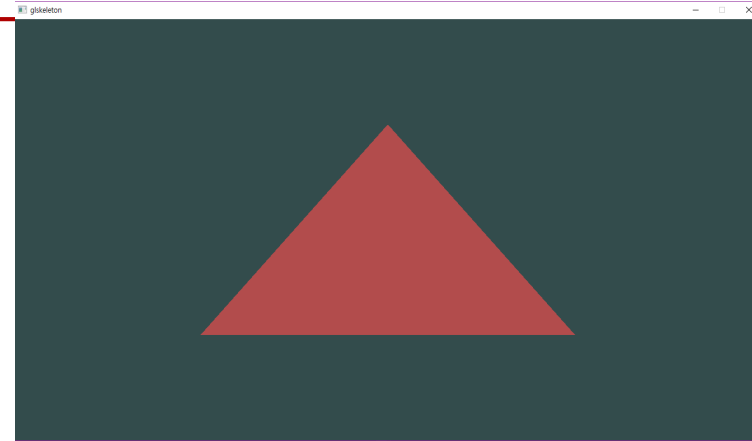
OpenGL

- Open Graphics Library (OpenGL):
 - Cross-platform application programming interface (API)
 - Typically interact with GPUs
 - Widely used API for interactive rendering

- Additional libraries
 - GLFW (Windowing library, <https://github.com/glfw/glfw.git>)
 - GLM (OpenGL math library, <https://github.com/g-truc/glm.git>)
 - GLAD(OpenGL loading library, <https://github.com/Dav1dde/glad.git>)
 - tinyobjloader(Mesh loading library, <https://github.com/syoyo/tinyobjloader.git>)

OpenGL Tools

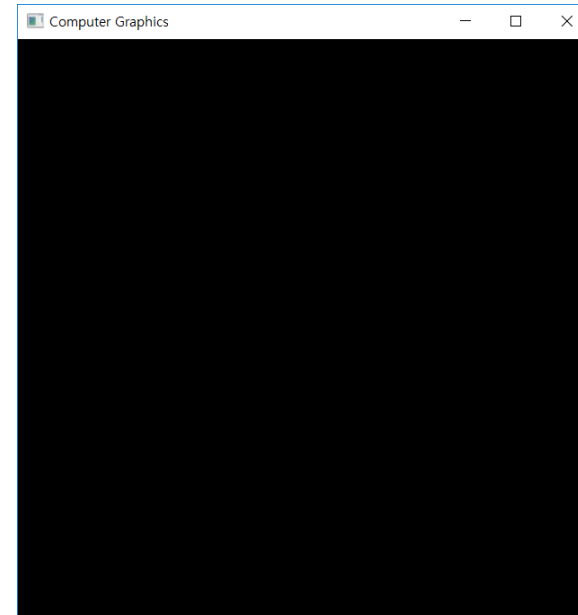
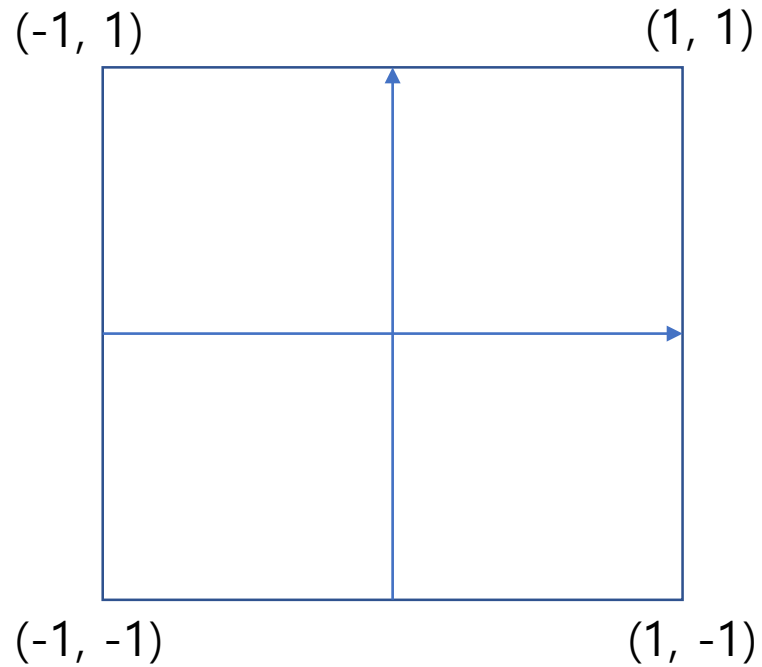
```
• int main() {  
•     ...  
•     // Init GLFW  
•     glfwInit();  
•     // Set all the required options for GLFW  
•     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);  
•     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 1);  
•     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_ANY_PROFILE);  
•     glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);  
  
•     // Create a GLFWwindow object that we can use for GLFW's functions  
•     GLFWwindow* window = glfwCreateWindow(WIDTH, HEIGHT, "glskeleton", NULL, NULL);  
•     glfwMakeContextCurrent(window);  
•     ...  
•     return 0;  
• }
```



OpenGL Tools

```
• int main()
• {
•     // Render loop
•     while (!glfwWindowShouldClose(window))
•     {
•         glfwPollEvents();
•         ...
•         glfwSwapBuffers(window);
•     }
•     ...
•     return 0;
• }
```

OpenGL Coordinates

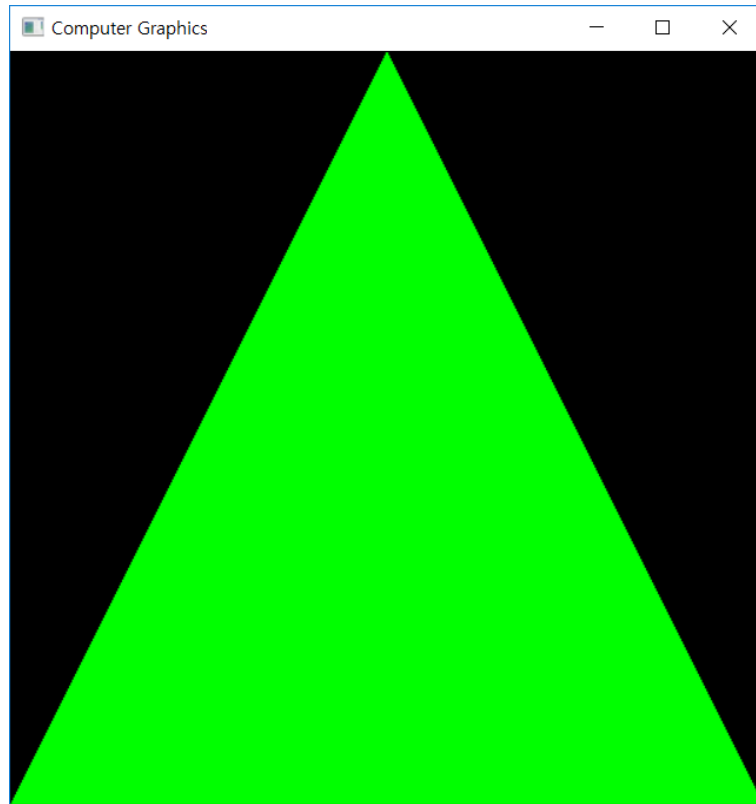


Draw Triangles

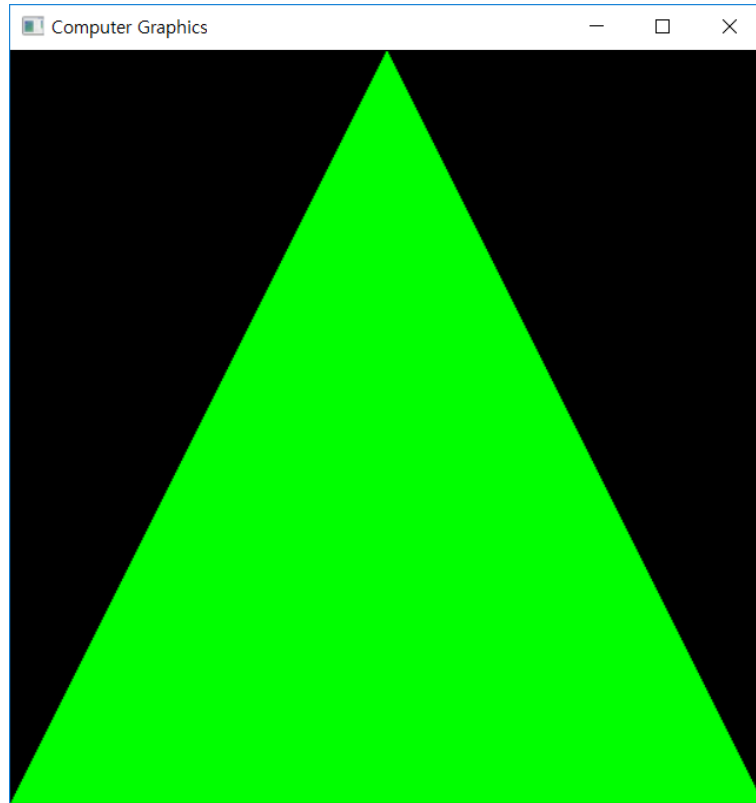


- `glColor3d(1.0, 0.0, 0.0);`
- `glBegin(GL_TRIANGLES);`
- `glVertex2d(-1.0, 1.0);`
- `glVertex2d(-1.0, -1.0);`
- `glVertex2d(1.0, -1.0);`
- `glEnd();`

Draw Triangles



Draw Triangles



- `glColor3d(0.0, 1.0, 0.0);`
- `glBegin(GL_TRIANGLES);`
- `glVertex2d(0.0, 1.0);`
- `glVertex2d(-1.0, -1.0);`
- `glVertex2d(1.0, -1.0);`
- `glEnd();`

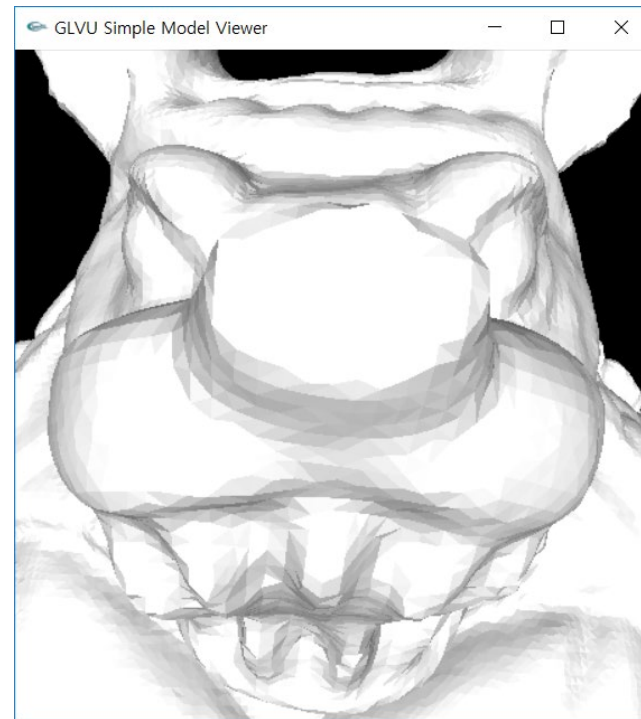
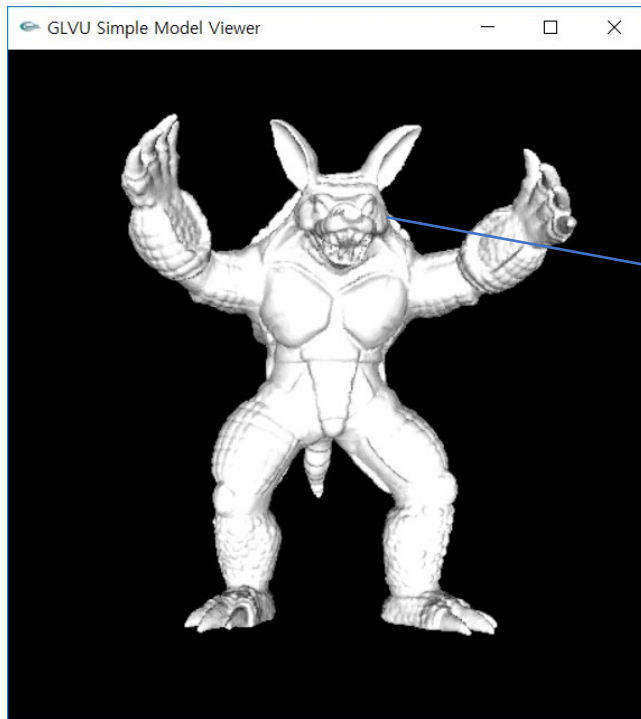
Full Source Code

- 81 lines of source code including boilerplate code and key callback function

```
glskeleton (선택 범위)
1 #include <iostream>
2 #include <glad/glad.h>
3 #include <GLEW/glfw3.h>
4 #include <glm/glm.hpp>
5
6 //soem changes
7 // Function prototypes
8 void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode);
9
10 // Window dimensions
11 const GLuint WIDTH = 1280, HEIGHT = 720;
12
13 // The MAIN function, from here we start the application and run the game loop
14 int main()
15 {
16     glm::vec3 v;
17
18     std::cout << "Starting GLFW context, OpenGL 3.1" << std::endl;
19     // Init GLFW
20     glfwInit();
21     // Set all the required options for GLFW
22     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
23     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 1);
24     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_ANY_PROFILE);
25     glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);
26
27     // Create a GLFWwindow object that we can use for GLFW's functions
28     GLFWwindow* window = glfwCreateWindow(WIDTH, HEIGHT, "glskeleton", NULL, NULL);
29     glfwMakeContextCurrent(window);
30     if (window == NULL)
31     {
32         std::cout << "Failed to create GLFW window" << std::endl;
33         glfwTerminate();
34         return -1;
35     }
36
37     // Set the required callback functions
38     glfwSetKeyCallback(window, key_callback);
39 }
```

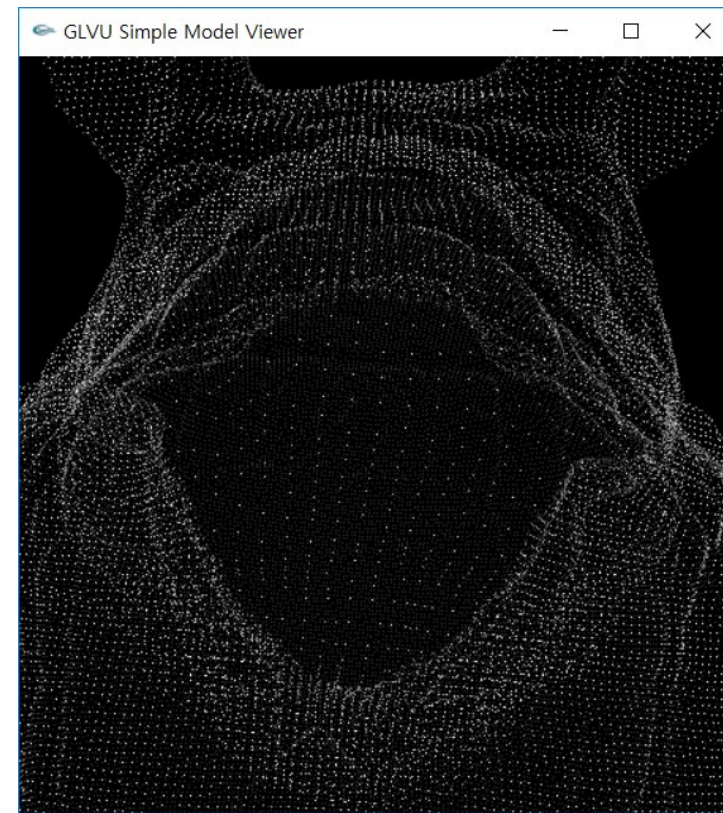
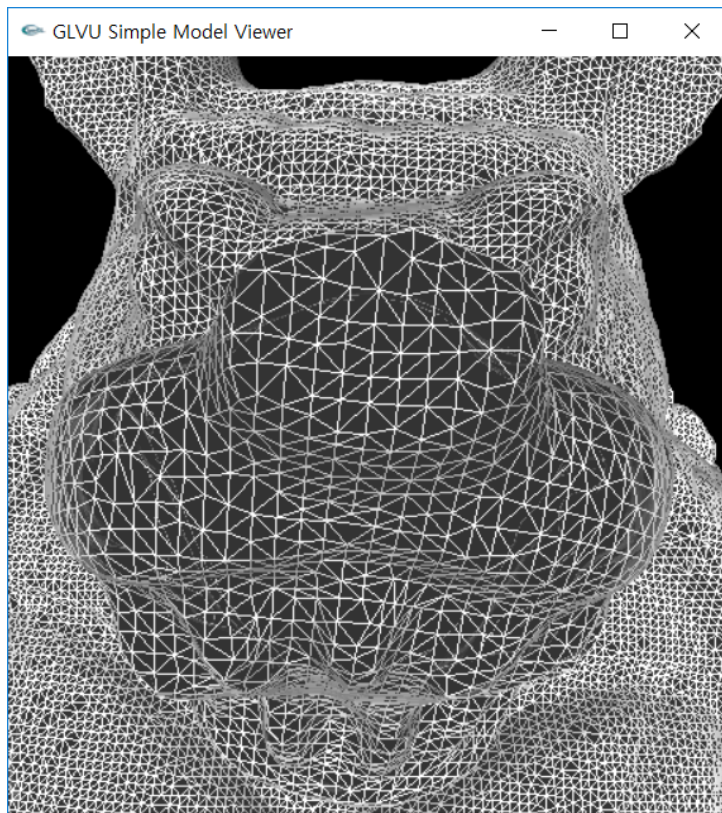
Triangles

- Fundamental modeling primitives

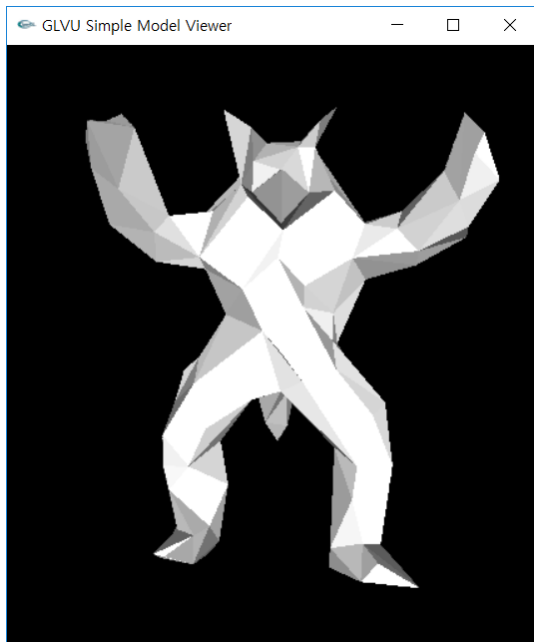


Triangles

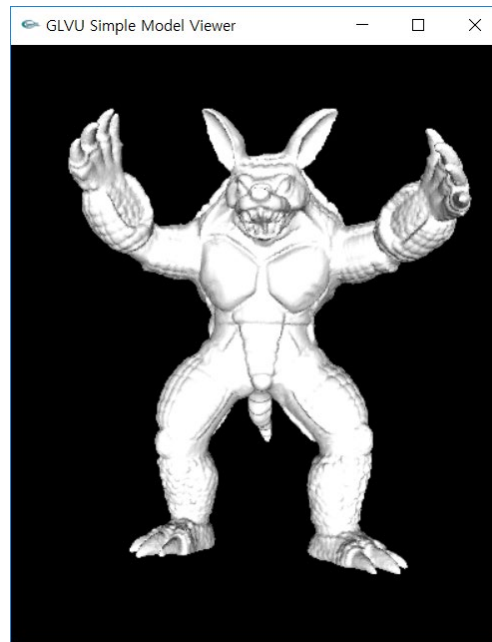
- Fundamental modeling primitives



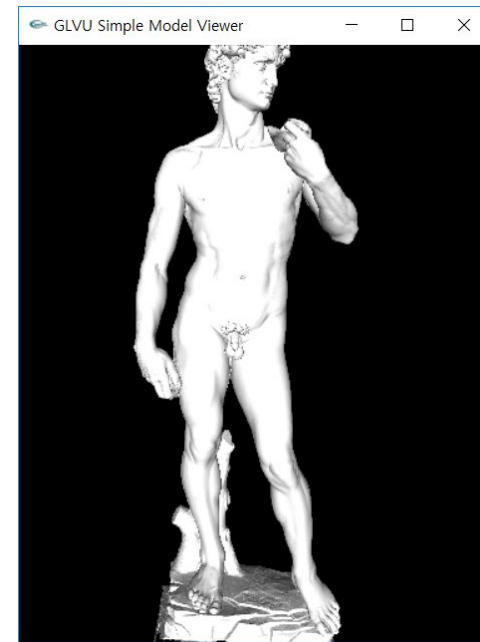
Triangles



300 triangles



345,944 triangles



8,254,150 triangles

Triangles

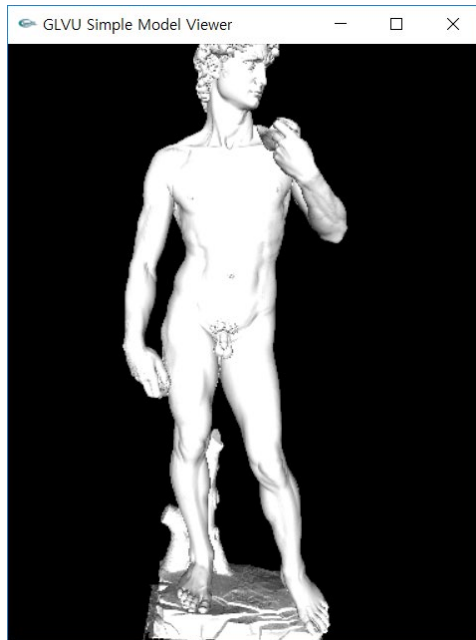
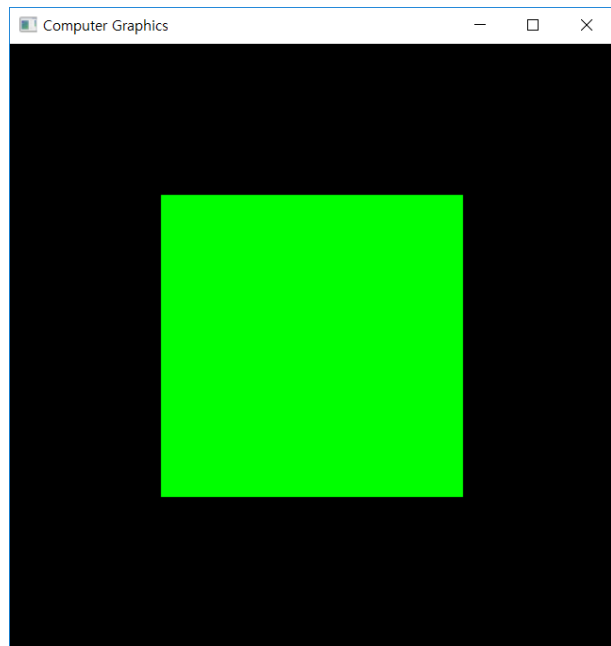


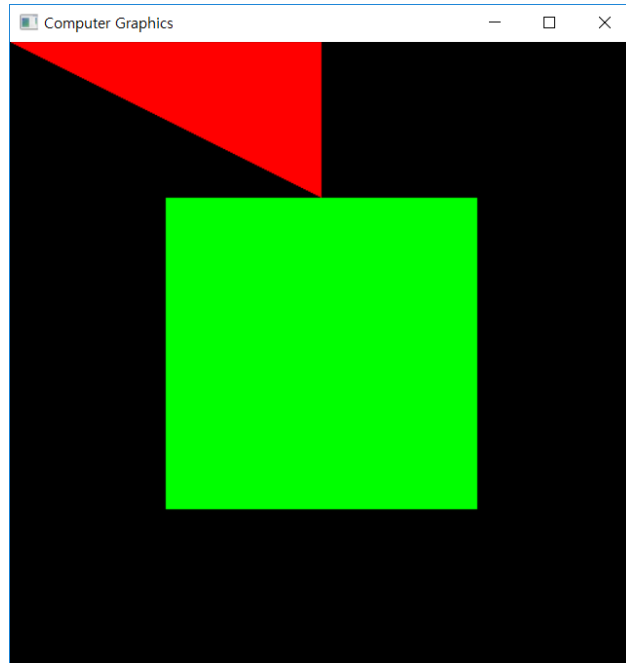
Image from
graphics.stanford.edu

Draw Other Shapes

- `glColor3d(0.0, 1.0, 0.0);`
- `glBegin(GL_QUADS);`
- `glVertex2d(-0.5, -0.5);`
- `glVertex2d(-0.5, 0.5);`
- `glVertex2d(0.5, 0.5);`
- `glVertex2d(0.5, -0.5);`
- `glEnd();`



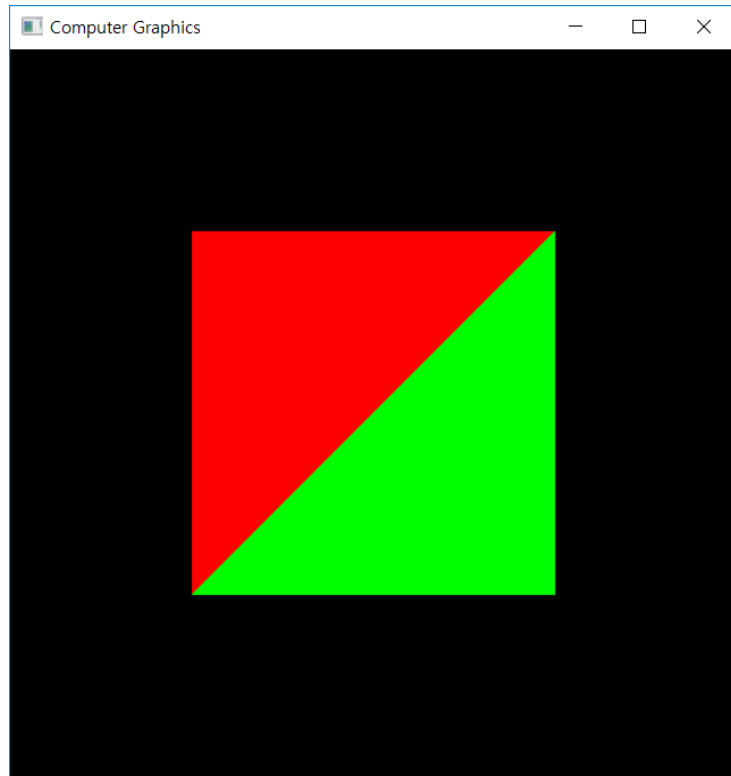
Draw Multiple Primitives



- glColor3d(1.0, 0.0, 0.0);
- glBegin(GL_TRIANGLES);
 - glVertex2d(-1.0, 1.0);
 - glVertex2d(0, 0.5);
 - glVertex2d(0, 1.0);
- glEnd();

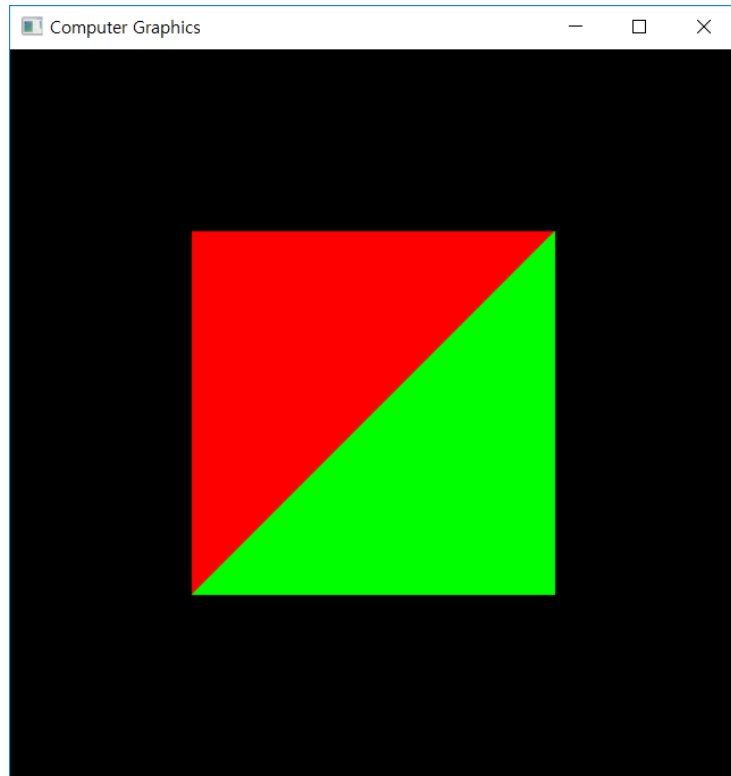
- glColor3d(0.0, 1.0, 0.0);
- glBegin(GL_QUADS);
 - glVertex2d(-0.5, -0.5);
 - glVertex2d(-0.5, 0.5);
 - glVertex2d(0.5, 0.5);
 - glVertex2d(0.5, -0.5);
- glEnd();

Example: Draw Two Triangles



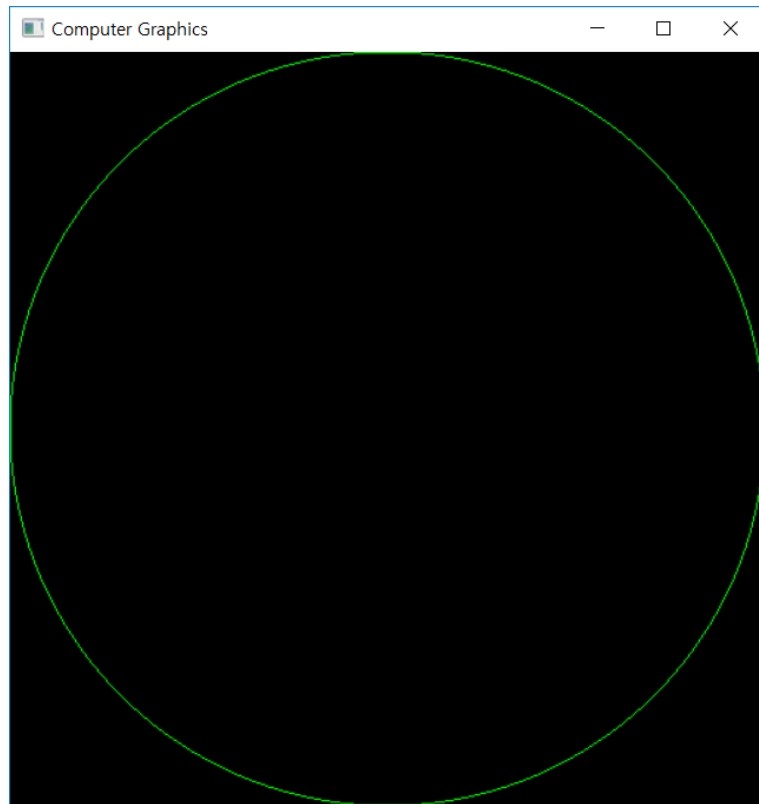
- `glColor3d(1.0, 0.0, 0.0);`
- `glBegin(GL_TRIANGLES);`
 - `glVertex2d(-0.5, -0.5);`
 - `glVertex2d(-0.5, 0.5);`
 - `glVertex2d(0.5, 0.5);`
- `glEnd();`
- `glColor3d(0.0, 1.0, 0.0);`
- `glBegin(GL_TRIANGLES);`
 - `glVertex2d(0.5, 0.5);`
 - `glVertex2d(-0.5, -0.5);`
 - `glVertex2d(0.5, -0.5);`
- `glEnd();`

Example: Draw Two Triangles



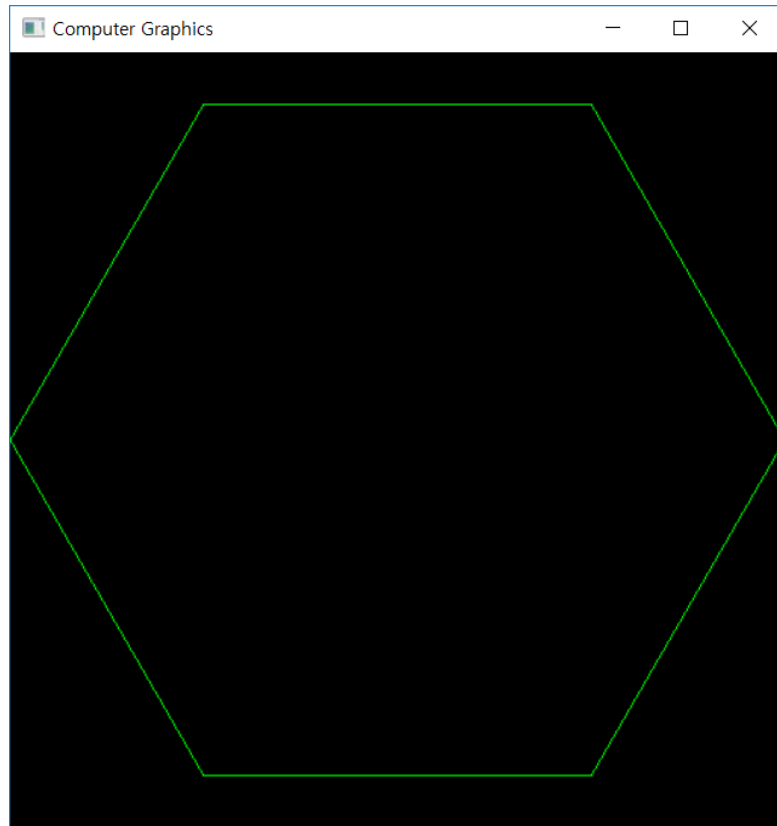
- `glColor3d(1.0, 0.0, 0.0);`
- `glBegin(GL_TRIANGLES);`
 - `glVertex2d(-0.5, -0.5);`
 - `glVertex2d(-0.5, 0.5);`
 - `glVertex2d(0.5, 0.5);`
- `glEnd();`
- `glColor3d(0.0, 1.0, 0.0);`
- `glBegin(GL_TRIANGLES);`
 - `glVertex2d(0.5, 0.5);`
 - `glVertex2d(-0.5, -0.5);`
 - `glVertex2d(0.5, -0.5);`
- `glEnd();`

Other Examples



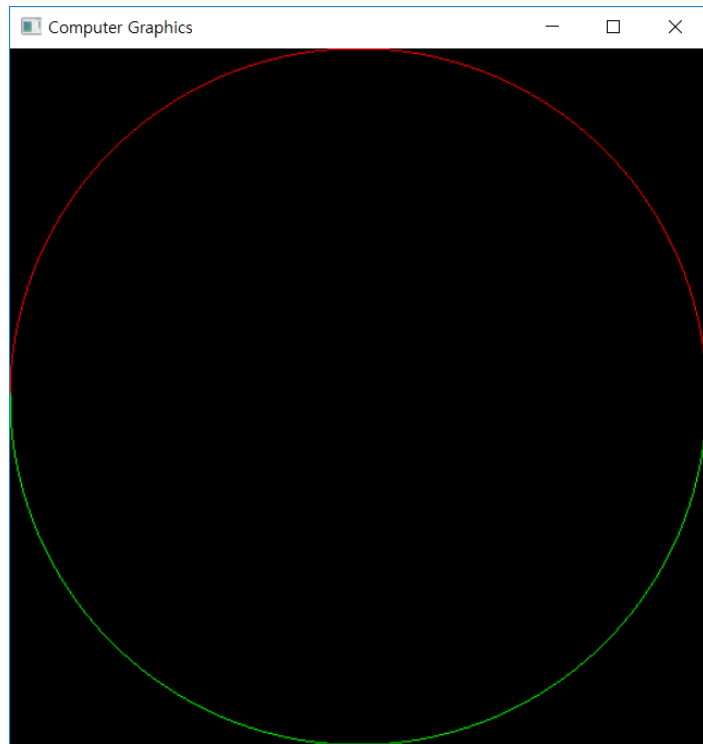
- `glColor3d(0.0, 1.0, 0.0);`
- `glBegin(GL_LINE_LOOP);`
- `for (int i = 0; i < 360; i = i + 1) {`
 - `double x = cos(i * PI / 180);`
 - `double y = sin(i * PI / 180);`
 - `glVertex2d(x, y);`
- `}`
- `glEnd();`

Other Examples



- `glColor3d(0.0, 1.0, 0.0);`
- `glBegin(GL_LINE_LOOP);`
- `for (int i = 0; i < 360; i = i + 60){`
 - `double x = cos(i * PI / 180);`
 - `double y = sin(i * PI / 180);`
 - `glVertex2d(x, y);`
- `}`
- `glEnd();`

Other Examples



- `glBegin(GL_LINE_LOOP);`
- `for (int i = 0; i < 360; i = i + 1){`
 - `double x = cos(i * PI / 180);`
 - `double y = sin(i * PI / 180);`
 - `if (i < 180)`
 - `glColor3d(1.0, 0.0, 0.0);`
 - `else`
 - `glColor3d(0.0, 1.0, 0.0);`
 - `glVertex2d(x, y);`
- `}`
- `glEnd();`