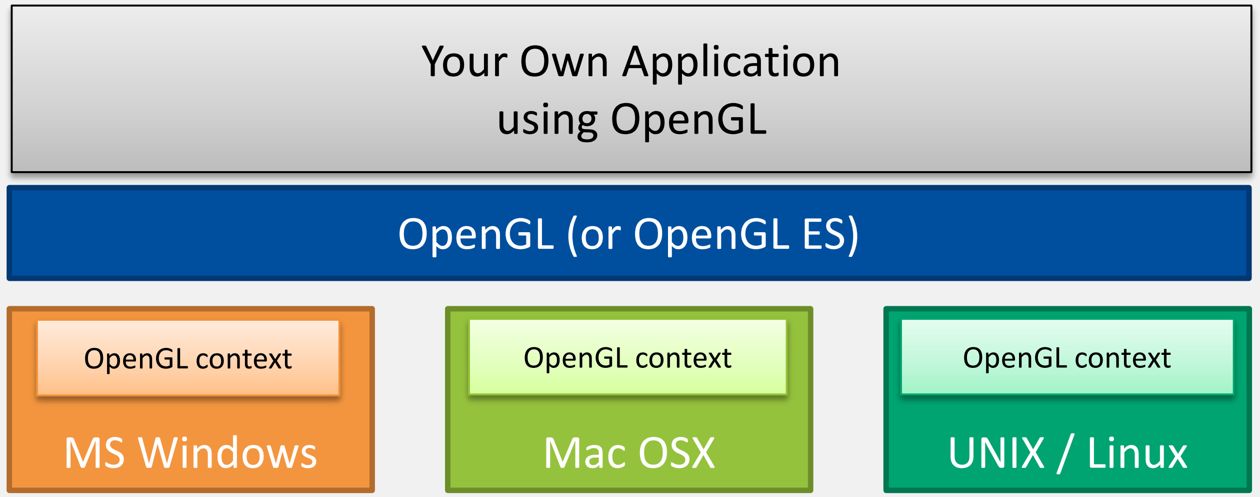
컴퓨터그래픽스 실습

김준호

비주얼컴퓨팅 연구실
국민대학교 소프트웨어학부

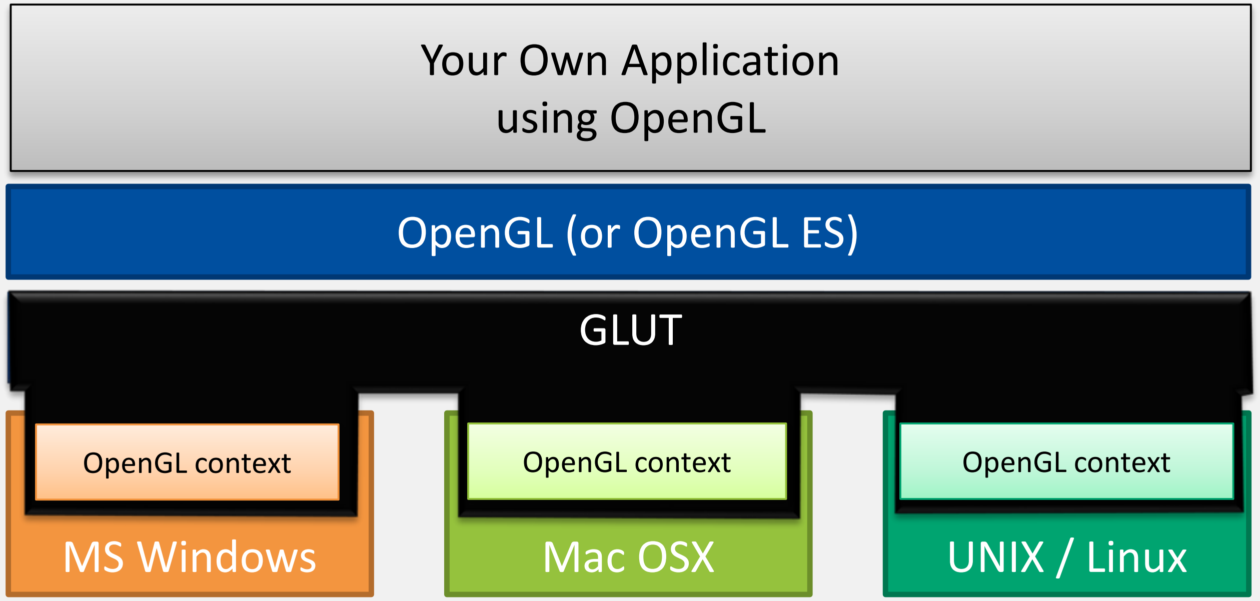
OpenGL & Window System

- OpenGL is a system-independent graphics library
  - But, context functions related to windows system-level I/O is system-dependent



GLUT

- OpenGL Utility Toolkit
  - Wrapping system-level I/O with hosting OS
    - Windows definition & controls, keyboard & mouse events
    - Routines for drawing several geometric primitives
  - Written by Mark J. Kilgard



GLUT installation

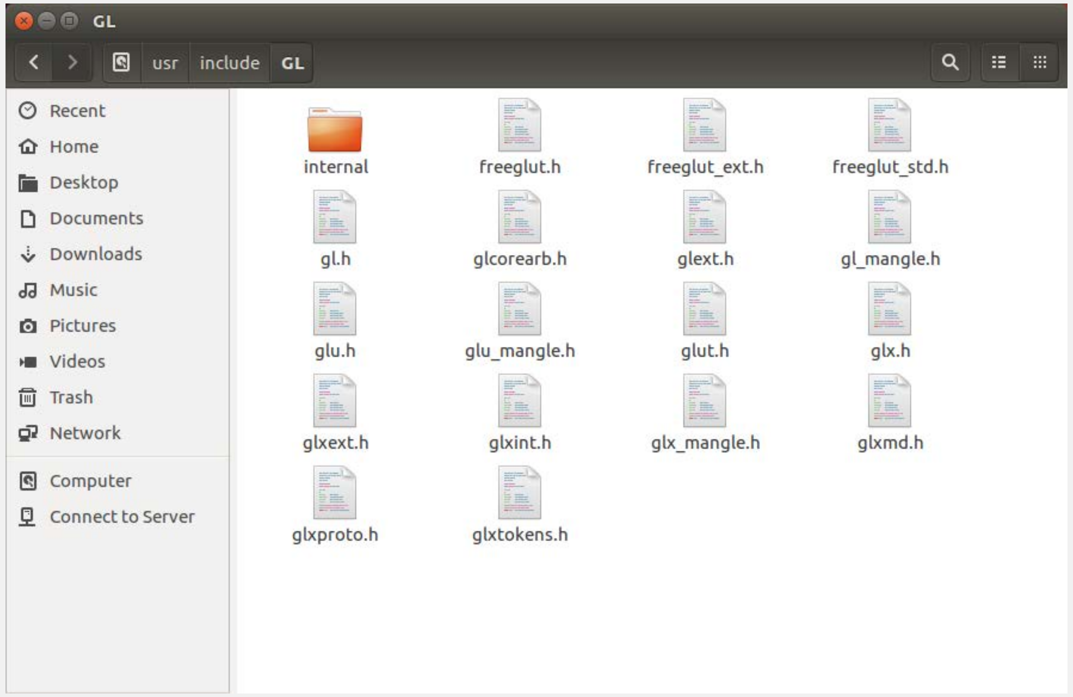
- Installation steps
  - Install using apt-get on Ubuntu 16.04 LTS
    - 아래의 명령어를 터미널에 입력

```
sudo apt-get install freeglut3-dev
```

    - See the details in the next slide
  - See the references when programming with GLUT
    - GLUT references: http://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf
    - Freeglut API doc: http://freeglut.sourceforge.net/docs/api.php
    - OpenGL Wikibook: http://en.wikibooks.org/wiki/OpenGL_Programming

GLUT installation

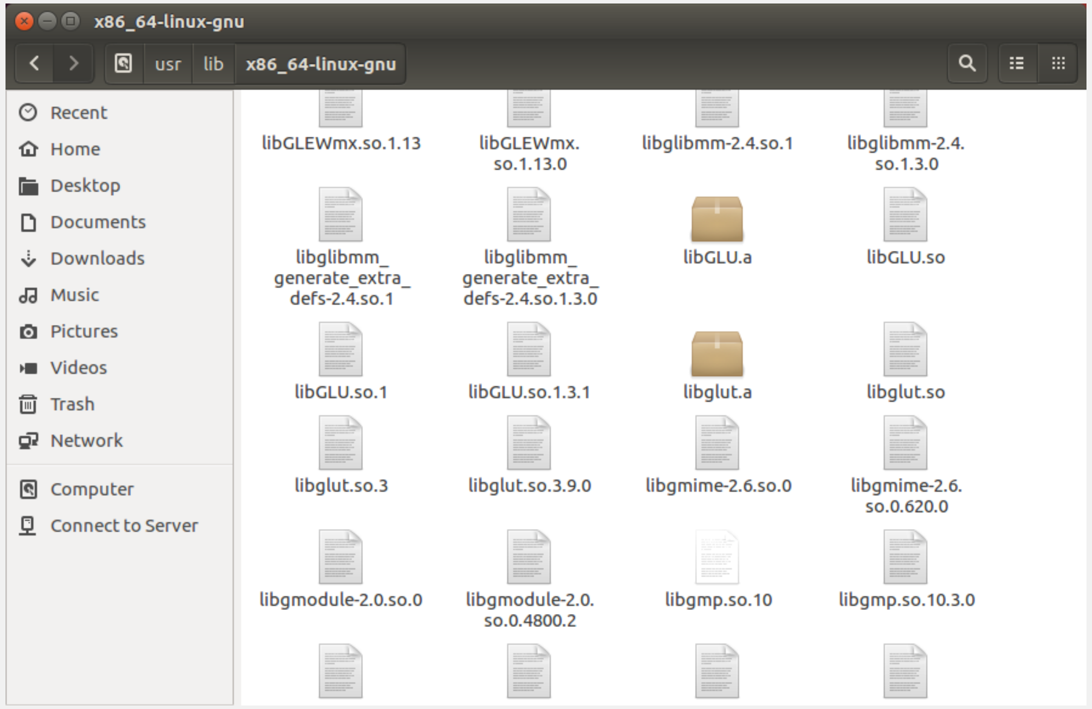
- Check freeglut include files
  - Find the usr include directory
    - `/usr/include/GL/`
  - Check 4 files (i.e., `freeglut*.h, glut.h`) in the include directory



internal, freeglut.h, freeglut_ext.h, freeglut_std.h, gl.h, glcorearb.h, glext.h, gl_mangle.h, glu.h, glu_mangle.h, glut.h, glx.h, glxext.h, glxint.h, glx_mangle.h, glxmd.h, glxproto.h, glxtokens.h

GLUT installation

- Check freeglut library files
  - Find the usr library directory
    - `/usr/lib/x86_64-linux-gnu/`
  - Check `libglut.a & libglut.so` in that folder



x86_64-linux-gnu

usr | lib | x86_64-linux-gnu

Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server

libGLEWmx.so.1.13 | libGLEWmx.so.1.13.0 | libglibmm-2.4.so.1 | libglibmm-2.4.so.1.3.0
libglibmm_generate_extra_defs-2.4.so.1 | libglibmm_generate_extra_defs-2.4.so.1.3.0 | libGLU.a | libGLU.so
libGLU.so.1 | libGLU.so.1.3.1 | libglut.a | libglut.so
libglut.so.3 | libglut.so.3.9.0 | libgmime-2.6.so.0 | libgmime-2.6.so.0.620.0
libgmodule-2.0.so.0 | libgmodule-2.0.so.0.4800.2 | libgmp.so.10 | libgmp.so.10.3.0

Programming with GLUT & Ubuntu 16.04 LTS

- Write source codes in `hello_world.cpp`

```cpp
#include <gl/glut.h>

void mydisplay();

int main(int argc, char* argv[])
{
  glutInit(&argc, argv);
  glutInitWindowSize(500, 500);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
  glutCreateWindow("simple");

  glutDisplayFunc(mydisplay);
  glutMainLoop();

  return 0;
}

void mydisplay()
{
}
```

Programming with GLUT & Ubuntu 16.04 LTS

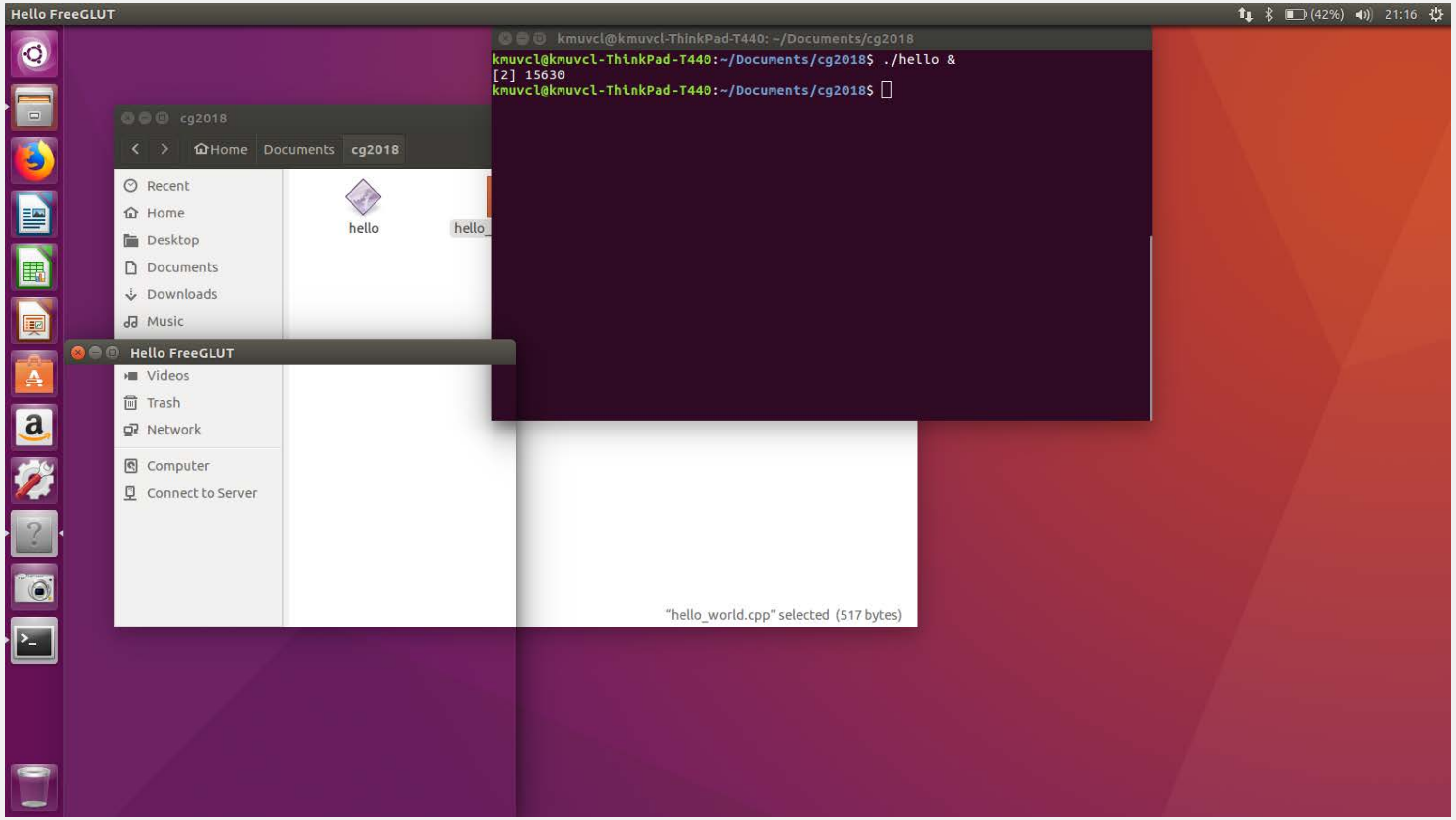
- Compile your program

```
g++ hello_world.cpp -o hello -lglut -lGL
```

  - `g++`: C++ 컴파일러로 g++를 이용함
  - `hello_world.cpp`: 컴파일할 소스파일 이름 지정
  - `-o hello`: 컴파일 후 만들어질 실행가능한 파일 이름을 hello로 설정
  - `-lglut`: FreeGLUT 라이브러리 파일을 찾아 링크하도록 함
  - `-lGL`: OpenGL 라이브러리 파일을 찾아 링크하도록 함

Programming with GLUT & Ubuntu 16.04 LTS

- Compile, link, and execute your program
  - In the first time, your program runs in a strange way

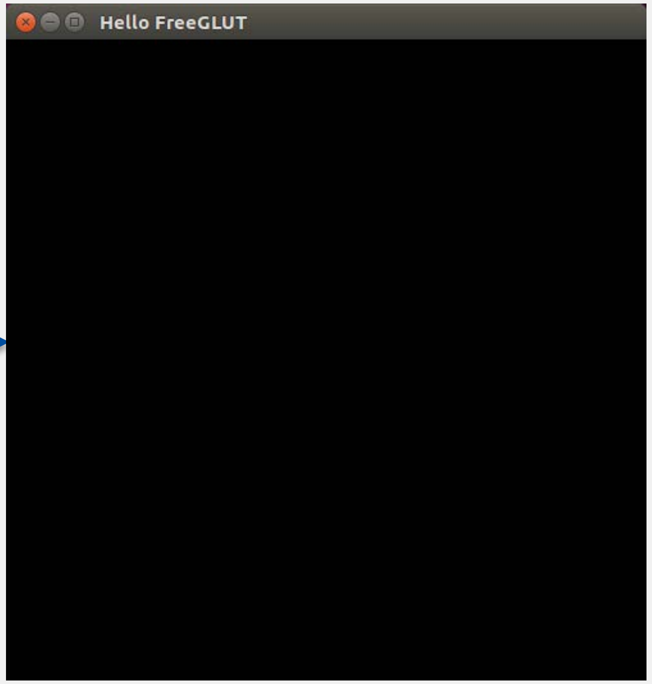


```
kmuvcl@kmuvcl-ThinkPad-T440:~/Documents/cg2018$ ./hello &
[2] 15630
kmuvcl@kmuvcl-ThinkPad-T440:~/Documents/cg2018$
```

Programming with GLUT & Ubuntu 16.04 LTS

- Update sources to get a properly working program
  - Clear the framebuffer & flush it

```c
#include <gl/glut.h>

void mydisplay();

int main(int argc, char* argv[])
{
  glutInit(&argc, argv);
  glutInitWindowSize(500, 500);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
  glutCreateWindow("simple");

  glutDisplayFunc(mydisplay);
  glutMainLoop();
  return 0;
}

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glFlush();
}
```



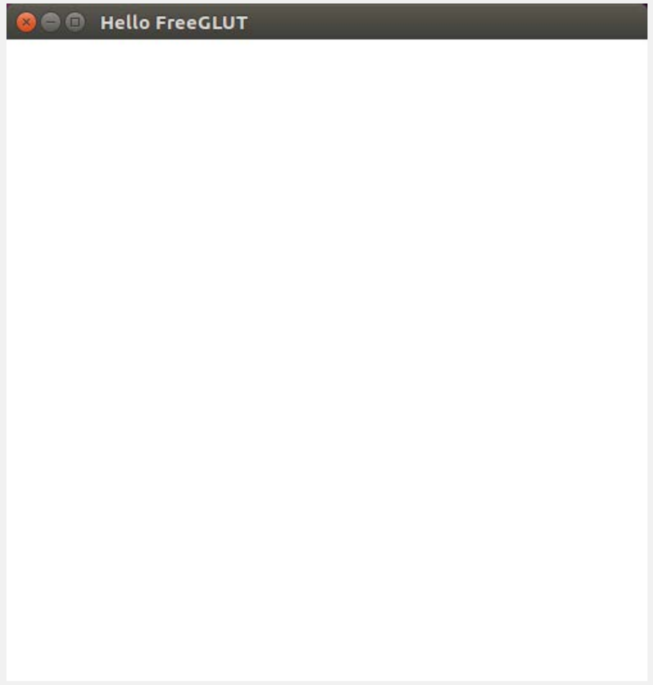
Refining Your Program: 2D Graphics

Set Clear Color

OpenGL codes

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

void mydisplay();
void init();

int main(int argc, char* argv[])
{
  // Same as usual
  // ...
  init();
  glutDisplayFunc(mydisplay);
  glutMainLoop();
  return 0;
}

void init()
{
  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
}

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glFlush();
}
```

Execution result

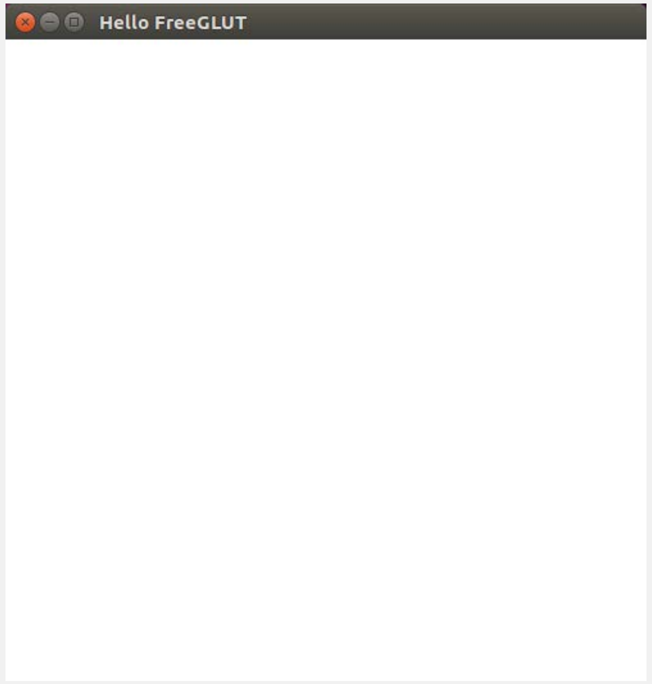


Draw a Rectangle

OpenGL codes

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

void mydisplay();
void init();

float vertices[] = {
  0.5f, 0.5f, -0.5f, 0.5f, -0.5f,-0.5f,
  0.5f, 0.5f, -0.5f,-0.5f,  0.5f,-0.5f,
};

// ...

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glEnableClientState(GL_VERTEX_ARRAY);
  glVertexPointer(2, GL_FLOAT, 0, vertices);

  glDrawArrays(GL_TRIANGLES, 0, 6);

  glDisableClientState(GL_VERTEX_ARRAY);

  glFlush();
}
```

Execution result



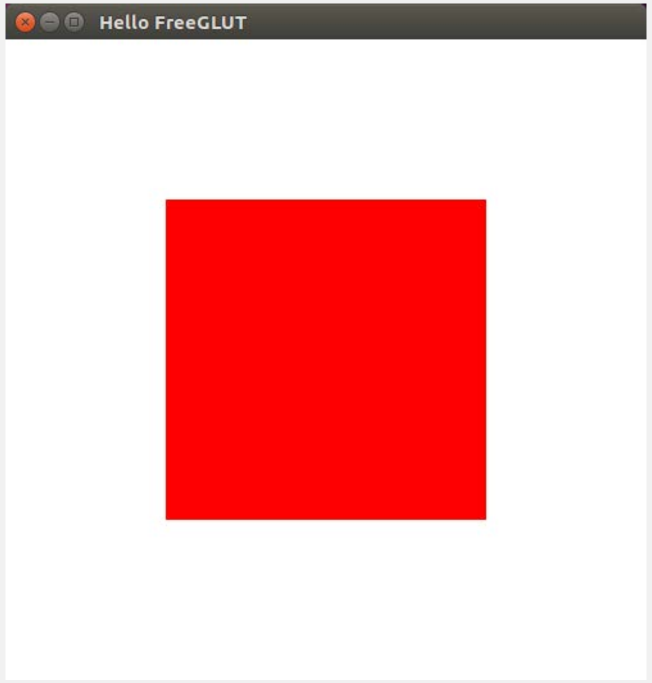
- There is nothing!!!
  - What is happening?

Set Current Color

OpenGL codes

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

void mydisplay();
void init();

float vertices[] = {
  0.5f, 0.5f, -0.5f, 0.5f, -0.5f,-0.5f,
  0.5f, 0.5f, -0.5f,-0.5f,  0.5f,-0.5f,
};

// ...

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(1.0f, 0.0f, 0.0f);
  glEnableClientState(GL_VERTEX_ARRAY);
  glVertexPointer(2, GL_FLOAT, 0, vertices);

  glDrawArrays(GL_TRIANGLES, 0, 6);

  glDisableClientState(GL_VERTEX_ARRAY);

  glFlush();
}
```

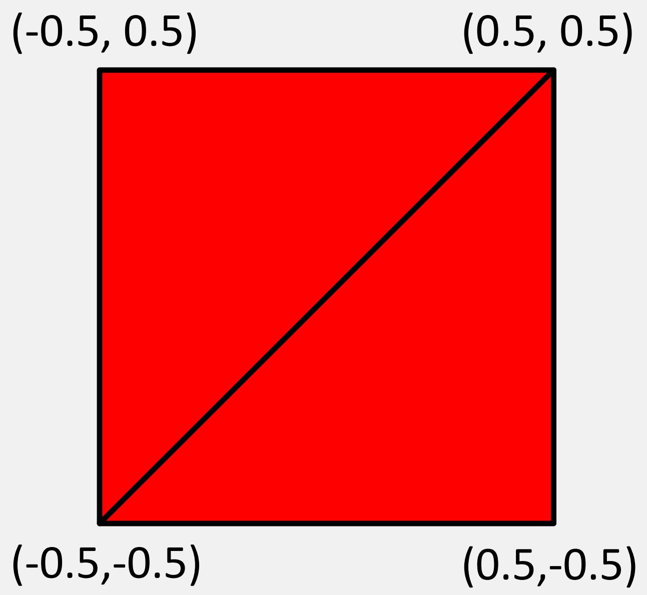
Execution result



Draw a Rectangle using Two Triangles

OpenGL codes

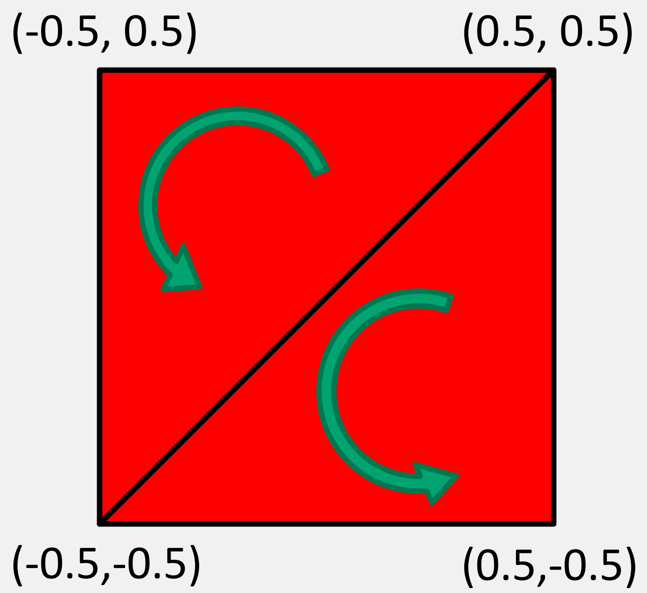
```
#include <gl/glut.h>

void mydisplay();
void init();

float vertices[] = {
  0.5f, 0.5f, -0.5f, 0.5f, -0.5f,-0.5f,
  0.5f, 0.5f, -0.5f,-0.5f,  0.5f,-0.5f,
};

// ...

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(1.0f, 0.0f, 0.0f);
  glEnableClientState(GL_VERTEX_ARRAY);
  glVertexPointer(2, GL_FLOAT, 0, vertices);

  glDrawArrays(GL_TRIANGLES, 0, 6);

  glDisableClientState(GL_VERTEX_ARRAY);

  glFlush();
}
```



Watch Out Your Vertex-Order

OpenGL codes

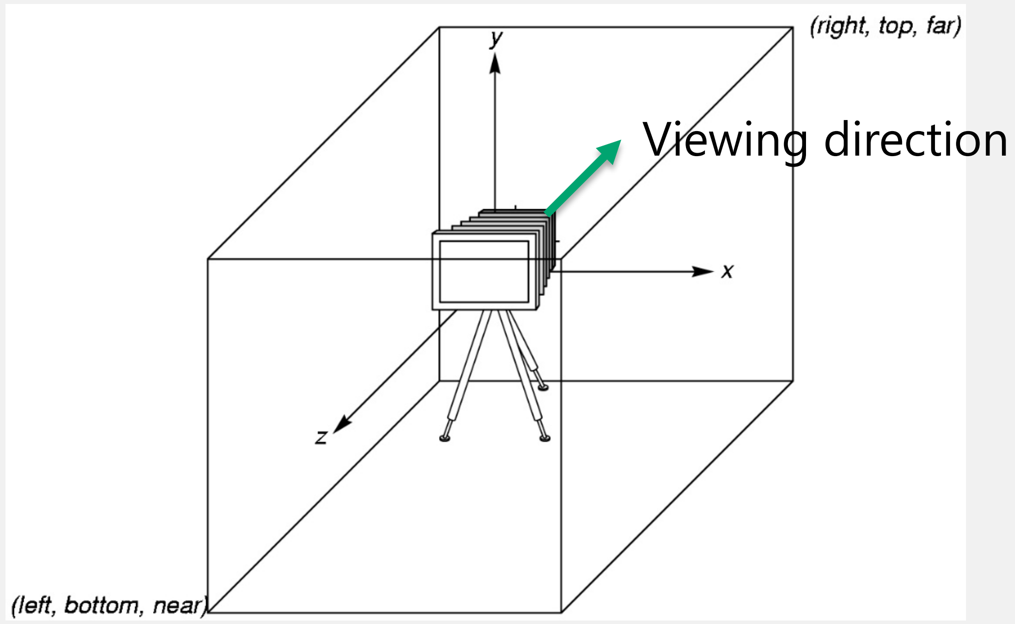
```
#include <gl/glut.h>

void mydisplay();
void init();

float vertices[] = {
  0.5f, 0.5f, -0.5f, 0.5f, -0.5f,-0.5f,
  0.5f, 0.5f, -0.5f,-0.5f,  0.5f,-0.5f,
};

// ...

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(1.0f, 0.0f, 0.0f);
  glEnableClientState(GL_VERTEX_ARRAY);
  glVertexPointer(2, GL_FLOAT, 0, vertices);

  glDrawArrays(GL_TRIANGLES, 0, 6);

  glDisableClientState(GL_VERTEX_ARRAY);

  glFlush();
}
```



Refining Your Program: 3D Graphics

Where is Your Camera?

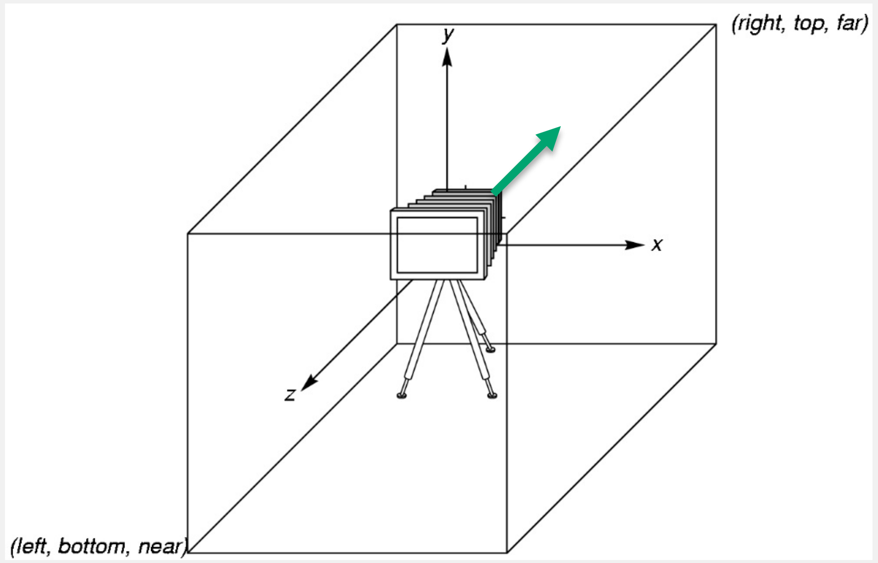
- OpenGL places a camera at the origin in object space pointing in the negative z direction



Set Your Camera

- Setting the intrinsic camera parameter

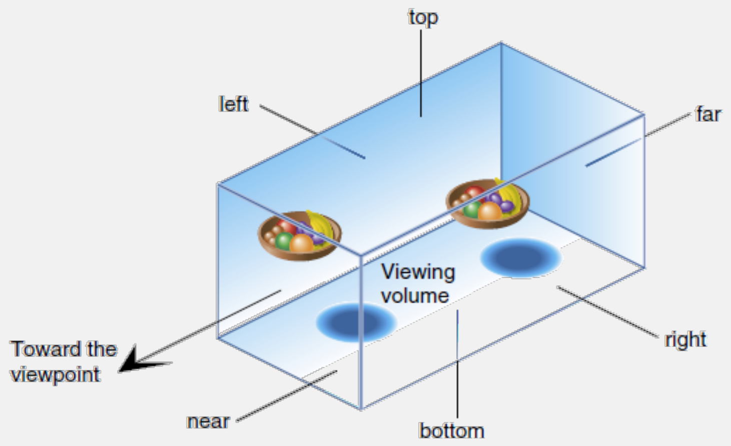
```
void init()
{
  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

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

void mydisplay()
{
  // ...
}
```



Camera Specification – Intrinsic parameters

- glOrtho()
  - OpenGL function for setting intrinsic parameters of OpenGL orthographic camera

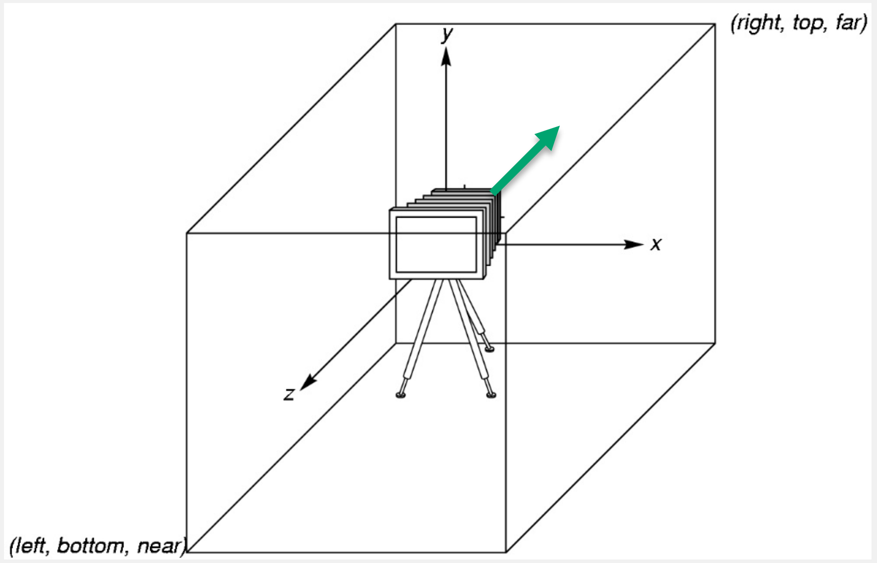
```
// OpenGL function for specifying intrinsic parameters of the OpenGL orthograpic camera
//
// left, right       specify the coordinates for the left and right vertical clipping planes
// bottom, top       specify the coordinates for the bottom and top horizontal clipping planes
// near and far      specify the distances to the nearer and farther depth clipping planes
                     (these values can be negative if plane is to be behind the viewer)

void glOrtho(GLdouble left, GLdouble right,
             GLdouble bottom, GLdouble top,
             GLdouble near, GLdouble far);
```



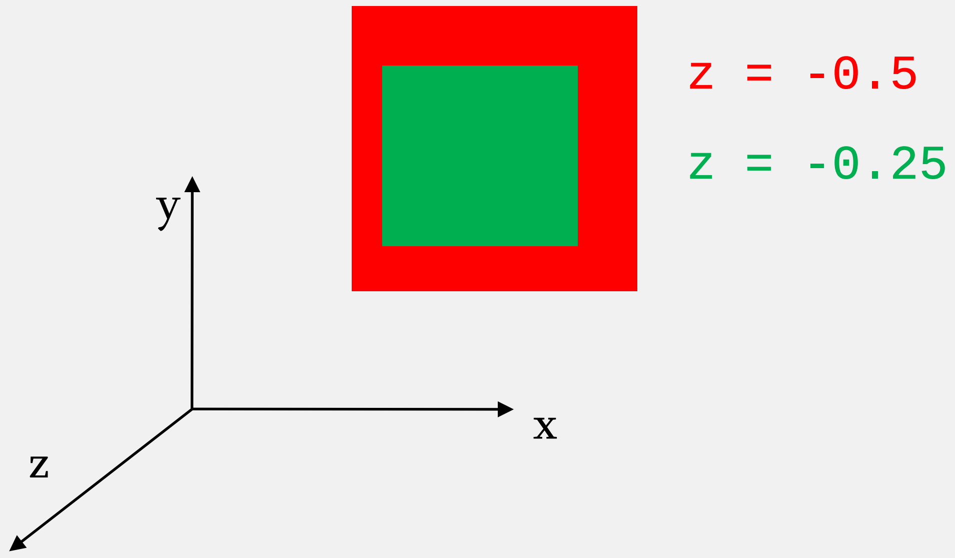
Set Your Camera

- How to make your camera zoom-in?

```
void init()
{
  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(???, ???,
          ???, ???,
          ???, ???);
}

void mydisplay()
{
  // ...
}
```



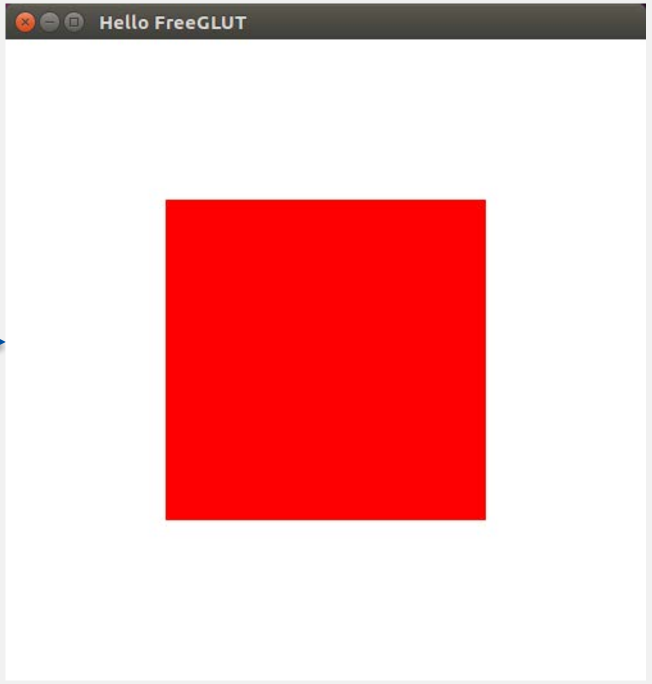
Draw One More Rectangle



```
float small_vertices[] = {
  0.25f, 0.25f, -0.25f,   -0.25f, 0.25f, -0.25f,   -0.25f,-0.25f, -0.25f,
  0.25f, 0.25f, -0.25f,   -0.25f,-0.25f, -0.25f,    0.25f,-0.25f, -0.25f
};
float vertices[] = {
  0.5f, 0.5f, -0.5f,   -0.5f, 0.5f, -0.5f,   -0.5f,-0.5f, -0.5f,
  0.5f, 0.5f, -0.5f,   -0.5f,-0.5f, -0.5f,    0.5f,-0.5f, -0.5f,
};

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(0.0f, 1.0f, 0.0f);
  glEnableClientState(GL_VERTEX_ARRAY);
  glVertexPointer(3, GL_FLOAT, 0, small_vertices);
  glDrawArrays(GL_TRIANGLES, 0, 6);
  glDisableClientState(GL_VERTEX_ARRAY);

  glColor3f(1.0f, 0.0f, 0.0f);
  glEnableClientState(GL_VERTEX_ARRAY);
  glVertexPointer(3, GL_FLOAT, 0, vertices);
  glDrawArrays(GL_TRIANGLES, 0, 6);
  glDisableClientState(GL_VERTEX_ARRAY);

  glFlush();
}
```

Draw One More Rectangle

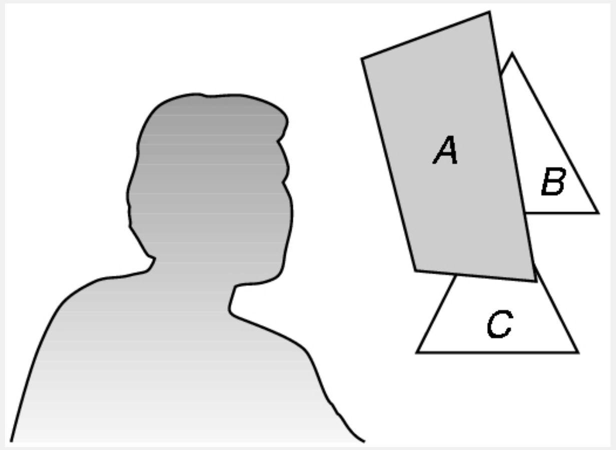
```
float small_vertices[] = {
  0.25f, 0.25f, -0.25f,   -0.25f, 0.25f, -0.25f,   -0.25f,-0.25f, -0.25f,
  0.25f, 0.25f, -0.25f,   -0.25f,-0.25f, -0.25f,    0.25f,-0.25f, -0.25f
};
float vertices[] = {
  0.5f, 0.5f, -0.5f,   -0.5f, 0.5f, -0.5f,   -0.5f,-0.5f, -0.5f,
  0.5f, 0.5f, -0.5f,   -0.5f,-0.5f, -0.5f,    0.5f,-0.5f, -0.5f,
};

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(0.0f, 1.0f, 0.0f);
  glEnableClientState(GL_VERTEX_ARRAY);
  glVertexPointer(3, GL_FLOAT, 0, small_vertices);
  glDrawArrays(GL_TRIANGLES, 0, 6);
  glDisableClientState(GL_VERTEX_ARRAY);

  glColor3f(1.0f, 0.0f, 0.0f);
  glEnableClientState(GL_VERTEX_ARRAY);
  glVertexPointer(3, GL_FLOAT, 0, vertices);
  glDrawArrays(GL_TRIANGLES, 0, 6);
  glDisableClientState(GL_VERTEX_ARRAY);

  glFlush();
}
```



What's wrong?

Hidden-Surface Removal

- We want to see only those surfaces in front of other surfaces
- OpenGL uses a *hidden-surface* method called the *z*-buffer algorithm that saves depth information as objects are rendered so that only the front objects appear in the image



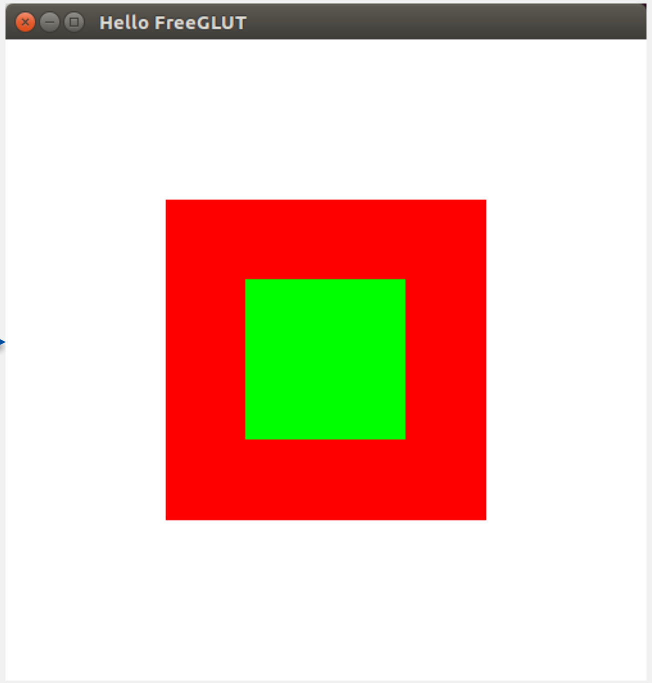
Using *z*-buffer algorithm

- The algorithm uses an extra buffer, the z-buffer, to store depth information as geometry travels down the pipeline
- It requires the followings
  - Inform to GLUT that you will use the 'Depth Test'
    - `glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA | GLUT_DEPTH)`
  - Enable the 'Depth Test'
    - `glEnable(GL_DEPTH_TEST)`
  - Clear the 'Depth buffer'
    - `glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)`

Using the *z*-buffer algorithm

- Enable depth test

```
int main(int argc, char* argv[])
{
  // ...
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA | GLUT_DEPTH);
  // ...
}

void init()
{
  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
  glEnable(GL_DEPTH_TEST);
  // ...
}

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  // draw two rectangles
}
```



Using the *z*-buffer algorithm

- Switch the order of the code segments
  - Draw the **red** rectangle, then draw the **green** rectangle
  - Draw the **green** rectangle, then draw the **red** rectangle
- What is happening?

Discussion

- Change the window size
- Change the window position
- Change the caption of your program
- What is the meaning of ‘`glutDisplayFunc(display)`’?
- How can I handle keyboard/mouse inputs?

Assignments

- Read Chapter 1, 2, 3
  - Especially, read chap. 3 carefully in order to understand how to use GLUT.

- Change the position of your camera by using keyboard inputs
  - See `glutKeyboardFunc(…)` in Chapter 3 & the GLUT reference (PDF file)