

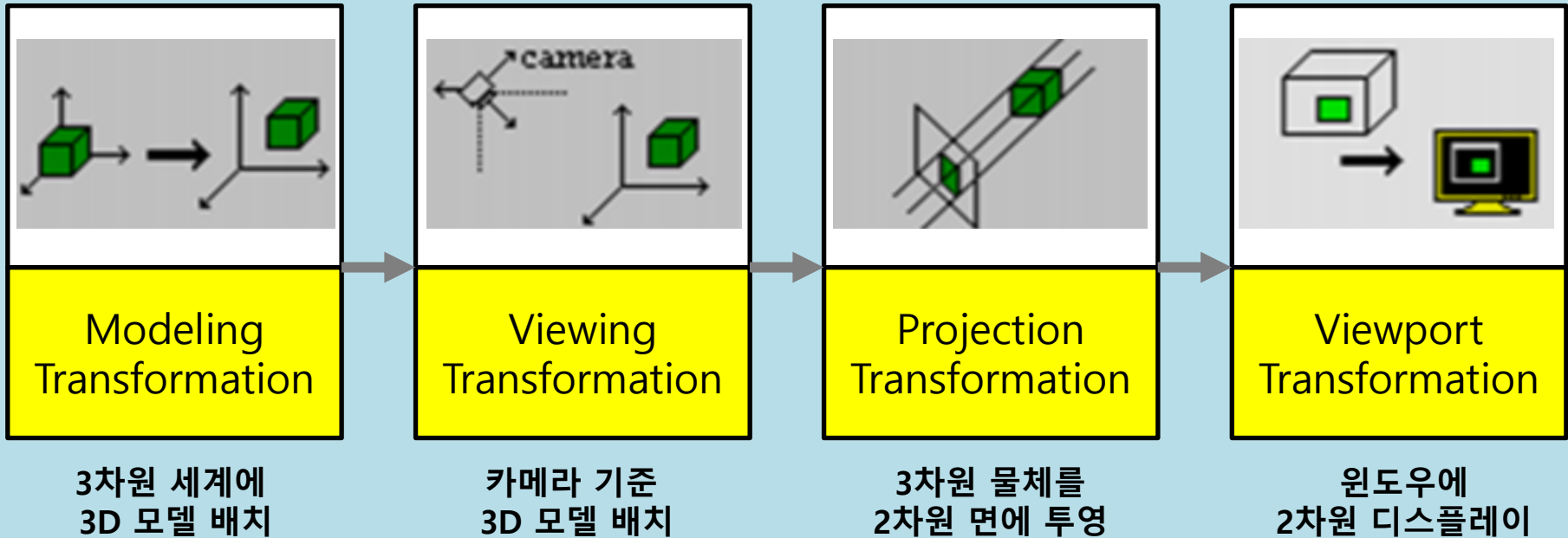
디지털 그래픽스 [10주차]

Viewing Transformation (2)

- Viewport Transformation

Perspective Transformation

- 3차원 모델을 2차원 모니터에 디스플레이 하는 과정



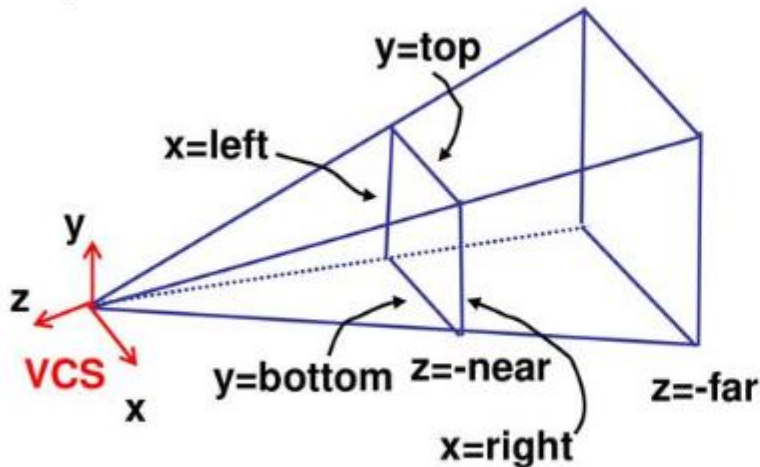
Modeling transform Viewing transform

Projection transform

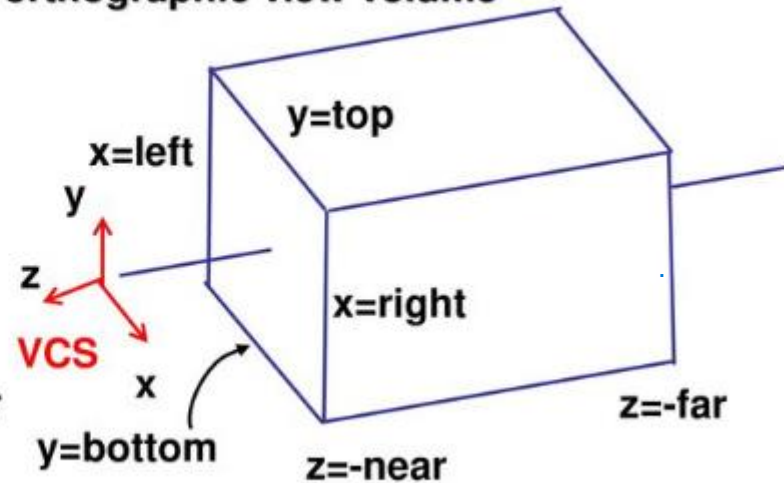
View Volume

- Perspective View Volume : 원근감
- Orthographic View Volume : 평면도 등에 사용

perspective view volume

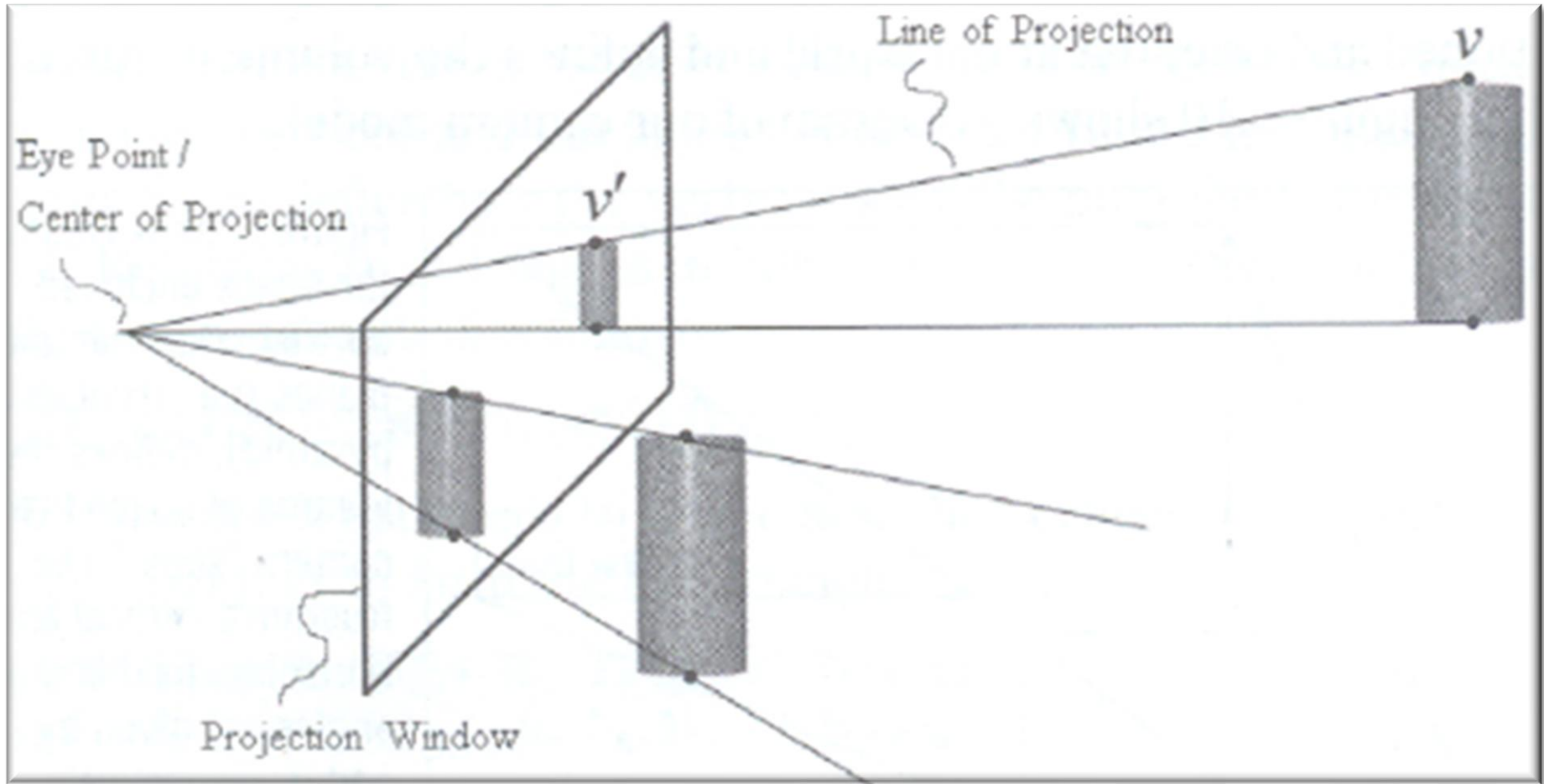


orthographic view volume



Perspective Illusion

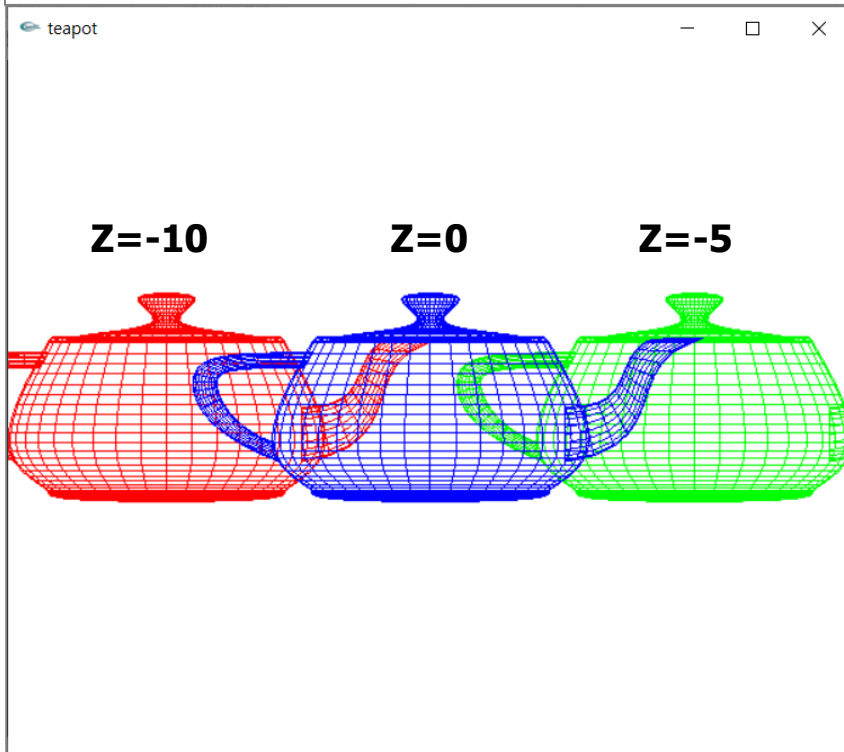
- 원근감



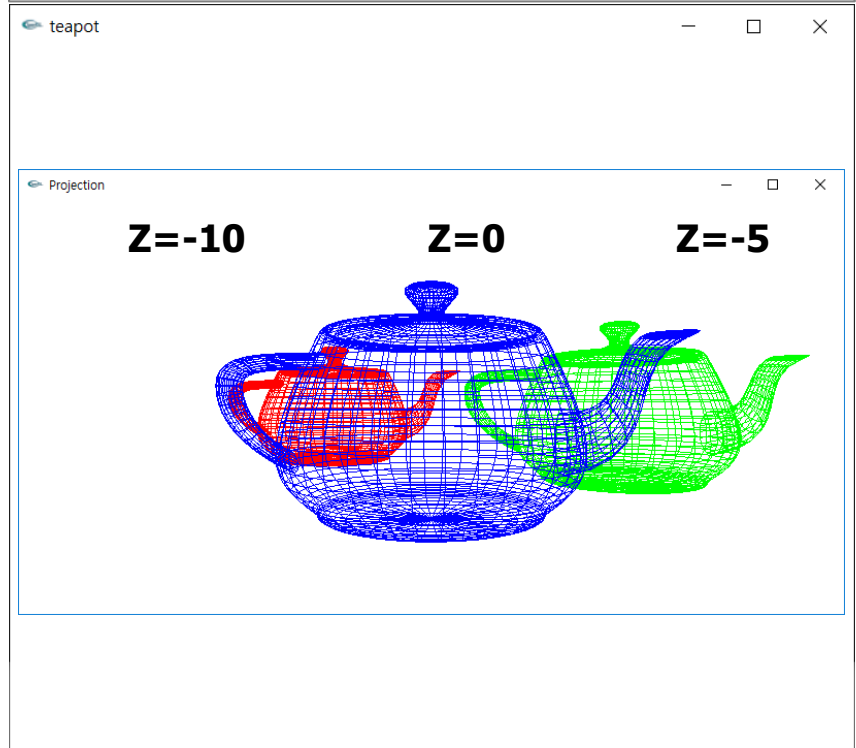
parallel vs. perspective projection

- `glOrtho(..)` vs. `glFrustum(..)`

`glOrtho(-8, 8, -8, 8, 1.5, 30)`

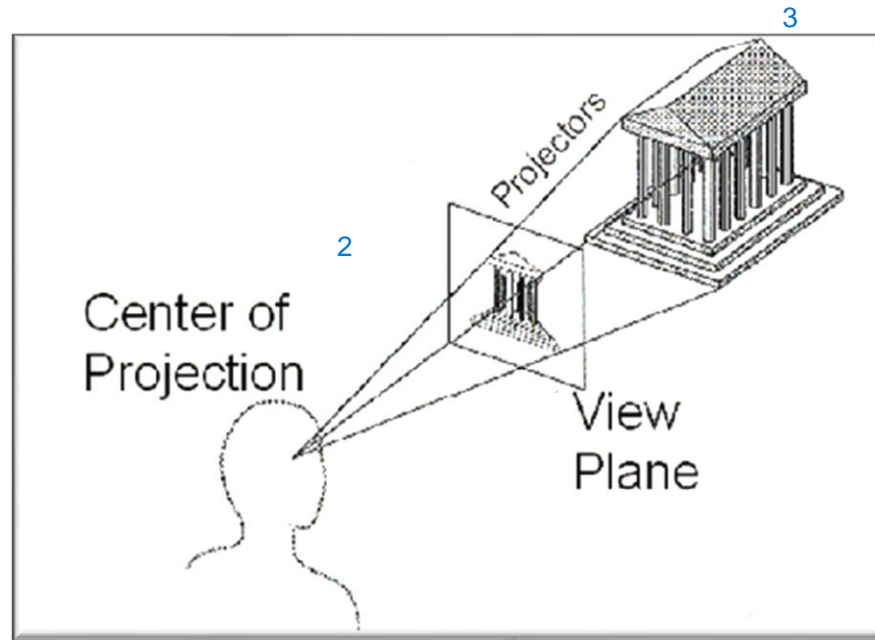


`glFrustum(-8, 8, -8, 8, 1.5, 30)`



Projection

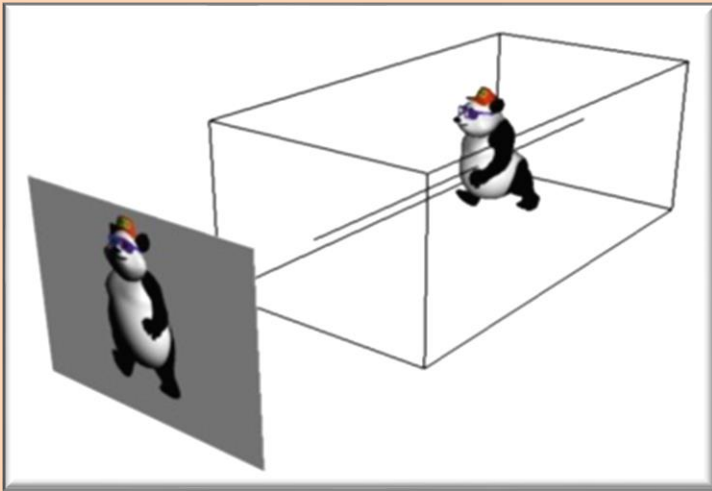
- **In Computer Graphics**
 - Map 3D coordinates to 2D coordinates
 - Based on the camera model



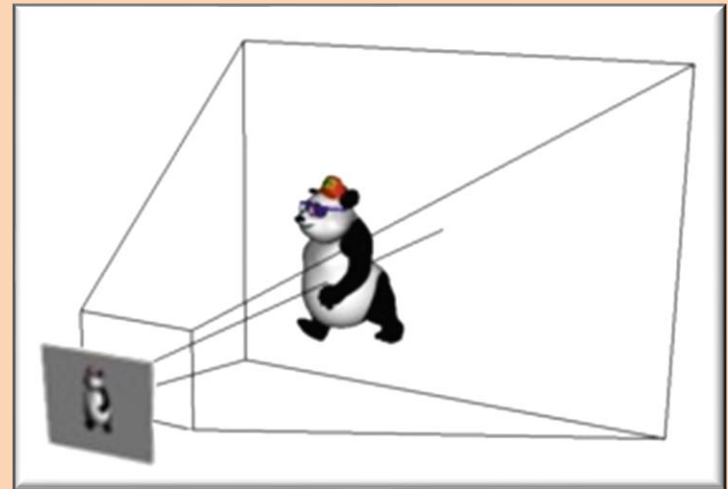
The way of projections

Geometric Projection

Parallel ==orthographic

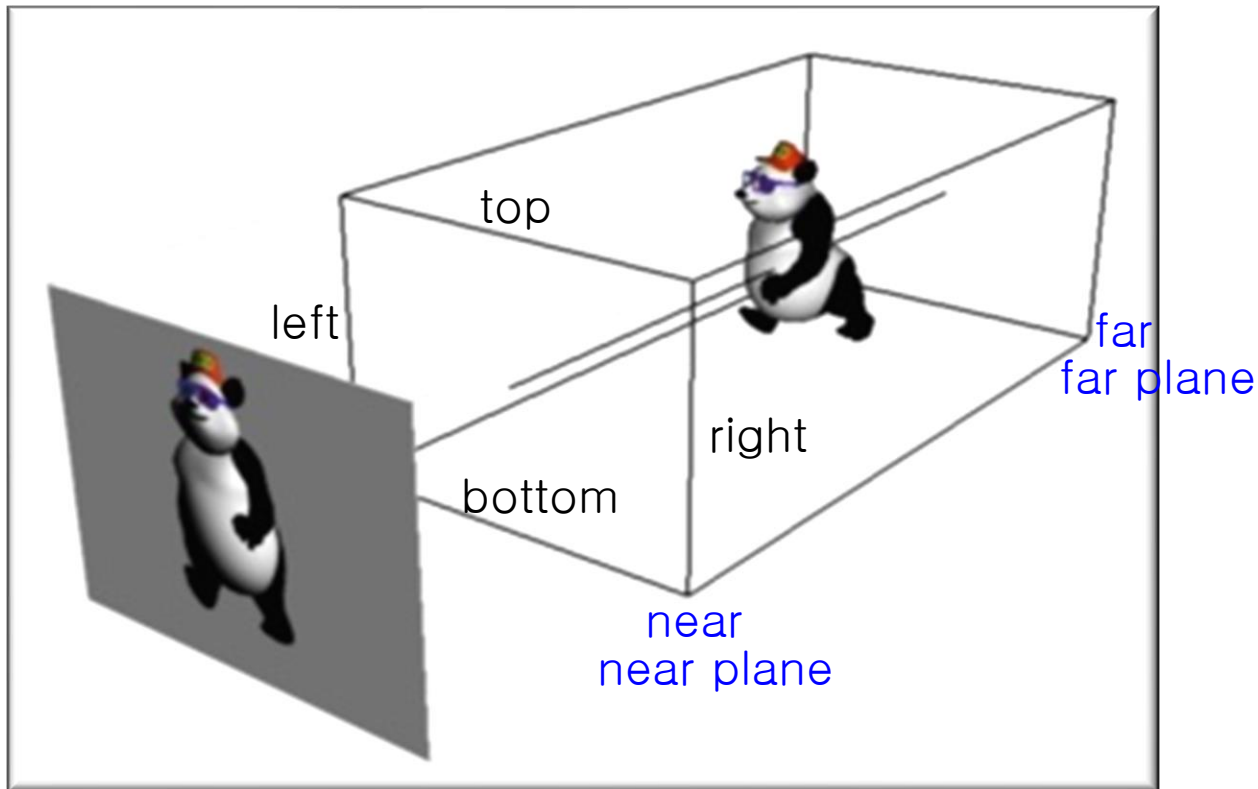


Perspective



Orthographic View Volume

- **View Volume parameters**
 - left, right, bottom, top, near, far

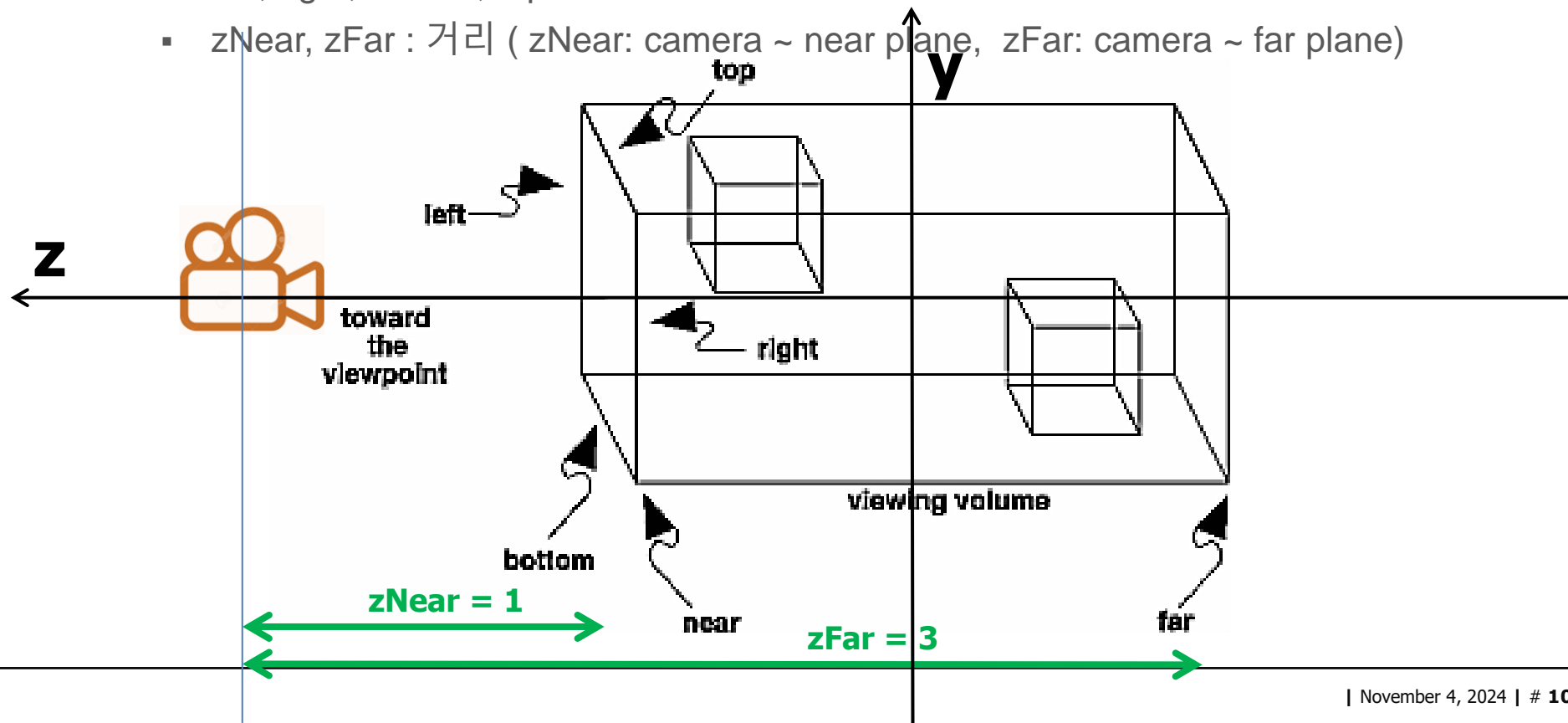


OpenGL Orthographic Projection

glOrtho(..)

OpenGL: Orthogonal Projection

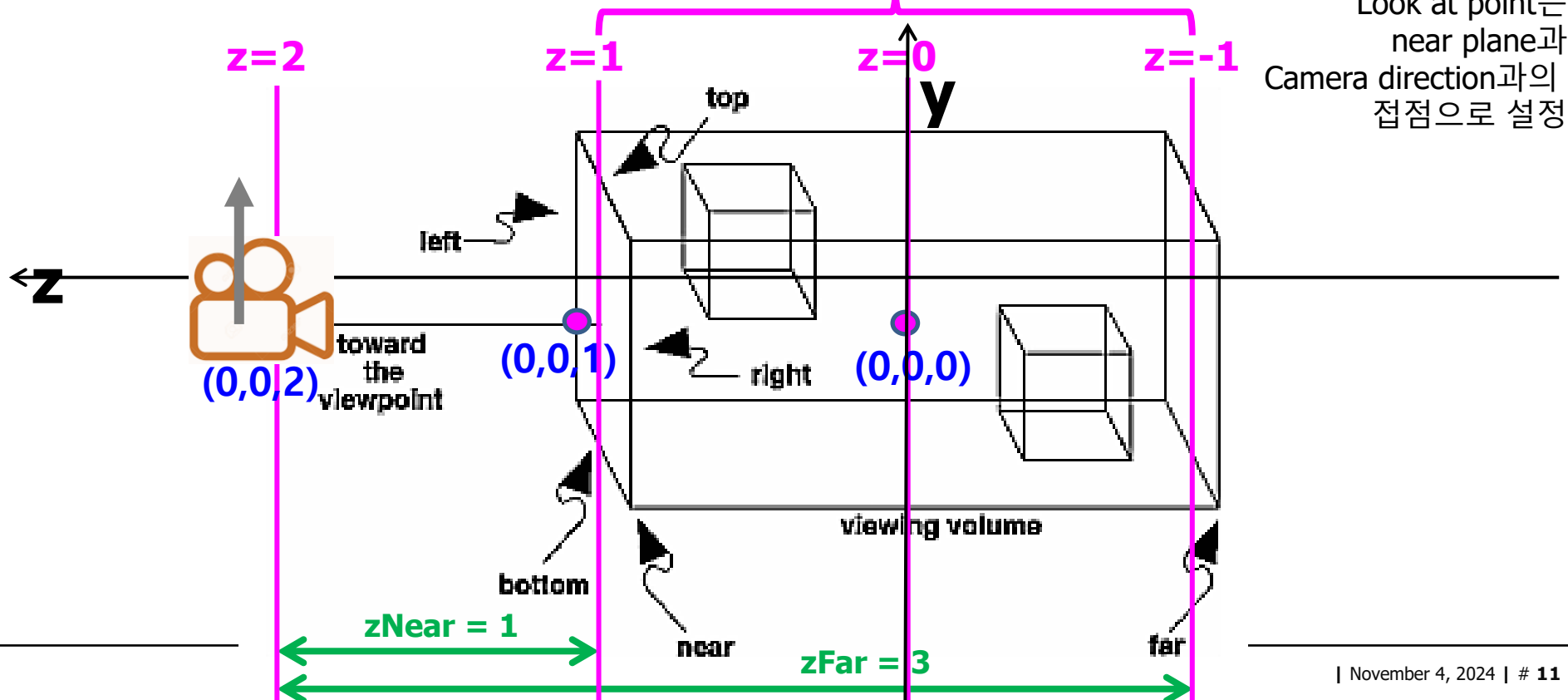
- void **glOrtho**(GLdouble left, GLdouble right,
GLdouble bottom, GLdouble top,
GLdouble zNear, GLdouble zFar)
 - left, right, bottom, top : 좌표
 - zNear, zFar : 거리 (zNear: camera ~ near plane, zFar: camera ~ far plane)



Orthogonal Projection

```
glOrtho(-1.0, 1.0, -1.0, 1.0, 1.0, 3.0);  
gluLookAt( 0.0, 0.0, 2.0,  
           0.0, 0.0, 1.0,  
           0.0, 1.0, 0.0);
```

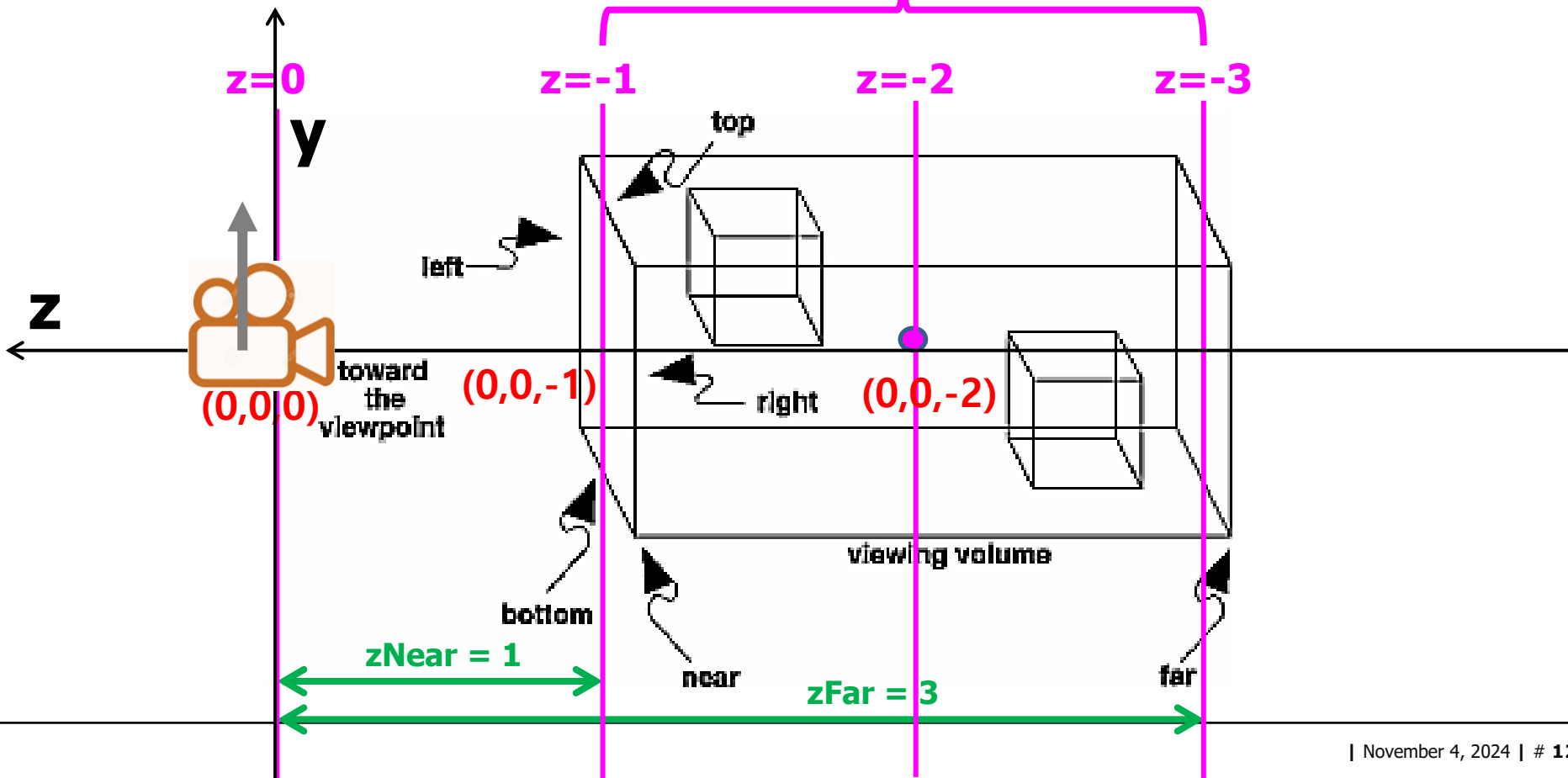
- void **glOrtho**(GLdouble left, GLdouble right,
 GLdouble bottom, GLdouble top,
 GLdouble zNear, GLdouble zFar)
 - glOrtho(**-1.0, 1.0, -1.0, 1.0, 1.0, 3.0**);



Orthogonal Projection

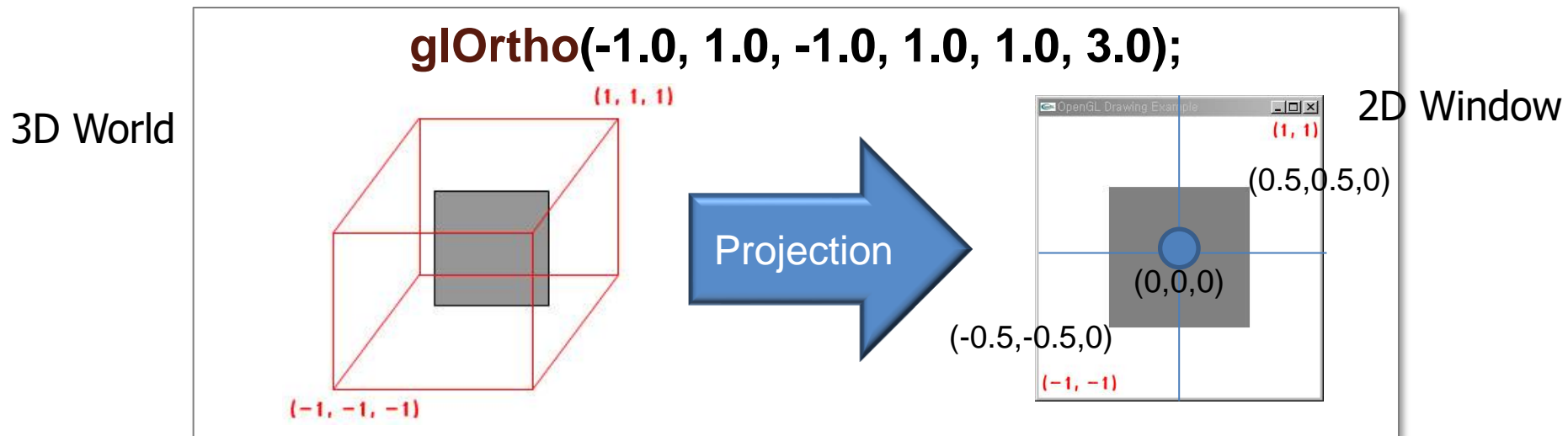
```
glOrtho(-1.0, 1.0, -1.0, 1.0, 1.0, 3.0);  
gluLookAt( 0.0, 0.0, 0.0,   
           0.0, 0.0, -1.0, <<  
           0.0, 1.0, 0.0);
```

- void **glOrtho**(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar)
 - glOrtho(-1.0, 1.0, -1.0, 1.0, 1.0, 3.0);



How the view volume works

- void **glOrtho**(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar)
 - glOrtho(**-1.0, 1.0**, **-1.0, 1.0**, **1.0, 3.0**);
- Rectangle of size 1 at the origin

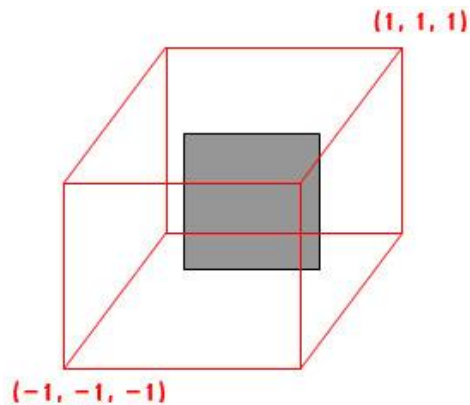


How the view volume works

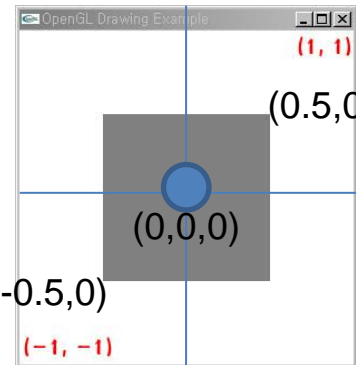
- Variations of view volume

3D World

`glOrtho(-1.0, 1.0, -1.0, 1.0, 1.0, 3.0);`

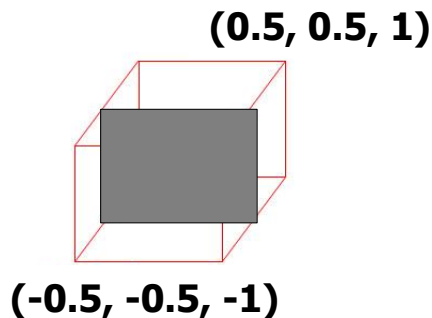


Projection

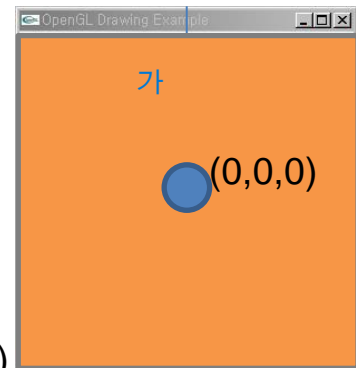


2D Window

`glOrtho(-0.5, 0.5, -0.5, 0.5, 1.0, 3.0);`

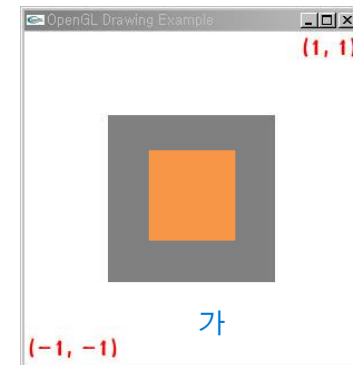
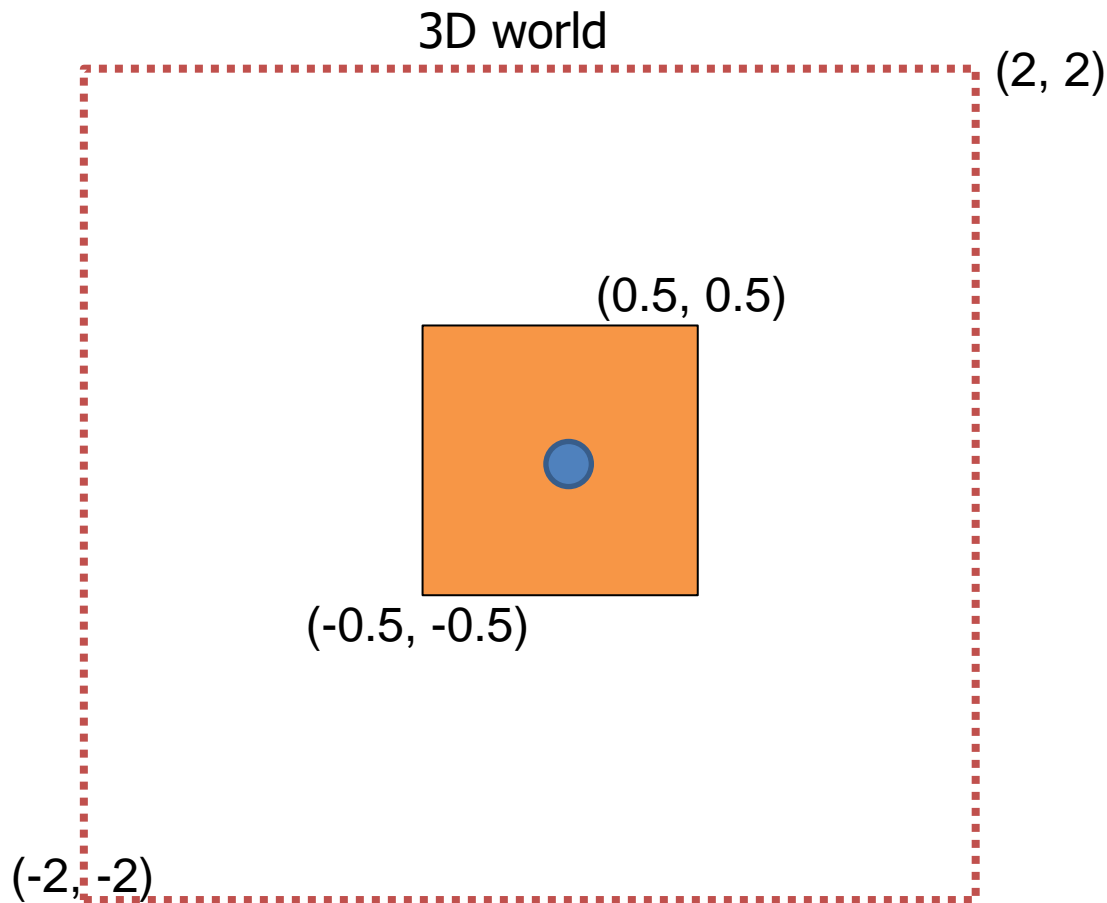


Projection

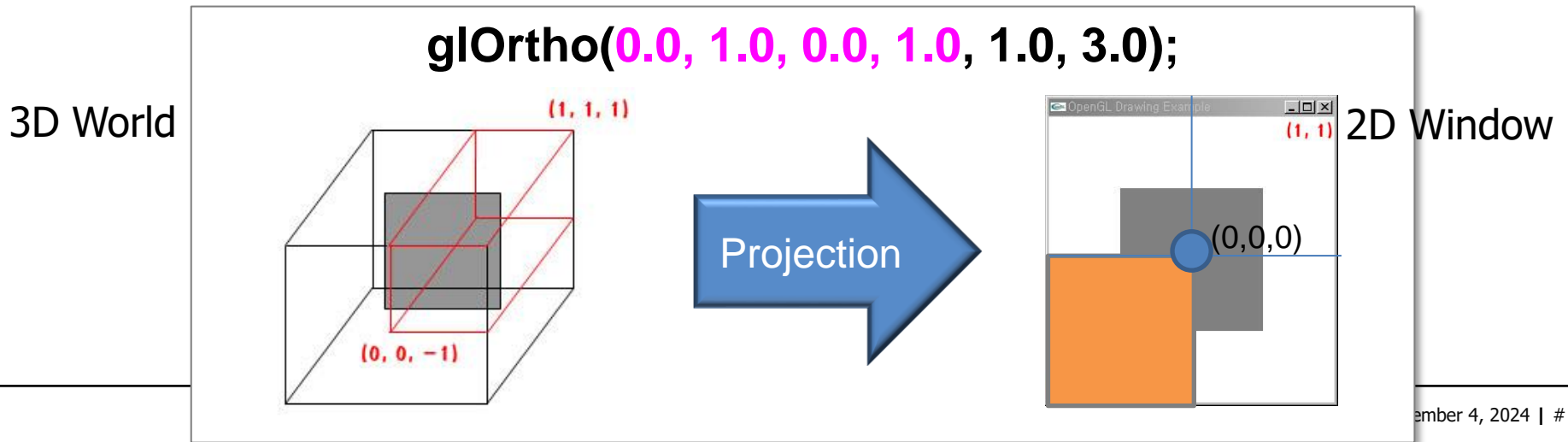
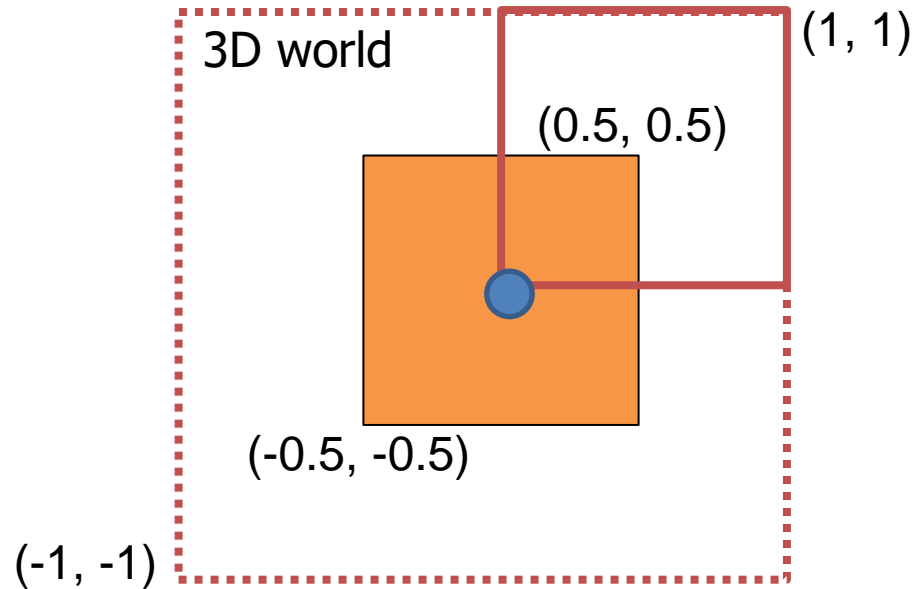


How the view volume works

- view volume
 - `glOrtho(-2.0, 2.0, -2.0, 2.0, 1.0, 3.0);`

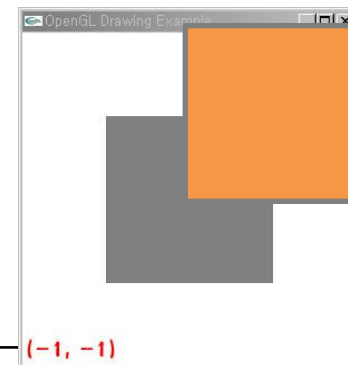
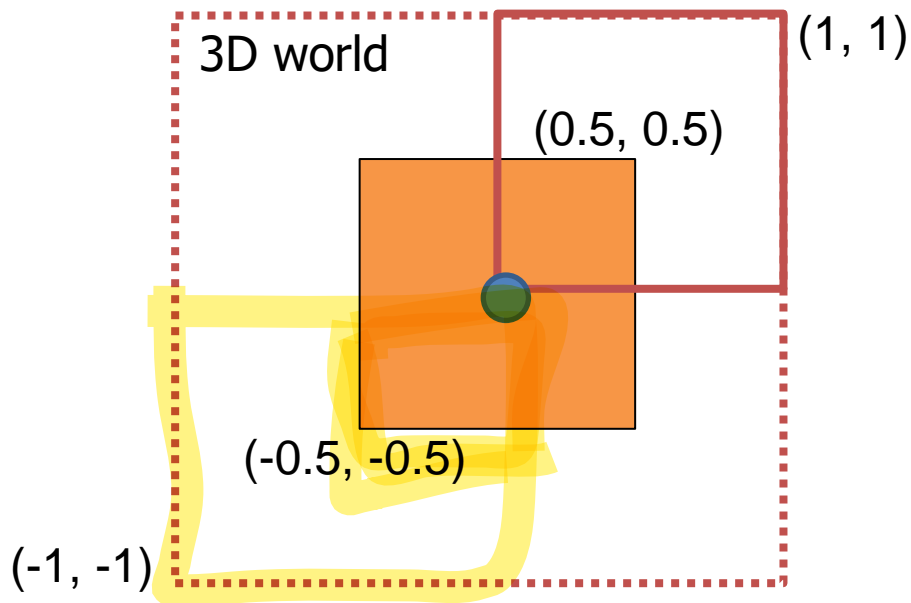


How the view volume works



How the view volume works

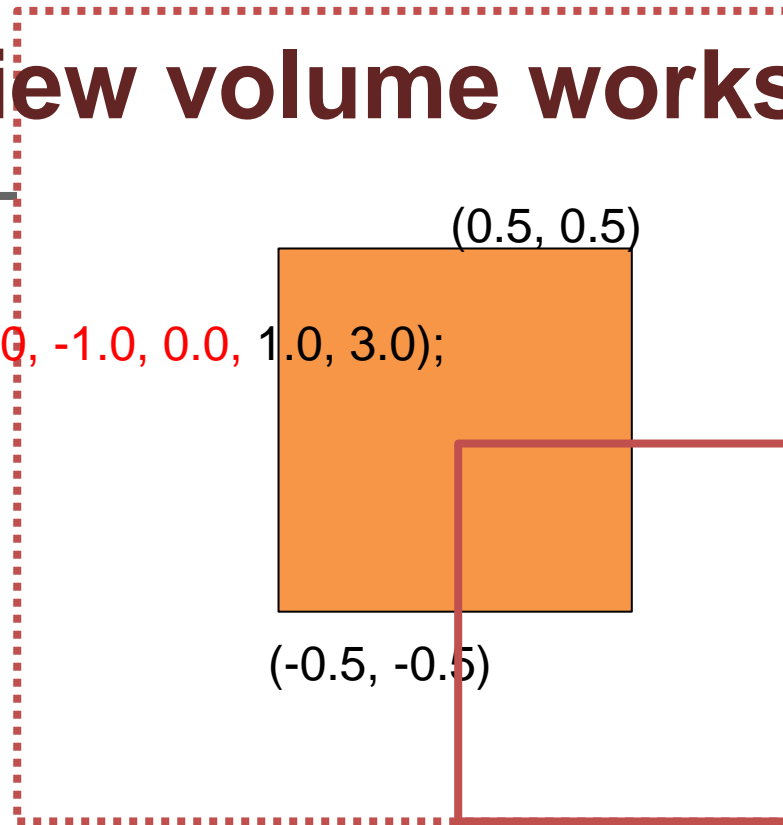
- view volume
 - `glOrtho(-1.0, 0.0, -1.0, 0.0, 1.0, 3.0);`



How the view volume works

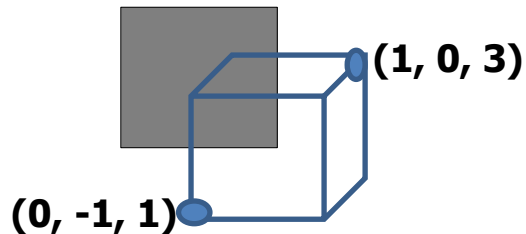
- view volume

- `glOrtho(0.0, 1.0, -1.0, 0.0, 1.0, 3.0);`



3D World

`glOrtho(0.0, 1.0, -1.0, 0.0, 1.0, 3.0);`



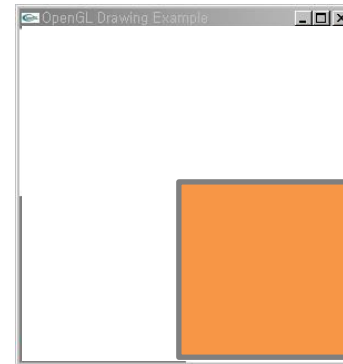
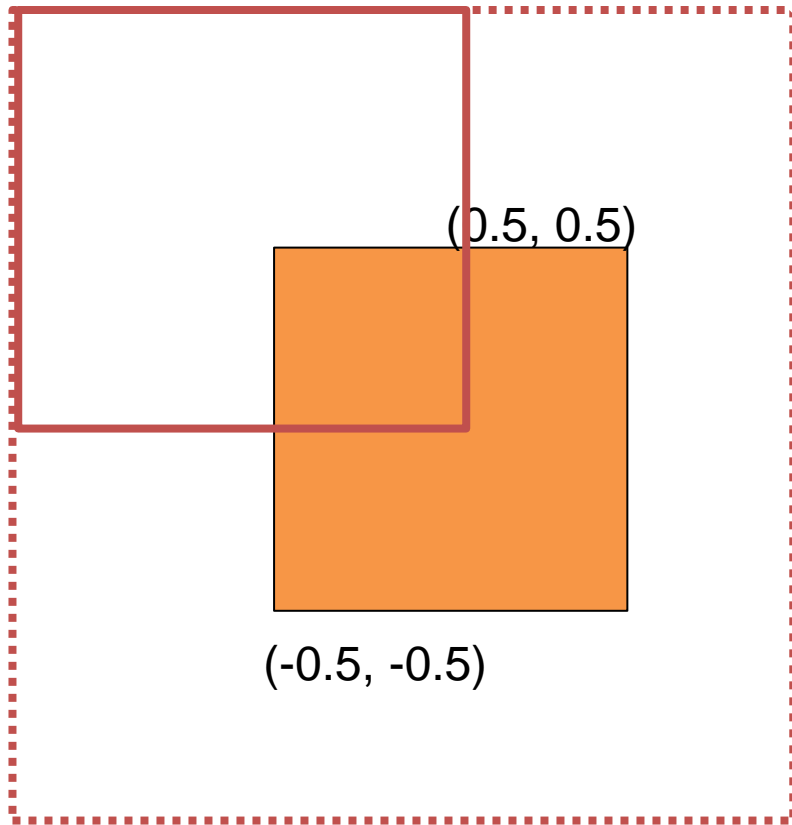
Projection



2D Window

How the view volume works

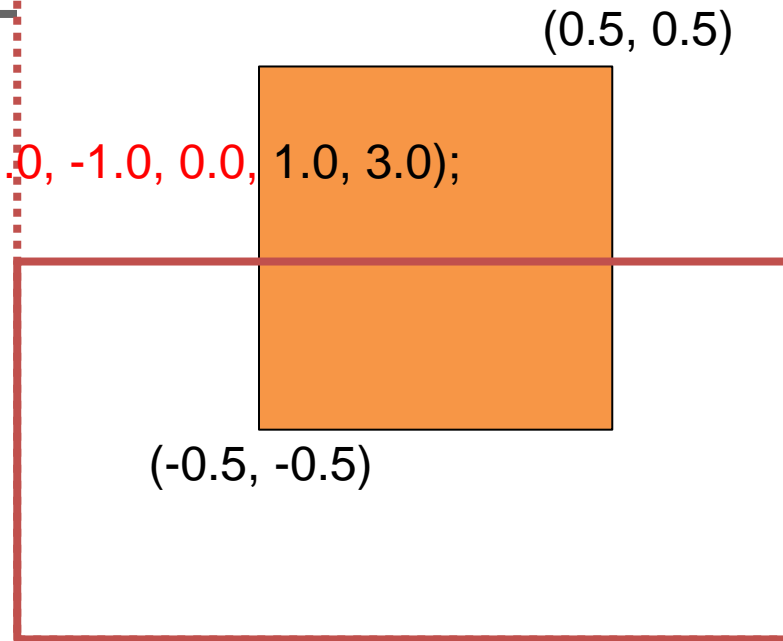
- view volume
 - `glOrtho(-1.0, 0.0, 0.0, 1.0, 1.0, 3.0);`



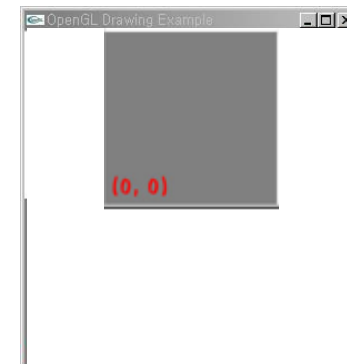
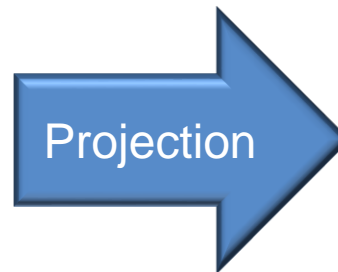
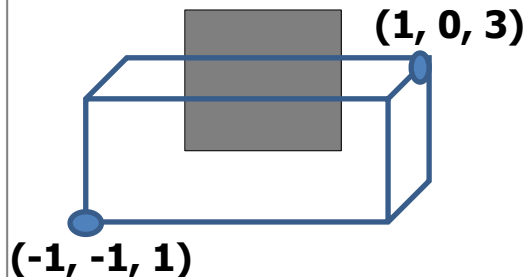
How the view volume works

- view volume

- `glOrtho(-1.0, 1.0, -1.0, 0.0, 1.0, 3.0);`

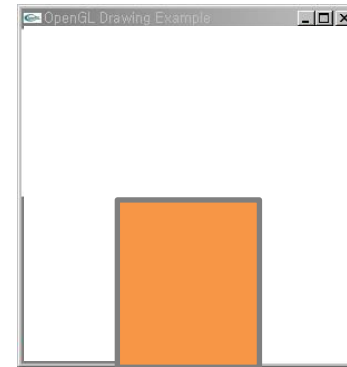
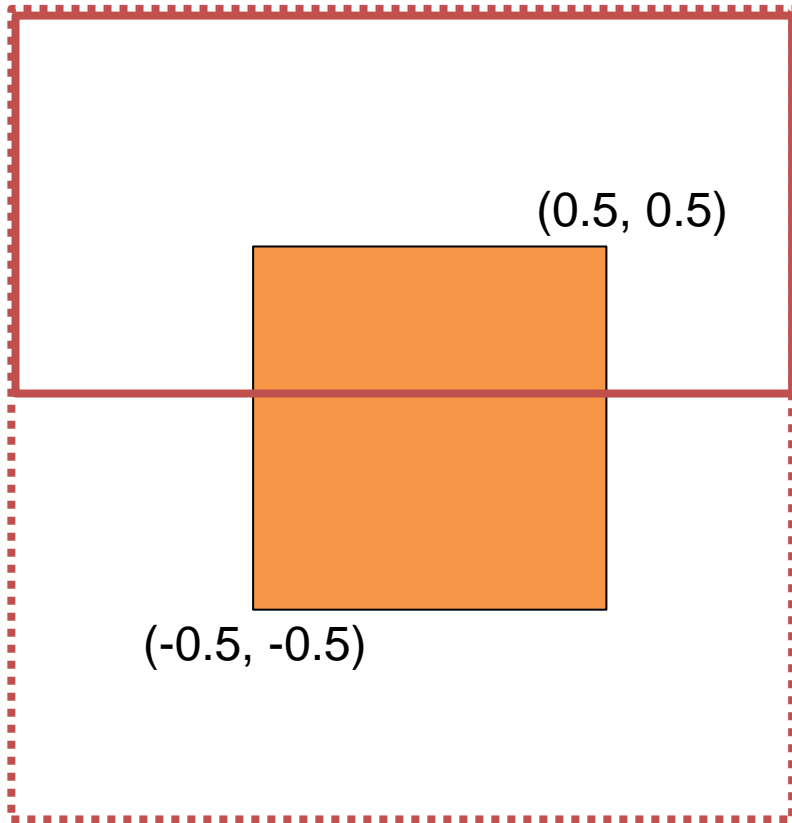


`glOrtho(-1.0, 1.0, -1.0, 0.0, 1.0, 3.0);`



How the view volume works

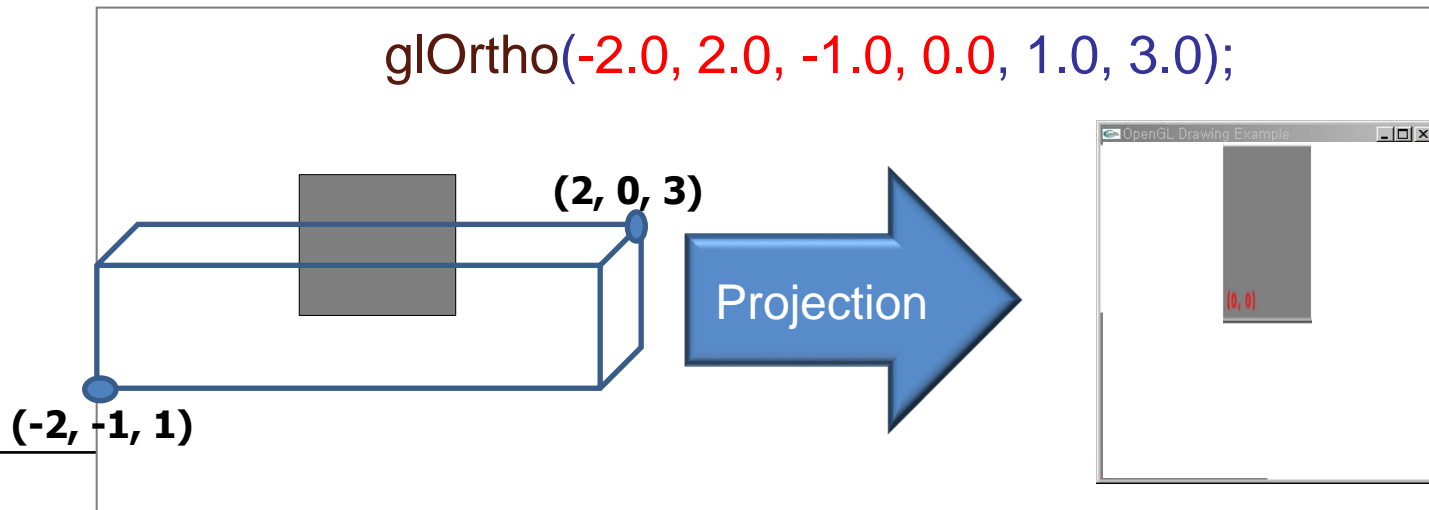
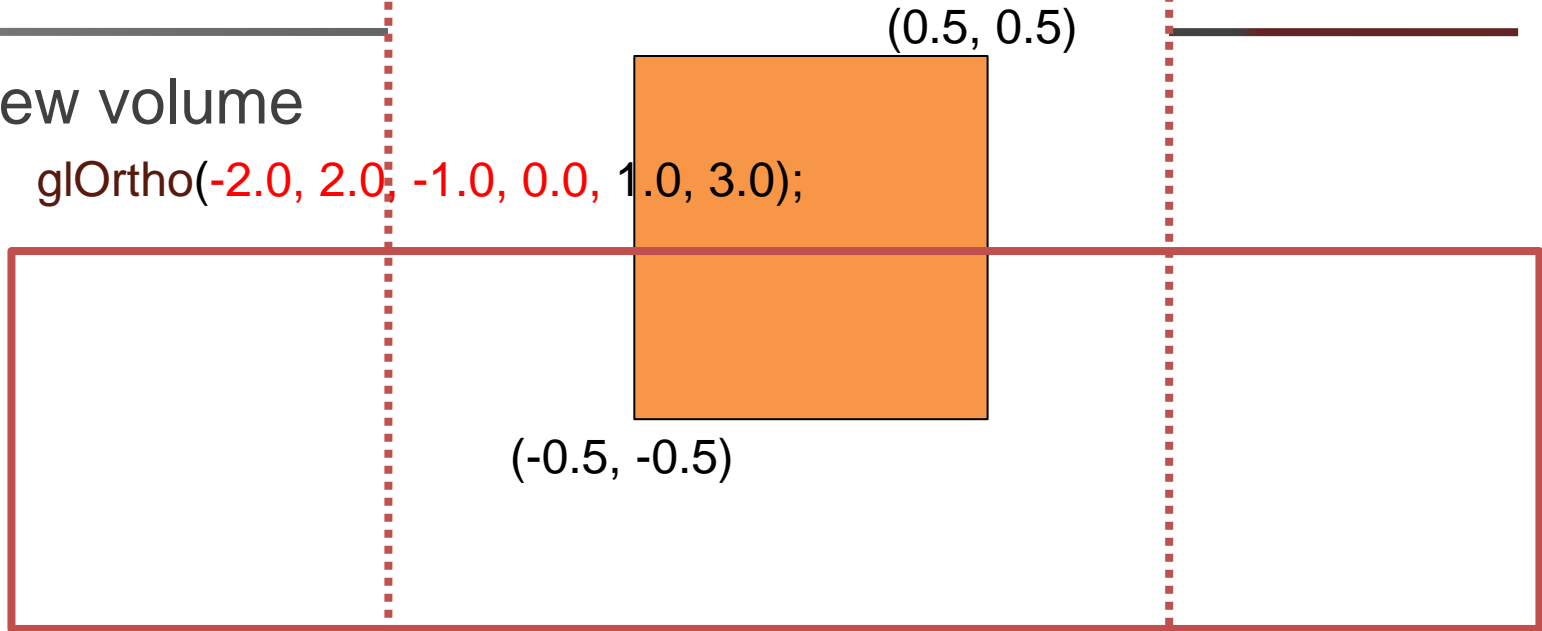
- view volume
 - `glOrtho(-1.0, 1.0, 0.0, 1.0, 1.0, 3.0);`



How the view volume works

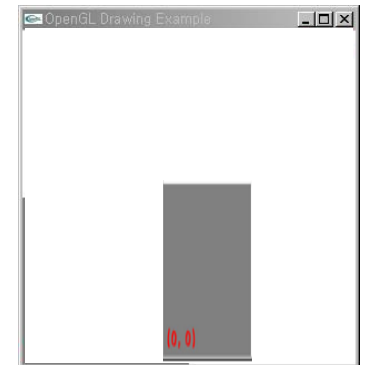
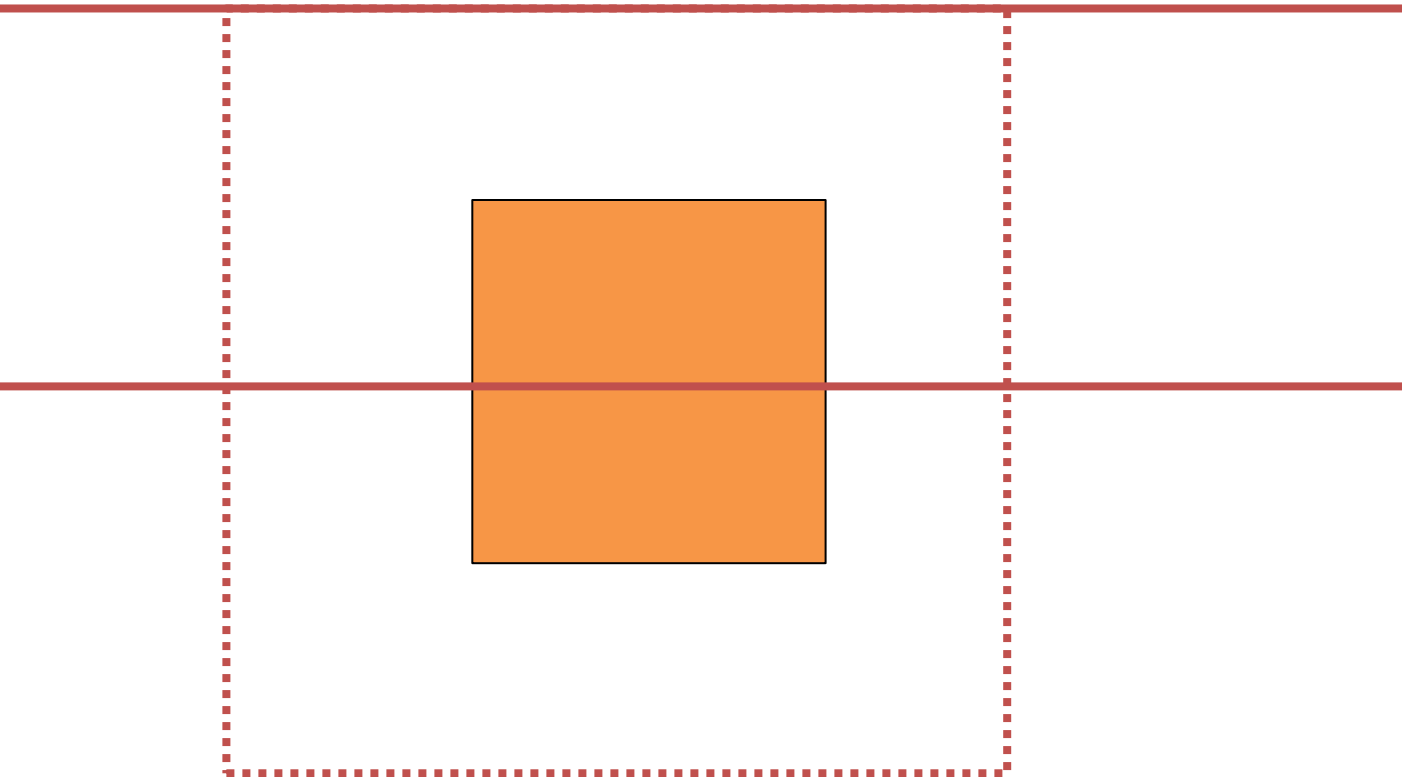
- view volume

- `glOrtho(-2.0, 2.0, -1.0, 0.0, 1.0, 3.0);`



How the view volume works

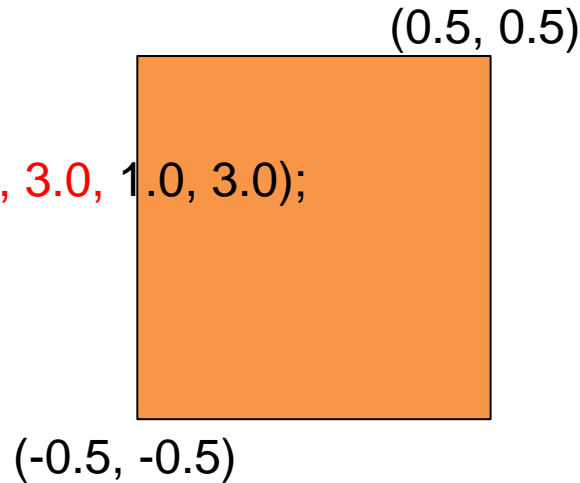
- view volume
 - `glOrtho(-2.0, 2.0, 0.0, 1.0, 1.0, 3.0);`



How the view volume works

- view volume

- `glOrtho(-3.0, 3.0, -3.0, 3.0, 1.0, 3.0);`

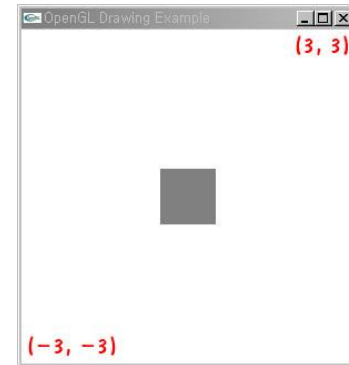


$(3, 3, 3)$

`glOrtho(-3.0, 3.0, -3.0, 3.0, 1.0, 3.0);`

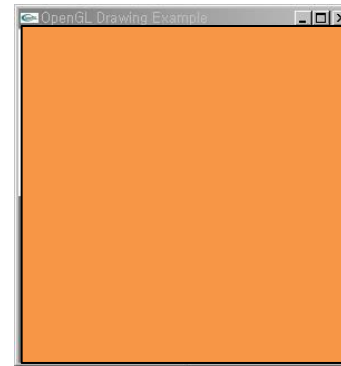
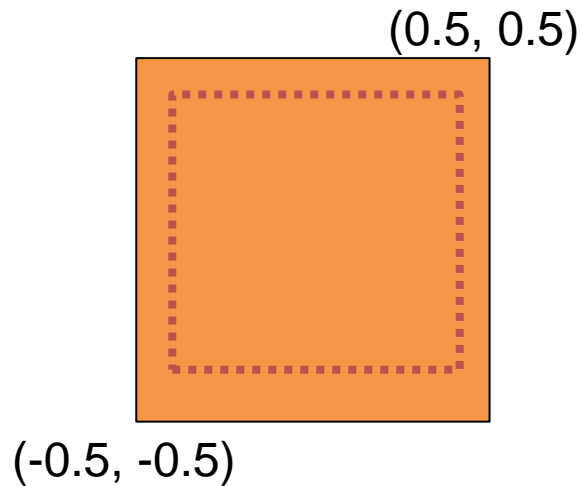
$(-3, -3, 1)$

Projection



How the view volume works

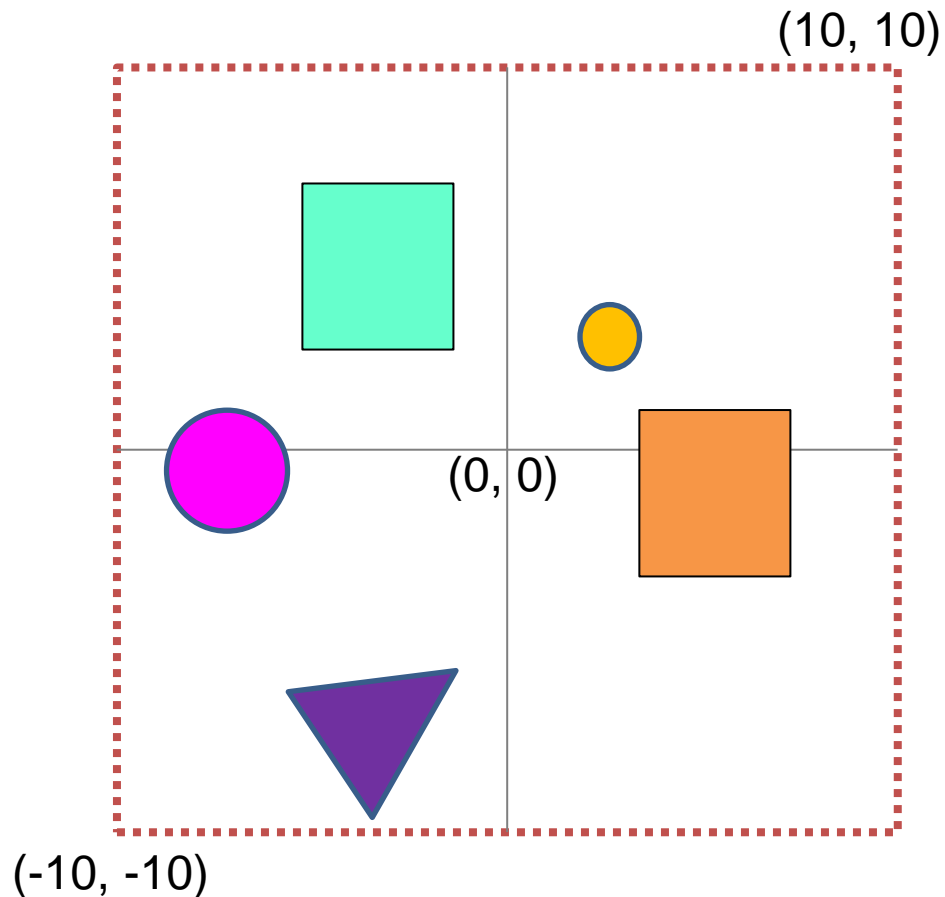
- view volume
 - `glOrtho(-0.3, 0.3, -0.3, 0.3, 1.0, 3.0);`



How the view volume works

- view volume
 - `glOrtho(-10, 10, -10, 10, 1.0, 3.0);`

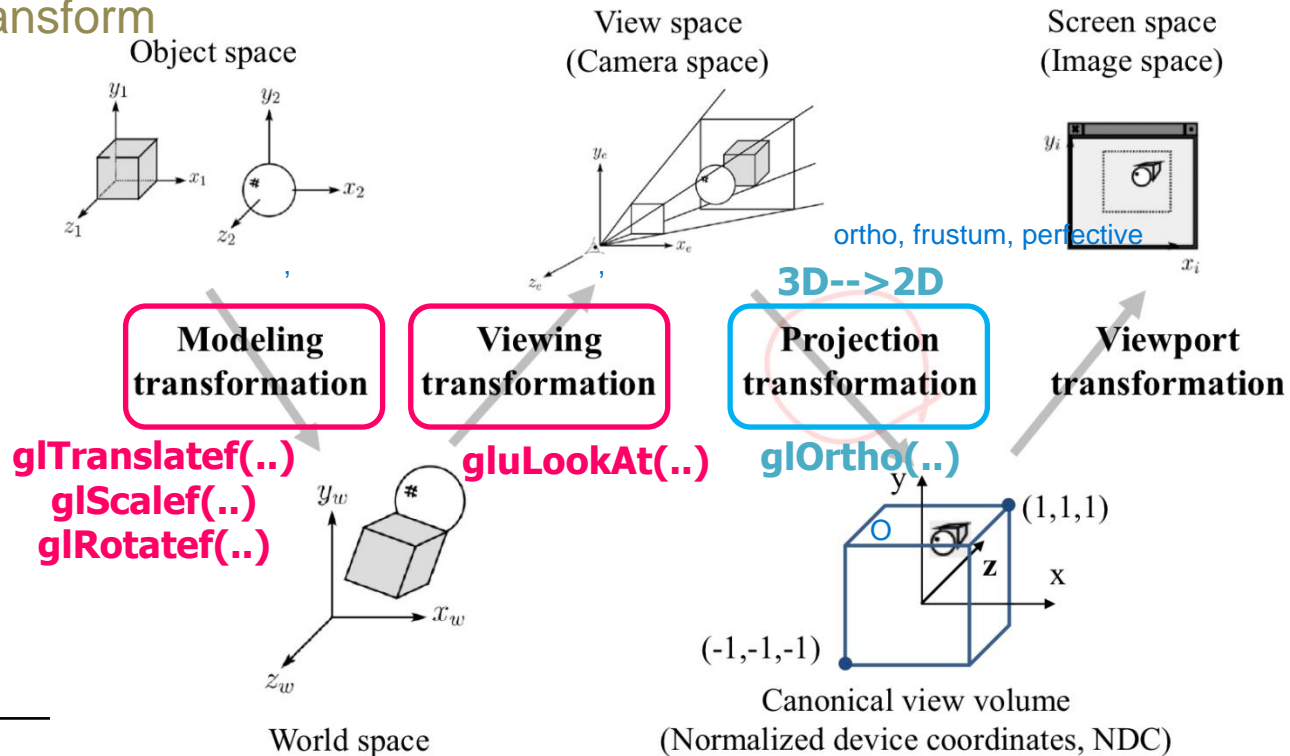
-1, 1, -1, 1



Transformation

- **Matrix mode**

- 행렬 모드, 행렬 종류
- `void glMatrixMode(mode)`
 - modeling transform, viewing transform(camera)
 - projection transform



glOrtho()

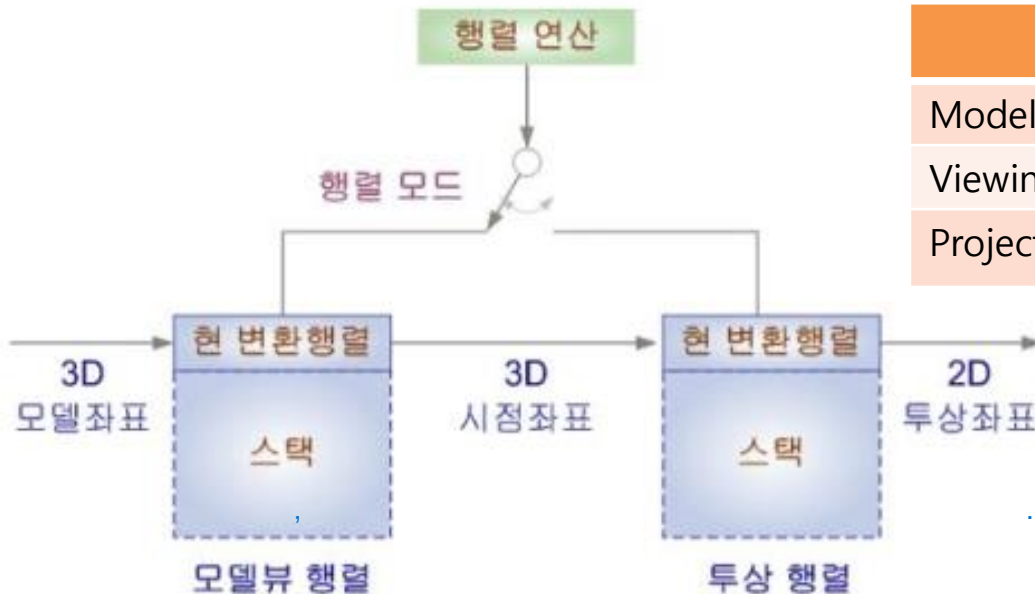
- Matrix mode

- 행렬 모드, 행렬 종류

- `void glMatrixMode(mode)`

가 가 .

- modeling transform, viewing transform(camera) --> **GL_MODELVIEW**
- projection transform --> **GL_PROJECTION** ORTHO, PERSPECTIVE, FRUSTUM



	OpenGL f.	Matrix Mode
Modeling Transform	glTranslatef(..)	GL_MODELVIEW
Viewing Transform	gluLookAt(..)	GL_MODELVIEW
Projection Transform	glOrtho(..)	GL_PROJECTION

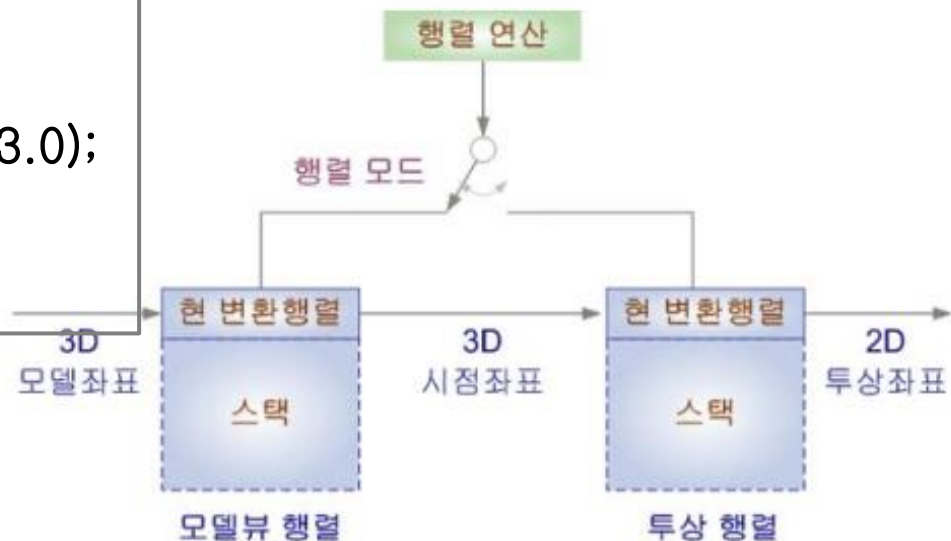
glOrtho()

- **Matrix mode**

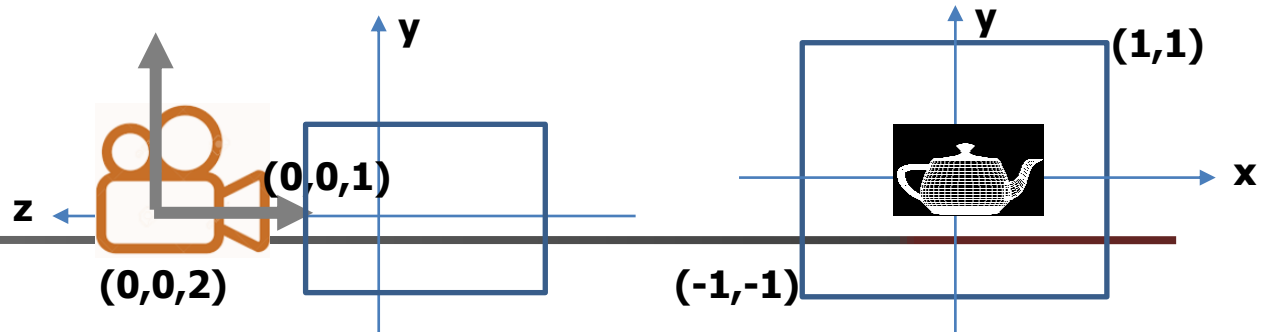
- 조작하고자 하는 행렬 선택
- `void glMatrixMode(mode)`
 - `GL_MODELVIEW` : model transform, viewing transform(camera)
 - `GL_PROJECTION` : projection transform

- **Current Transformation matrix (현 변환 행렬)**

```
glMatrixMode(GL_PROJECTION);  
glLoadIdentity();           identity  
glOrtho(-1.0, 1.0, -1.0, 1.0, 1.0, 3.0);  
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity();           identity
```



glOrtho()



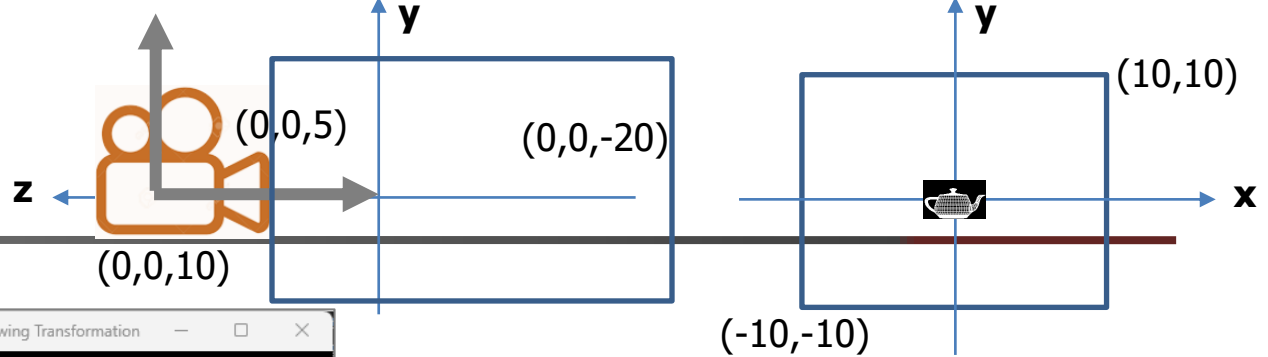
```
#include <iostream>
#include <gl/glut.h>
#include <gl/glu.h>
using namespace std;

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT |
            GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluLookAt(0.0, 0.0, 2.0, // camera pos
              0.0, 0.0, 1.0, // lookat point
              0.0, 1.0, 0.0); // up vector
    glColor3f(1, 1, 1);
    glutWireTeapot(0.3);
    glFlush();
}
```

```
void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-1.0, 1.0, -1.0, 1.0, 1.0, 3.0);
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA |
                        GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow("Viewing
                     Transformation");
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

glOrtho()



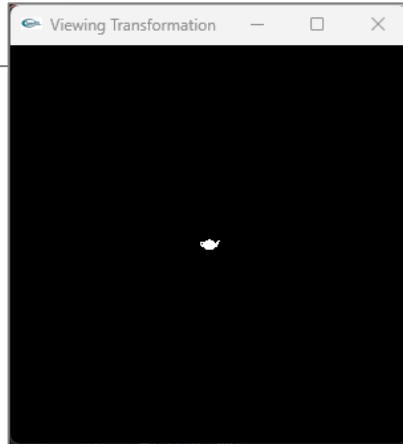
```
#include <iostream>
#include <gl/glut.h>
#include <gl/glu.h>
using namespace std;
```

```
void display(void)
{
```

```
    glClear(GL_COLOR_BUFFER_BIT |
            GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluLookAt(0.0, 0.0, 10.0, // camera pos
              0.0, 0.0, 0.0,   // lookat point
              0.0, 1.0, 0.0);  // up vector
```

```
    glColor3f(1, 1, 1);
    glutWireTeapot(0.3);
    glFlush();
```

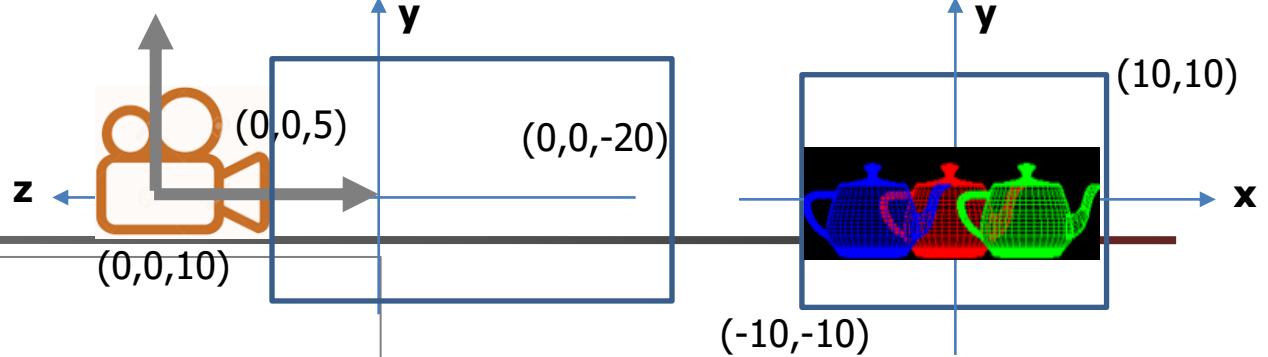
```
}
```



```
void reshape(int w, int h)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-10.0, 10.0, -10.0, 10.0, 5.0, 30.0);
    glMatrixMode(GL_MODELVIEW);
}
```

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA |
                        GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow("Viewing
                      Transformation");
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

glOrtho()



```
#include <iostream>
#include <gl/glut.h>
#include <gl/glu.h>
using namespace std;
```

```
void display(void)
{
```

```
    glClear(GL_COLOR_BUFFER_BIT |
            GL_DEPTH_BUFFER_BIT);
```

```
    glLoadIdentity();
```

```
    gluLookAt(0.0, 0.0, 10.0, 0.0, 0.0, 0.0,
              1.0, 0.0);
```

```
    glColor3f(1, 0, 0);
```

```
    glutWireTeapot(3);
```

```
    glColor3f(0, 1, 0);
```

```
    glTranslatef(5,0,-5);
```

```
    glutWireTeapot(3);
```

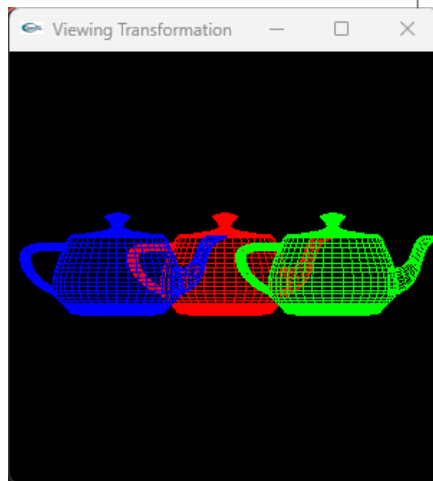
```
    glColor3f(0, 0, 1);
```

```
    glTranslatef(-10,0,-5);
```

```
    glutWireTeapot(3);
```

```
    glFlush();
```

```
}
```



```
void reshape(int w, int h)
```

```
{
```

```
    glMatrixMode(GL_PROJECTION);
```

```
    glLoadIdentity();
```

```
    glOrtho(-10.0, 10.0, -10.0, 10.0, 5.0, 30.0);
```

```
    glMatrixMode(GL_MODELVIEW);
```

```
}
```

```
int main(int argc, char** argv)
```

```
{
```

```
    glutInit(&argc, argv);
```

```
    glutInitDisplayMode(GLUT_RGBA |
```

```
    GLUT_DEPTH | GLUT_SINGLE);
```

```
    glutCreateWindow("Viewing
                        Transformation");
```

```
    glClearColor(0.0, 0.0, 0.0, 0.0);
```

```
    glutReshapeFunc(reshape);
```

```
    glutDisplayFunc(display);
```

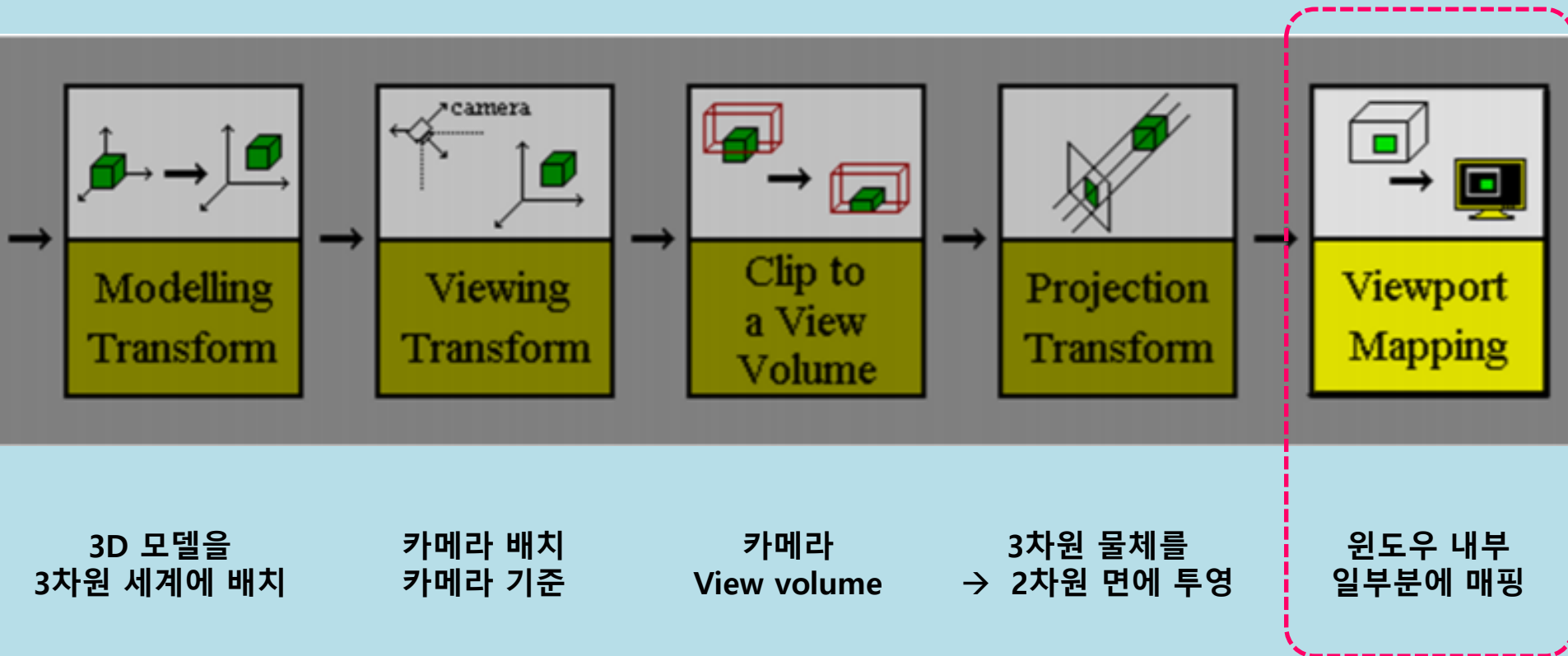
```
    glutMainLoop();
```

```
    return 0;
```

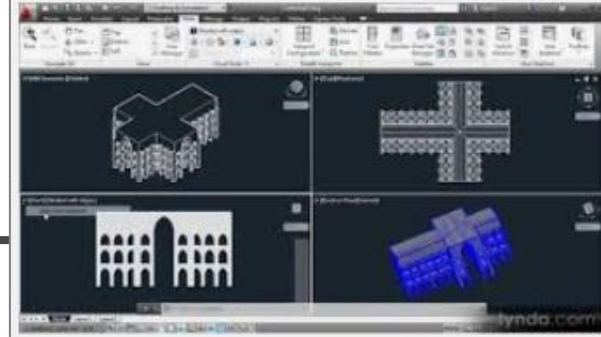
```
}
```


Viewing Transformation

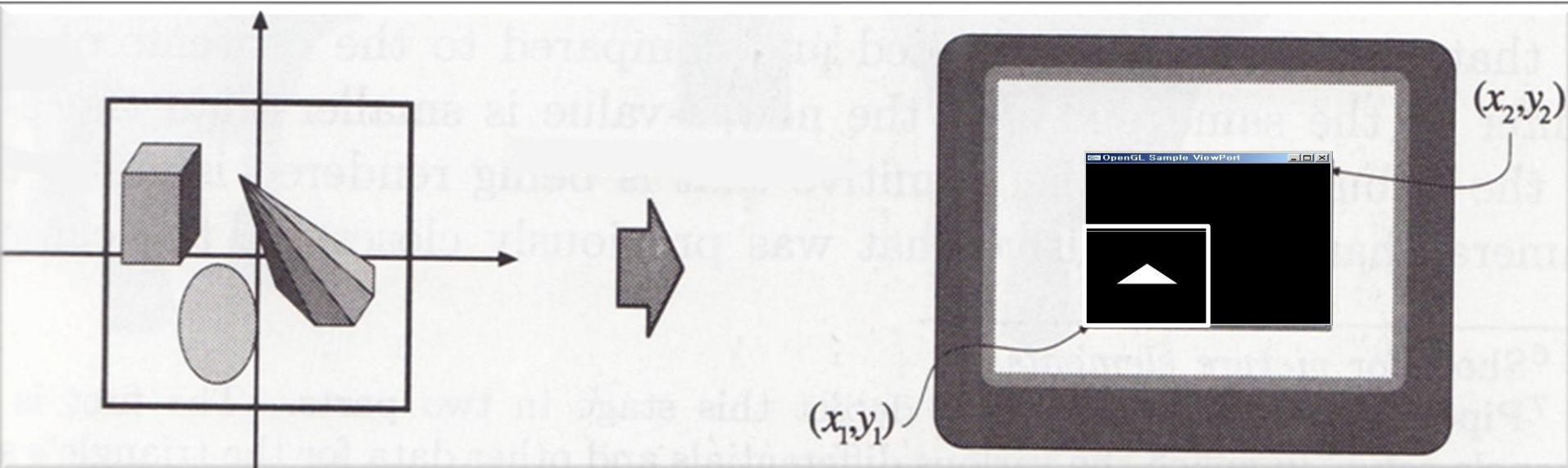
- 3차원 모델을 2차원 모니터에 디스플레이 하는 과정



Viewport Transformation

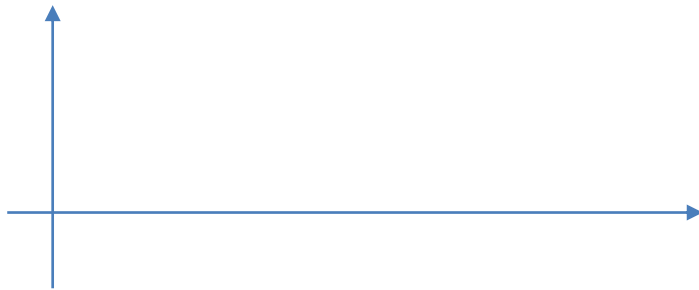


- Translates the viewing coordinates
 - into the device coordinates
- View volume vs. viewport
 - Where we want to see vs. where we want to display



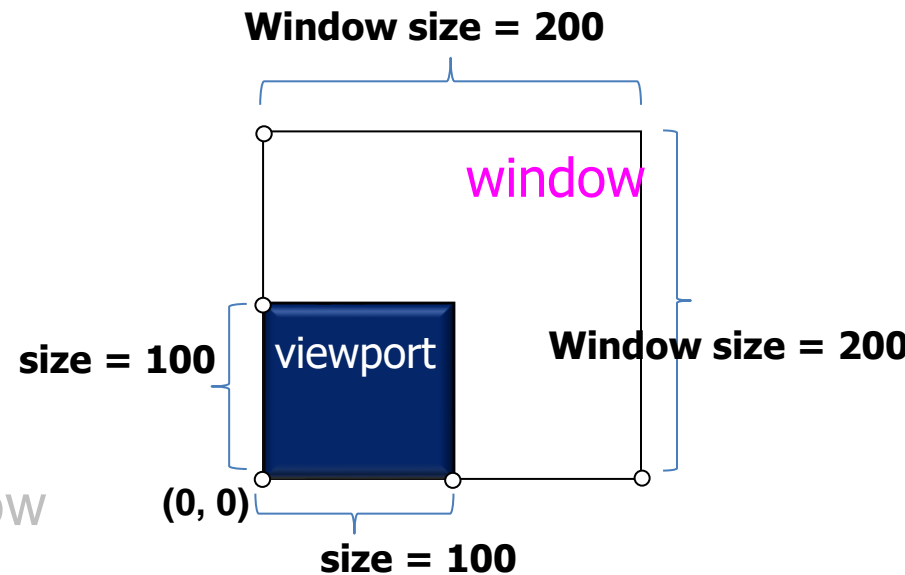
Viewport Transformation

- void **glViewport**(GLint x, GLint y, GLsizei width, GLsizei height)
 - left-low corner coordinate (x, y)
 - viewport size (width, height)



- **Example**

```
// initialize the size of a window  
glutInitWindowSize(200, 200);  
// sets the position and size of a viewport  
glViewport(0, 0, 100, 100); // 기준: left lower corner
```



glViewport()

`glViewport(GLint x, GLint y, GLsizei width, GLsizei height)`

- 위치 : `reshape()`

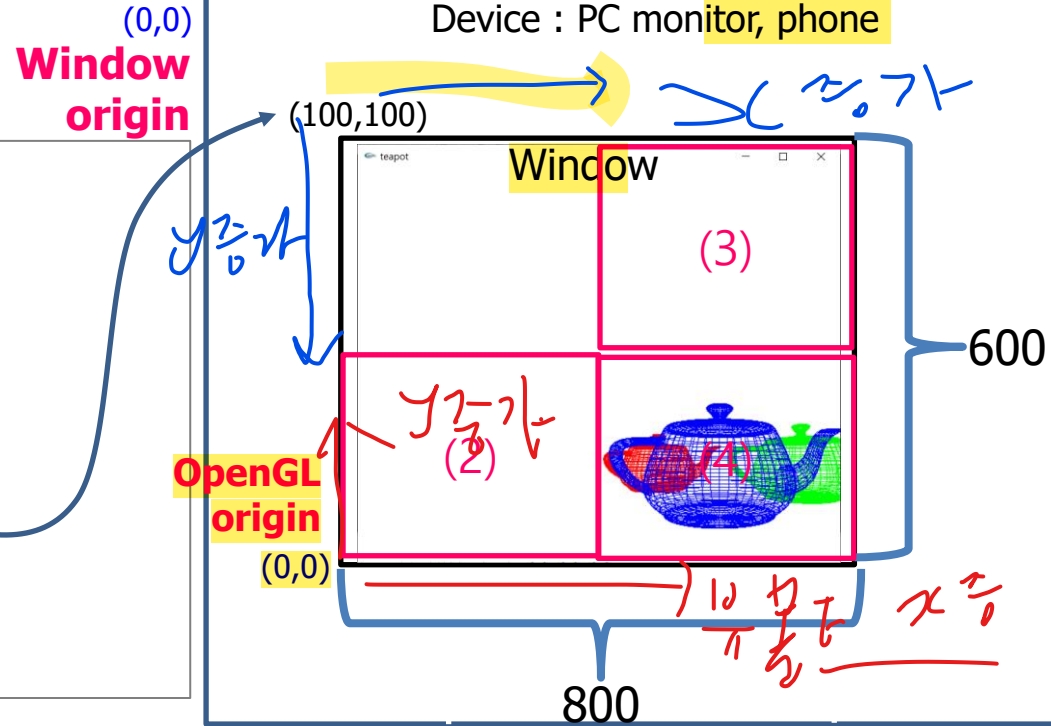
```
#define WIDTH 800  
#define HEIGHT 600
```

```
void init()  
{
```

```
    glutInitWindowPosition(100, 100);  
    glutInitWindowSize(WIDTH, HEIGHT);  
}
```

```
void reshape(int w, int h)  
{
```

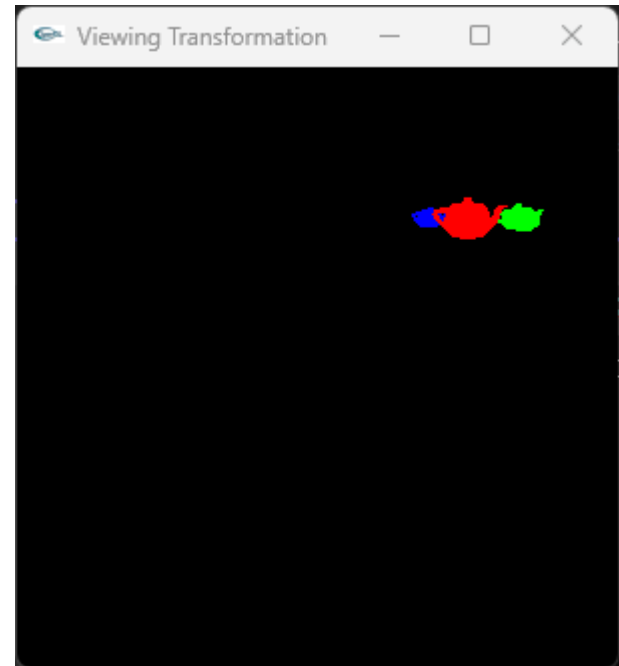
```
    (1) glViewport(0, 0, w, h);  
    (2) glViewport(0, 0, w/2, h/2);  
    (3) glViewport(w/2, h/2, w/2, h/2);  
    (4) glViewport(w/2, 0, w/2, h/2);  
}
```



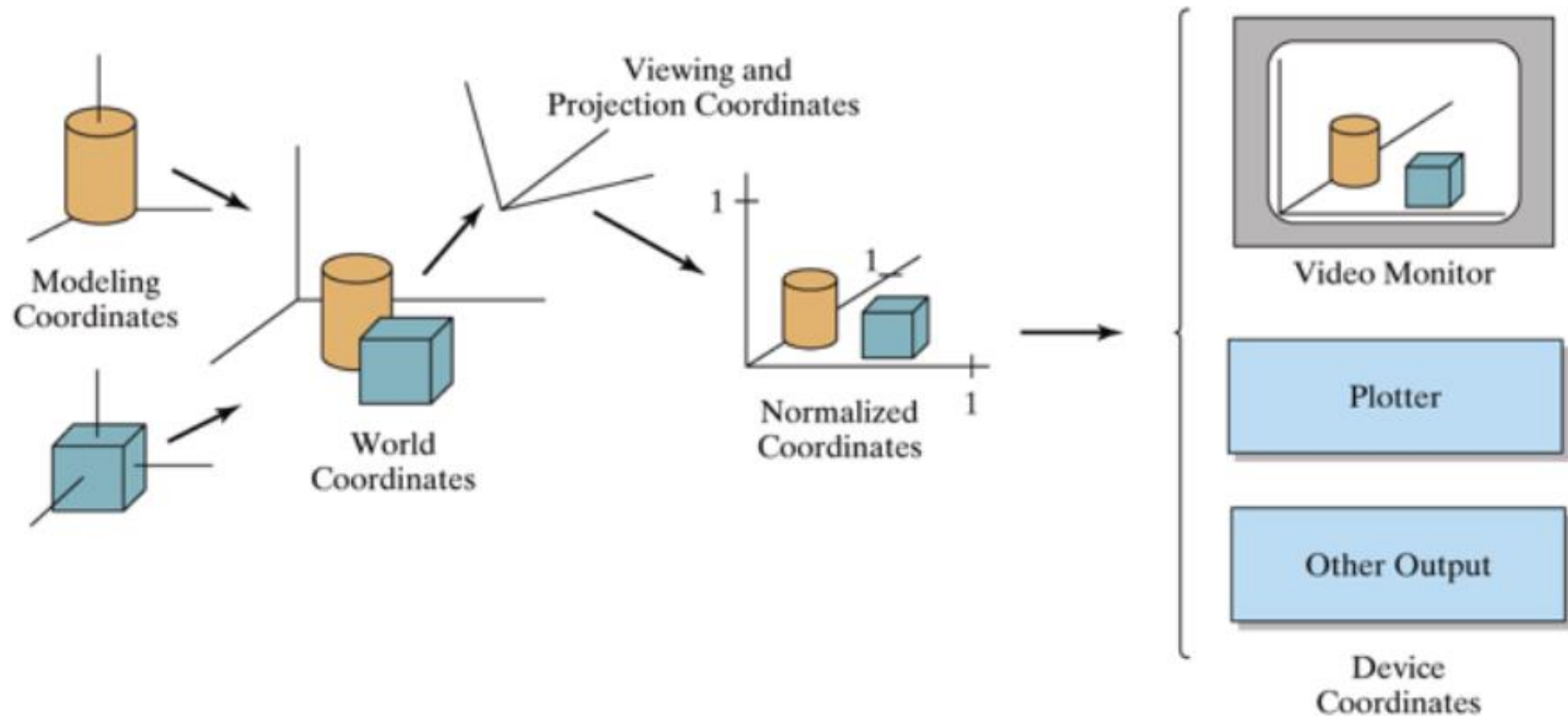
glViewport()

- 위치 : reshape()

```
void reshape(int w, int h) // window width, height 파라미터 전달
{
    glViewport(w/2, h/2, w/2, h/2);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-10.0, 10.0, -10.0, 10.0, 5.0, 30.0);
    glMatrixMode(GL_MODELVIEW);
}
```



summary: Viewing Pipeline



Viewing Transformation
Thank you !

glOrtho
glFrustum
gluPerspective