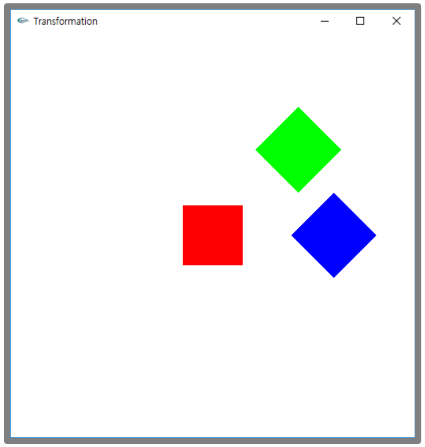


디지털 그래픽스 [5주차]

Geometric Coding

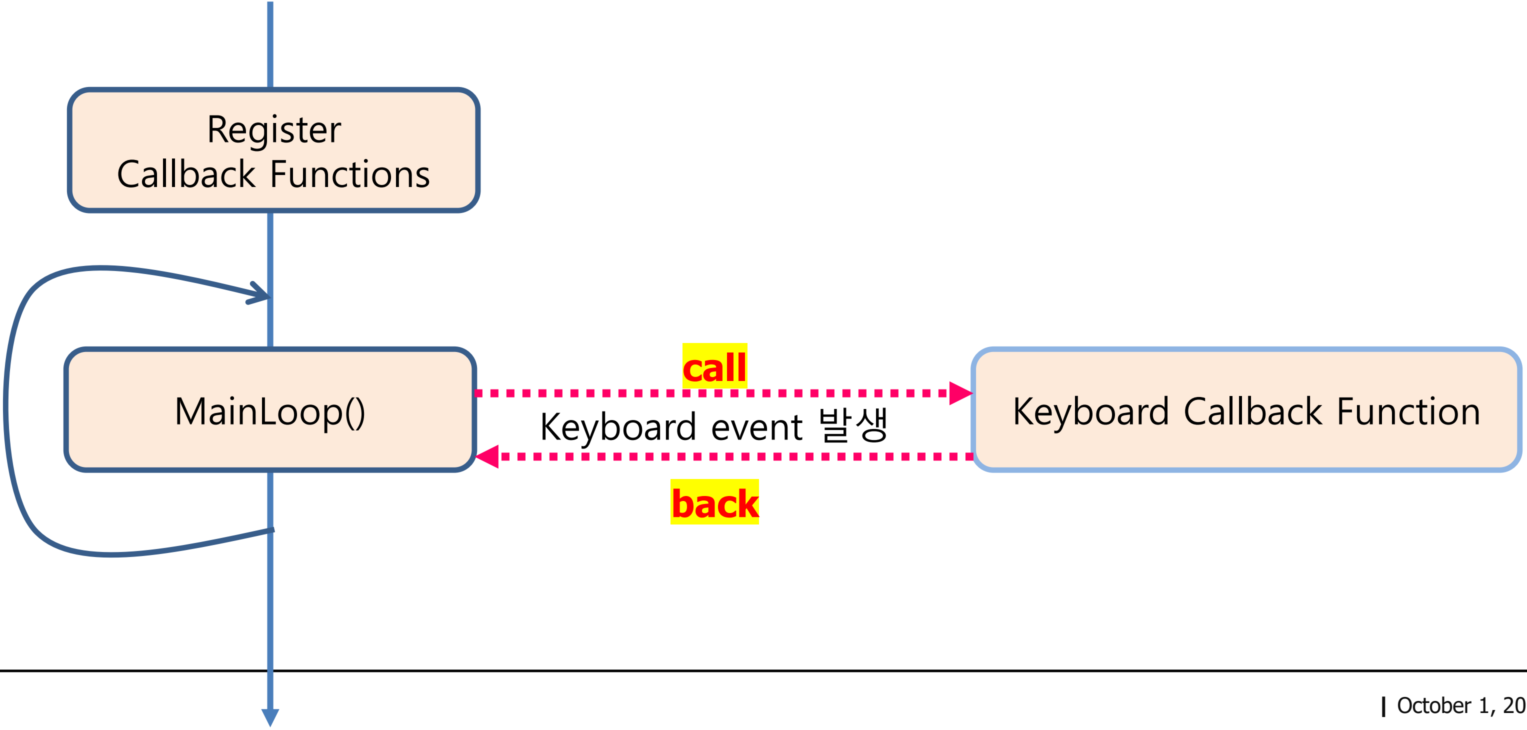
- GL Basic Programming
- GL Drawing Function
- GL Transformation Function

Goals

- GL Basic Programming
- GL Drawing Function
- Transformation Matrix
- Coding GL Transformation
- Order matters in Transformations

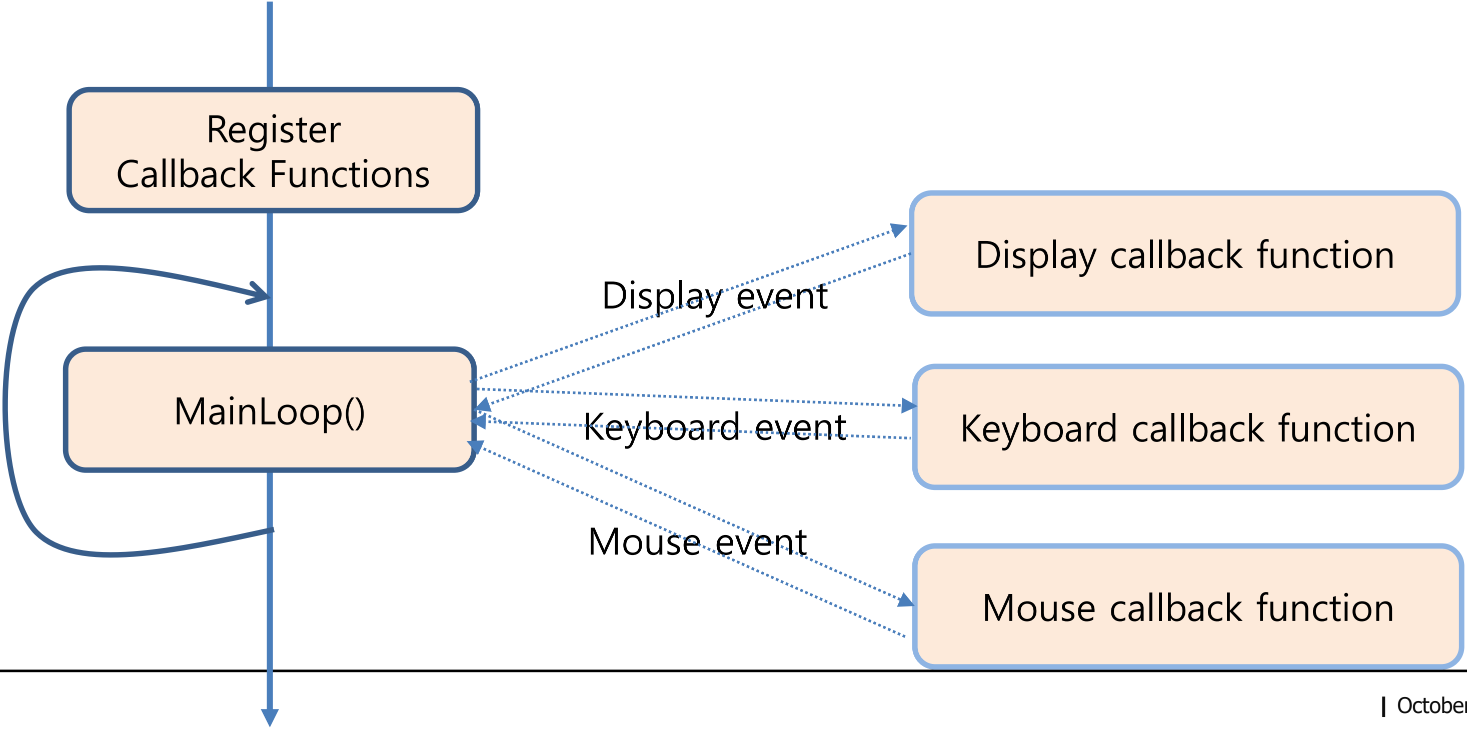
OpenGL 프로그래밍 작동 방식

- **프로그램이 특정 이벤트가 발생할 때, 미리 정의한 함수를 호출하는 방식**
- Callback function 등록
  - callback function (특정 이벤트가 발생했을 때 자동 호출되는 함수)
- 기본적으로 Mainloop() 실행
- event 발생했을 때 callback function 호출
- event : mouse, keyboard, window..



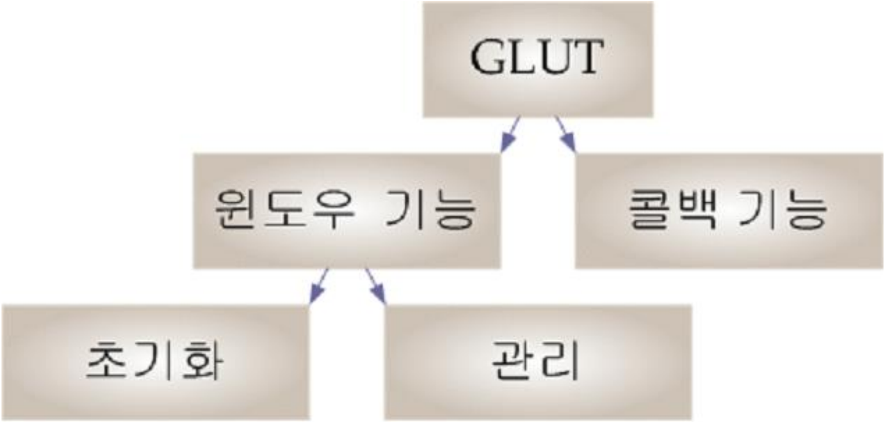
OpenGL 프로그래밍 작동 방식

- **프로그램이 특정 이벤트가 발생할 때, 미리 정의한 함수를 호출하는 방식**
- Callback function 등록
  - callback function (특정 이벤트가 발생했을 때 자동 호출되는 함수)
- 기본적으로 Mainloop() 실행
- event 발생했을 때 callback function 호출
- event : mouse, keyboard, window..



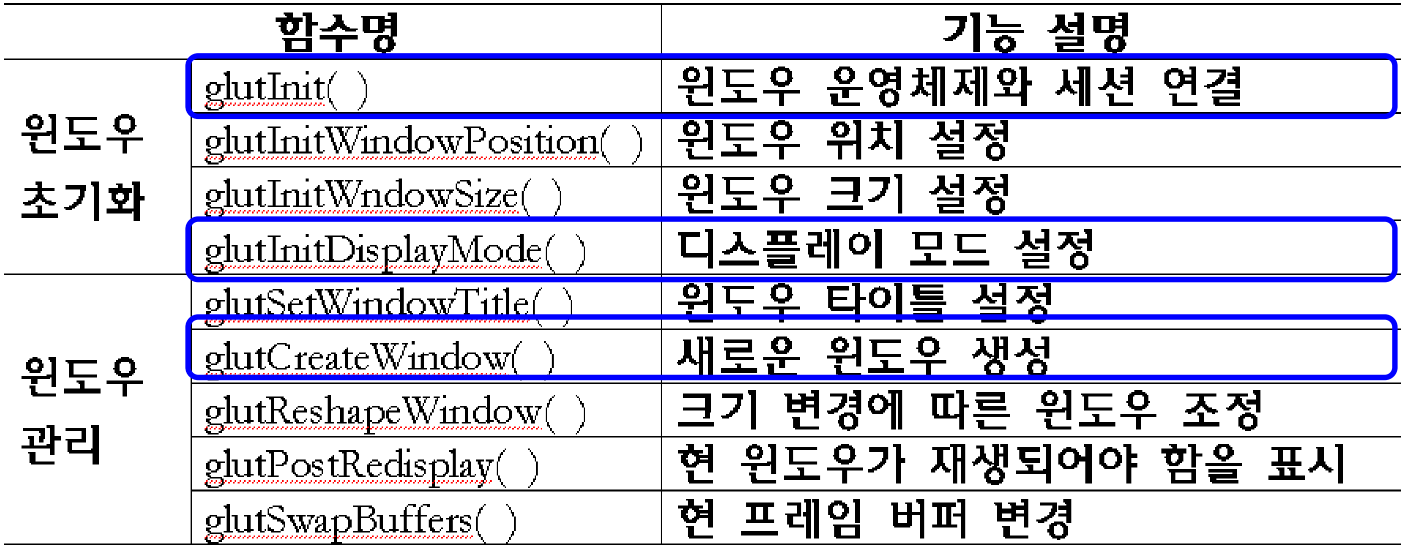
GLUT 라이브러리

- **윈도우 기능**: 프로그램 실행에 필요한 창(Window)을 관리
  - MS window, Unix X-window..
- **콜백(Callback) 기능**: 프로그램 실행 중 발생하는 디스플레이나 사용자 입력을 처리
  - display, mouse, keyboard



GLUT 라이브러리 - 윈도우 기능

- **윈도우 초기화**: 윈도우 운영체제와 OpenGL 세션 연결 & 초기화 모드 설정
- **윈도우 관리**: 윈도우 생성과 생성 이후 윈도우 관리



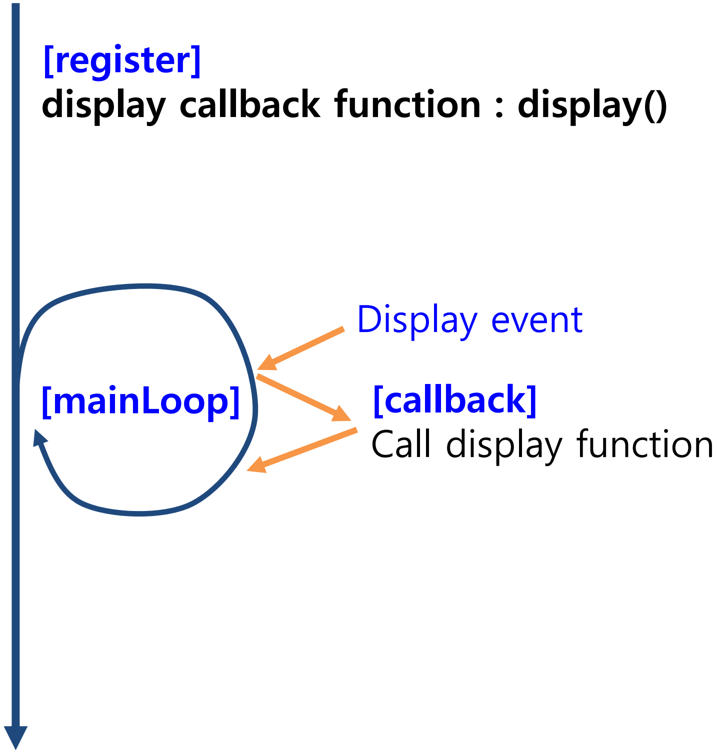
함수명		기능 설명
윈도우 초기화	glutInit( )	윈도우 운영체제와 세션 연결
	glutInitWindowPosition( )	윈도우 위치 설정
	glutInitWndowSize( )	윈도우 크기 설정
	glutInitDisplayMode( )	디스플레이 모드 설정
윈도우 관리	glutSetWindowTitle( )	윈도우 타이틀 설정
	glutCreateWindow( )	새로운 윈도우 생성
	glutReshapeWindow( )	크기 변경에 따른 윈도우 조정
	glutPostRedisplay( )	현 윈도우가 재생되어야 함을 표시
	glutSwapBuffers( )	현 프레임 버퍼 변경

GLUT 라이브러리 - 콜백 기능

- Register callback function
- Callback the function

```
void display()
{
……
}
```

Event-driven programming



[register]
display callback function : display()

Display event

[mainLoop]

[callback]
Call display function

[Callback Function() 등록]

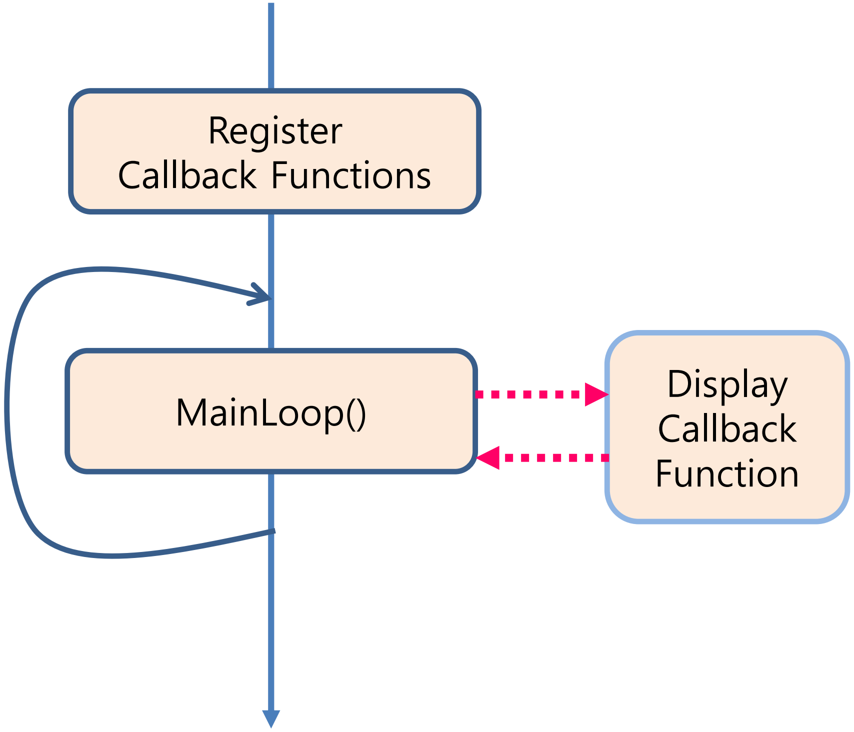
glutDisplayFunc(display)

[Main Loop]

glutMainLoop()

OpenGL program

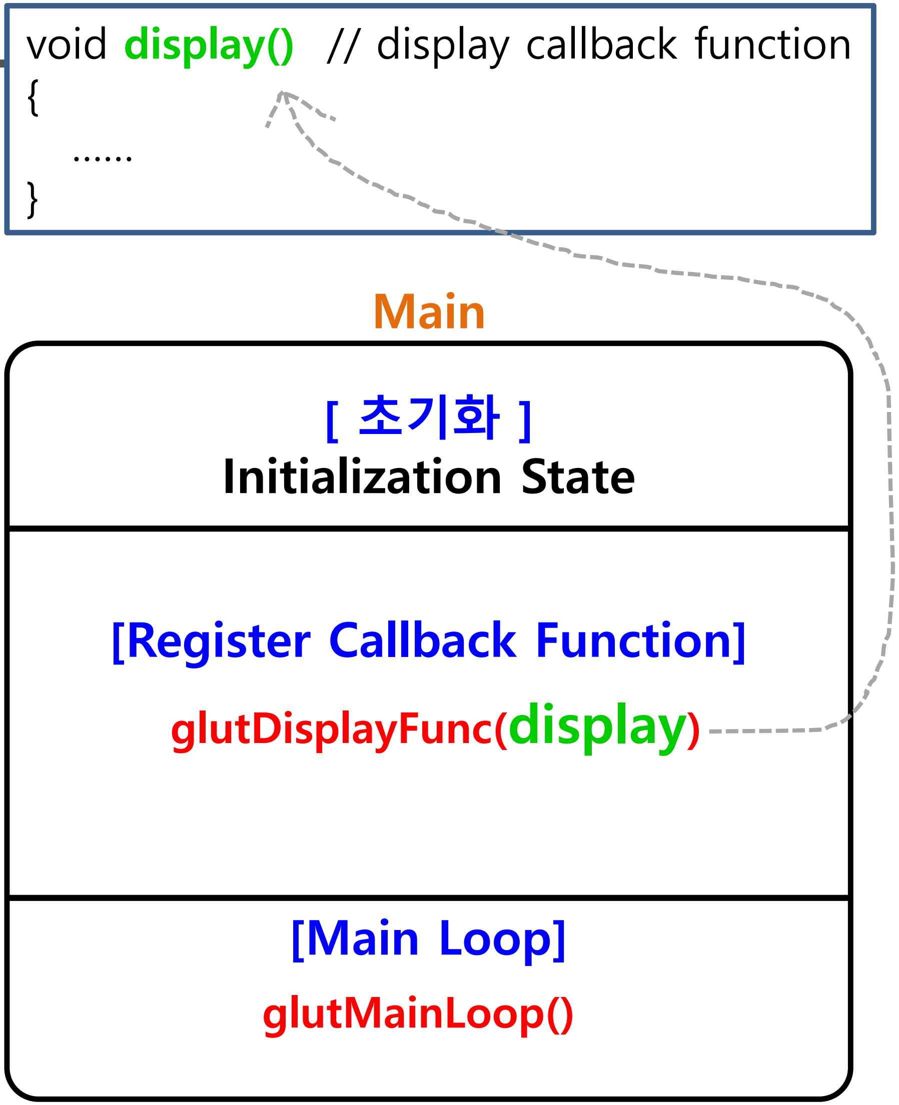
- OpenGL 프로그램 구성
  - 초기화
  - Callback Function 등록
  - Main Loop로 구성



Register Callback Functions

MainLoop()

Display Callback Function



```
void display()  // display callback function
{
    ......
}
```

Main

[초기화]
Initialization State

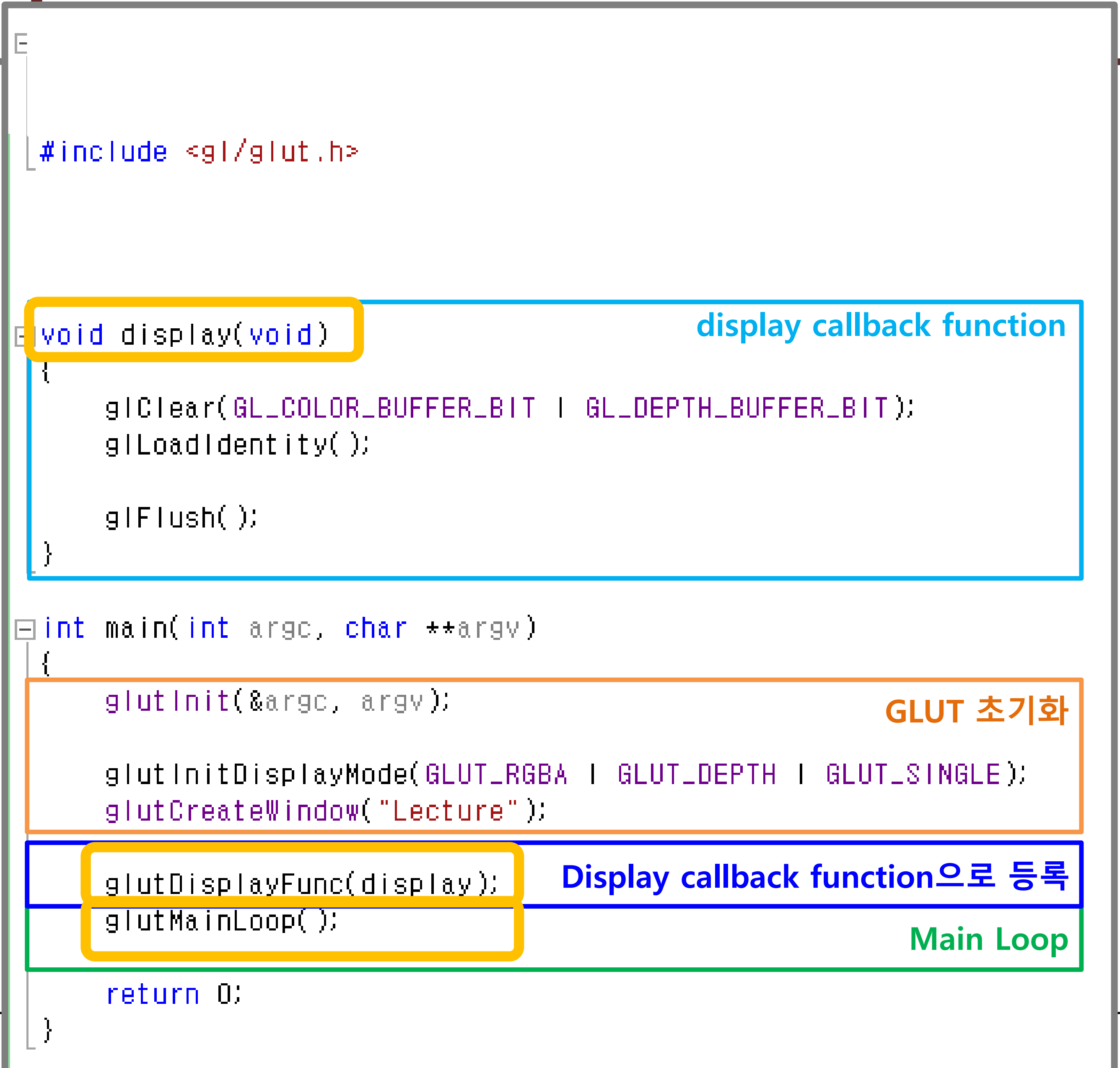
[Register Callback Function]

glutDisplayFunc(display)

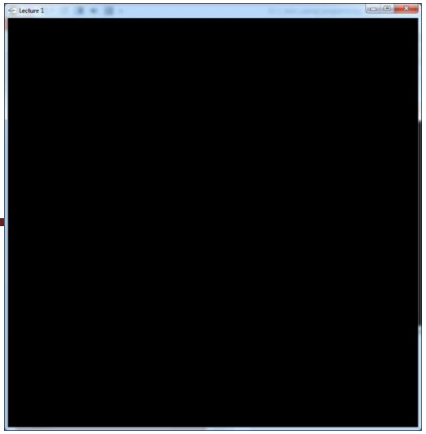
[Main Loop]

glutMainLoop()

OpenGL Basic Code

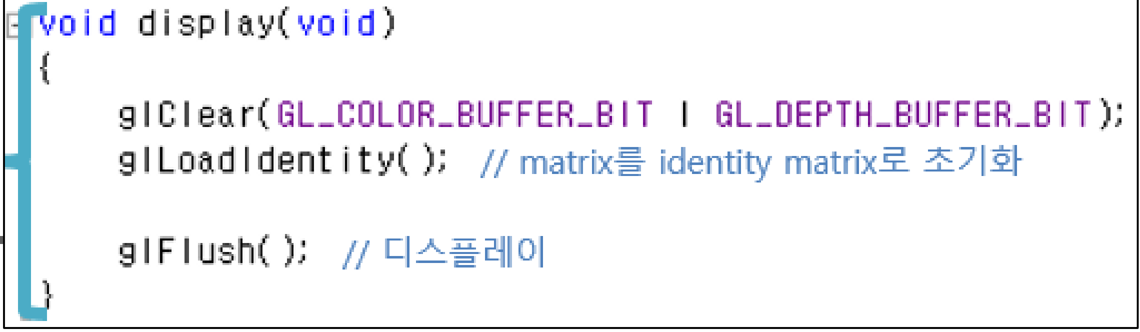


```c
#include <gl/glut.h>


void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity( );

	glFlush( );
}

int main( int argc, char **argv)
{
	glutInit(&argc, argv);

	glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
	glutCreateWindow("Lecture");

	glutDisplayFunc(display);
	glutMainLoop( );

	return 0;
}
```



<실행 화면>

glClear()



```
void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity(); // matrix를 identity matrix로 초기화

	glFlush(); // 디스플레이
}
```

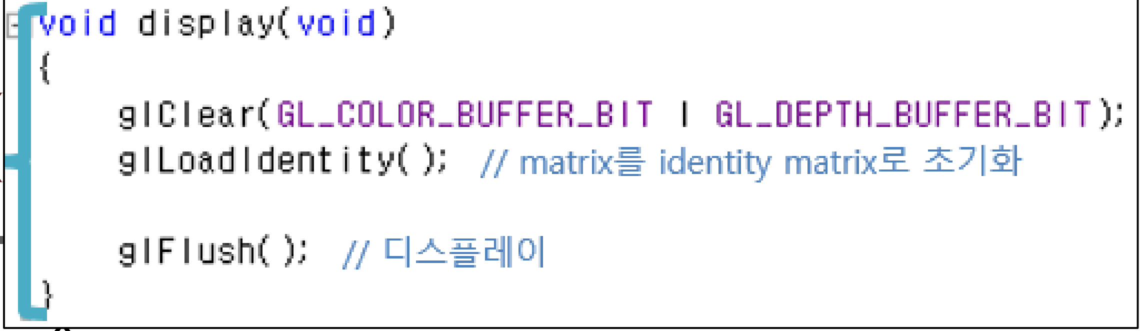
- glClear()
  - glClear -- clear frame buffer
  - frame buffer : the region of memory that holds the color data for the image displayed on a computer screen

- void glClear(GLbitfield mask) |
  - Bitwise OR of **masks** that indicate the buffers to be cleared.

- mask

GL_COLOR_BUFFER_BIT : 색상 값을 저장하는 버퍼
Indicates the buffers currently enabled for color writing.
GL_DEPTH_BUFFER_BIT : 깊이 값을 저장하는 버퍼 (3차원)
Indicates the depth buffer.

glLoadIdentity(

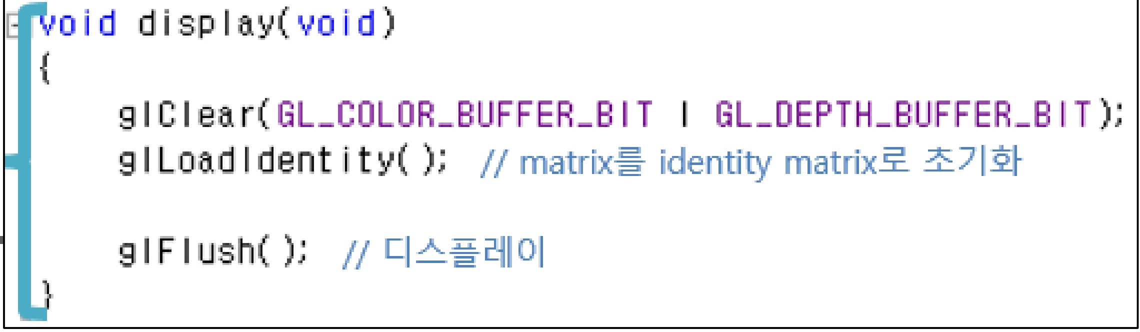


```
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity( );   // matrix를 identity matrix로 초기화

        glFlush( );   // 디스플레이
}
```

- void **glLoadIdentity**()
  - replace the current matrix with the identity matrix

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

glFlush()



```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity( );  // matrix를 identity matrix로 초기화

    glFlush( );  // 디스플레이
}
```

- **void glFlush()**
  - Forces previously issued OpenGL commands to begin ***execution***

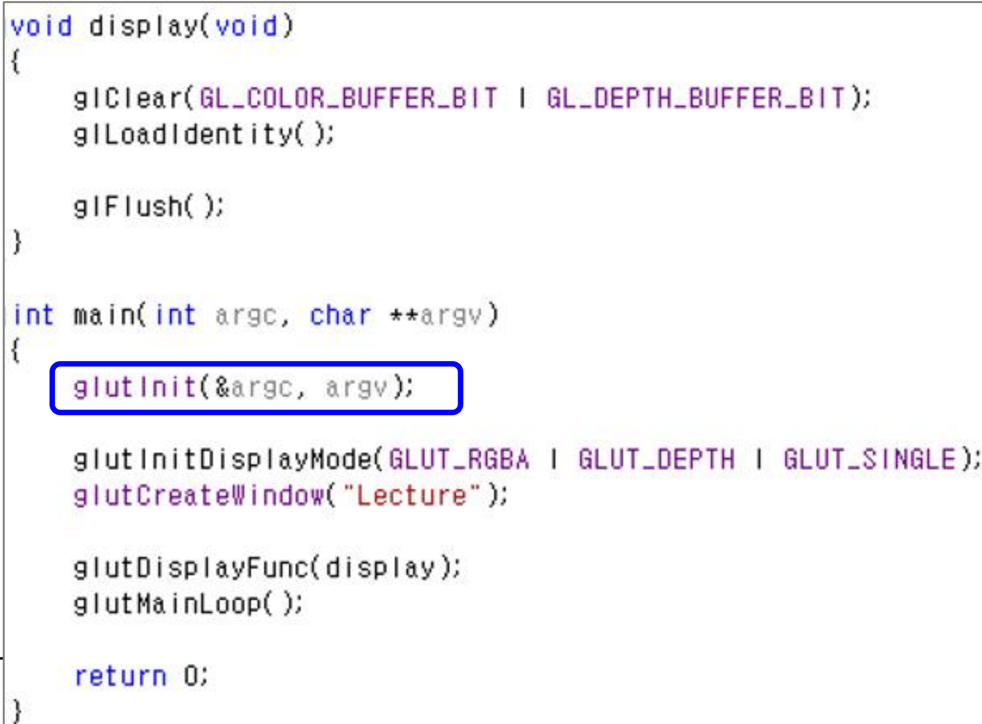
* 주의사항: glFlush()를 해야 디스플레이를 시작한다.

glutInit()

- **void glutInit(int *argc, char **argv)**
  - 프로그램을 실행할 때 **명령어 옵션**이나 **설정**을 전달받아 GLUT(OpenGL Utility Toolkit) 라이브러리를 초기화
  - argcp: A pointer to the program's unmodified argc variable from main. (명령어 옵션의 개수를 가리키는 포인터)
  - argv: The program's unmodified argv variable from main.

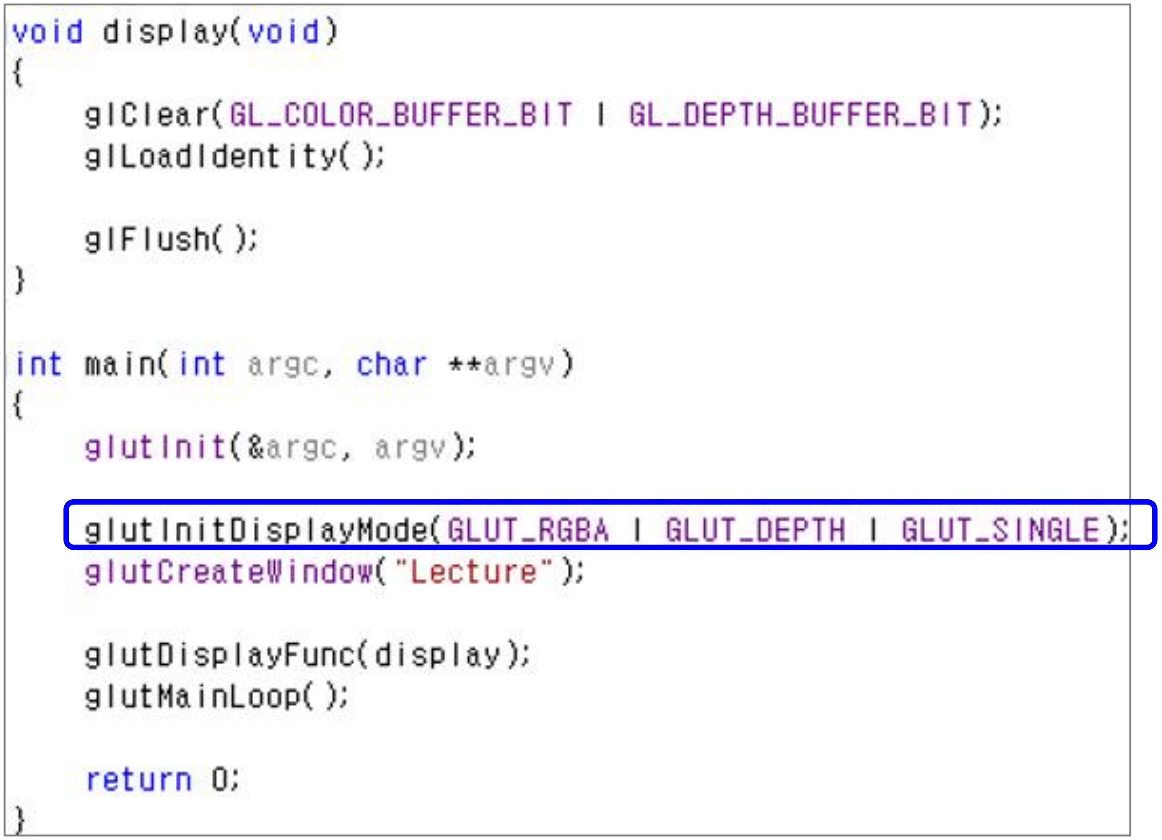
명령어 옵션의 **목록**을 담고 있는 배열. 이 배열은 문자열들로 이루어져 있다.

- **int main(int *argc, char **argv)**
  - 명령어창에서 프로그램 실행할 때
  - > test 3 5



```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity( );

    glFlush( );
}

int main( int argc, char **argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow( "Lecture" );

    glutDisplayFunc(display);
    glutMainLoop( );

    return 0;
}
```

glutInitDisplayMode()



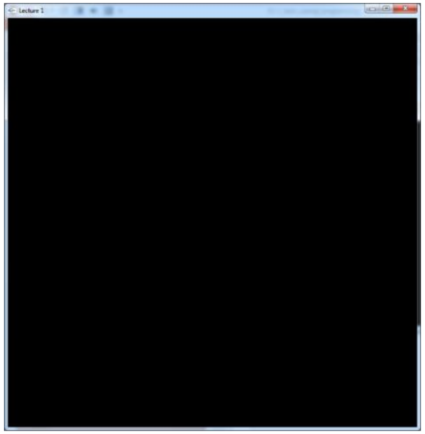
```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity( );

    glFlush( );
}

int main( int argc, char **argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow("Lecture");

    glutDisplayFunc(display);
    glutMainLoop( );

    return 0;
}
```

glutInitDisplayMode()

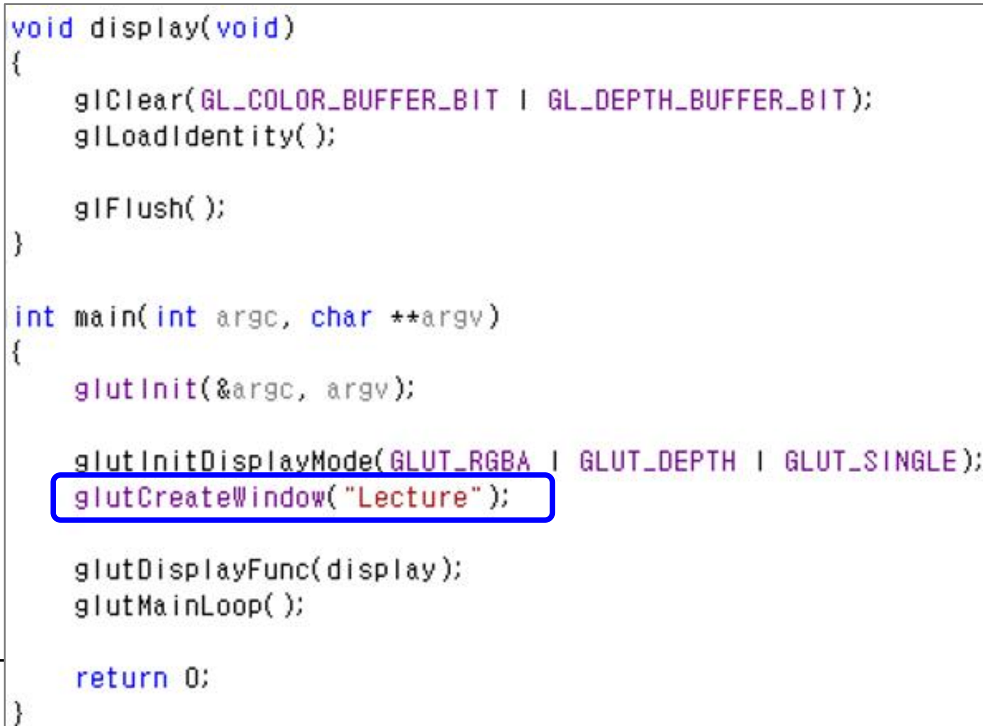
- **glutInitDisplayMode()**
  - sets the *initial display mode*.
- **void glutInitDisplayMode(unsigned int mode);**
  - the bitwise *OR*-ing of GLUT display mode bit masks
- **Mode**
  - **GLUT_RGBA**
    Bit mask to select an RGBA mode window. This is the default if neither GLUT_RGBA or GLUT_INDEX are specified..
  - **GLUT_DEPTH**
    Bit mask to select a window with a depth buffer.
  - **GLUT_SINGLE**
    Bit mask to select a single buffered window. This is the default if neither GLUT_DOUBLE or GLUT_SINGLE are specified.
  - **GLUT_RGB**
    An alias for GLUT_RGBA.
  - **GLUT_DOUBLE**
    Bit mask to select a double buffered window. This overrides GLUT_SINGLE if it is also specified.

glutCreateWindow()

- **int glutCreateWindow(char *name)**
    - Creates a top-level window.
    - name: ASCII character string for use as window name



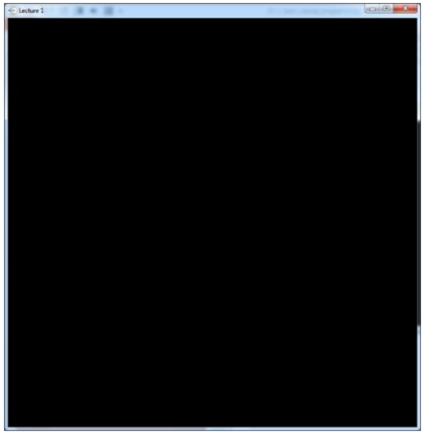
<실행 화면>



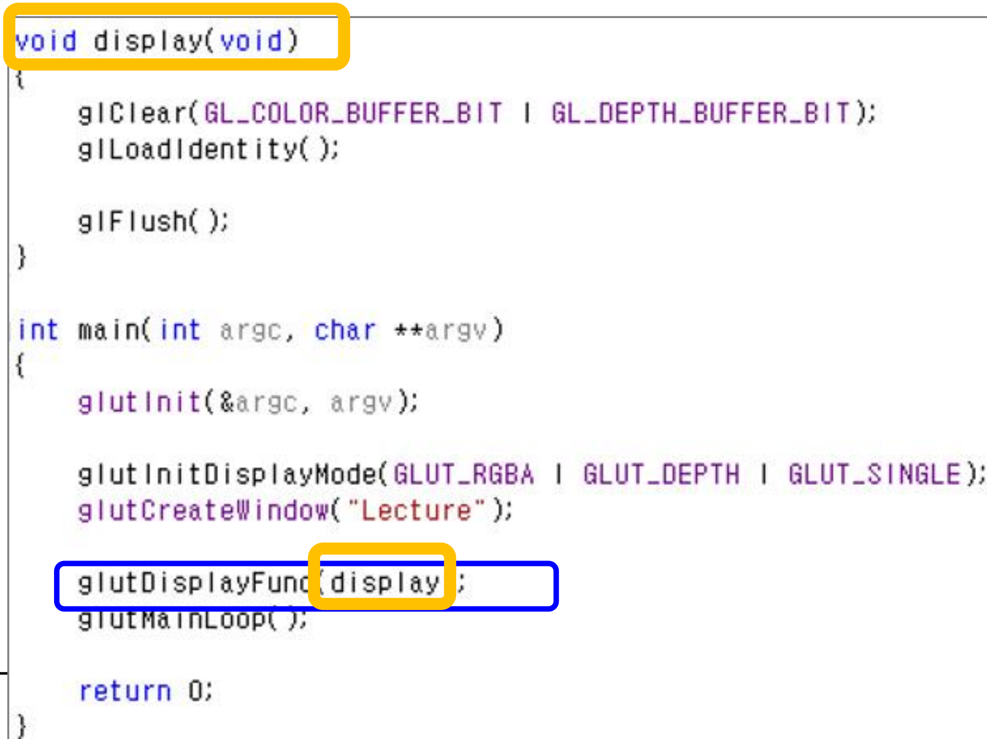
```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow("Lecture");

    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

glutDisplayFunc()

- **int glutDisplayFunc(void (*func)(void))**
  - Sets the display callback function for the current window
  - func: The new display callback function, 함수명



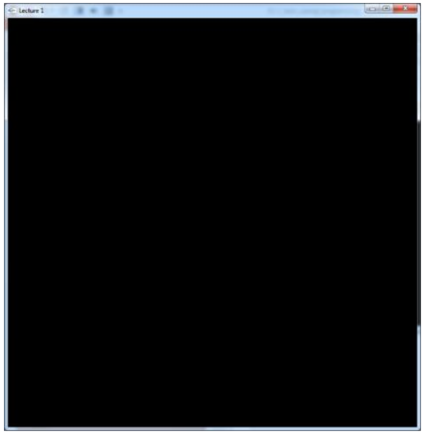
<실행 화면>



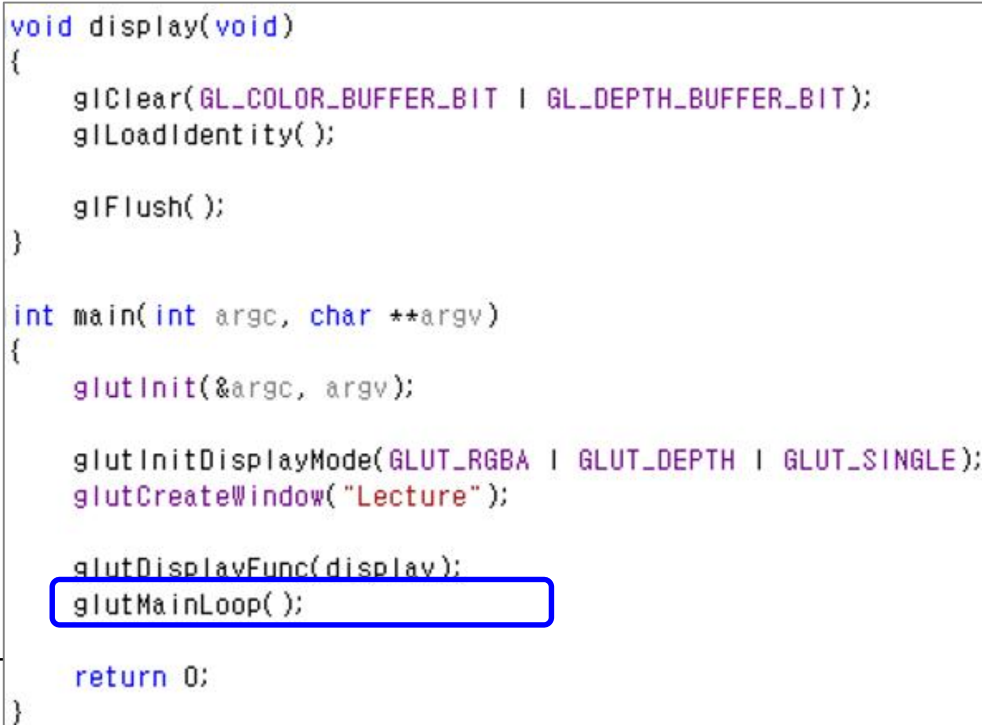
```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow("Lecture");

    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

glutMainLoop()

- **int glutMainLoop()**
  - Enters the GLUT event processing loop
  - This routine should be called at most once in a GLUT program. Once called, this routine will never return.
  - It will call as necessary any callbacks that have been registered.

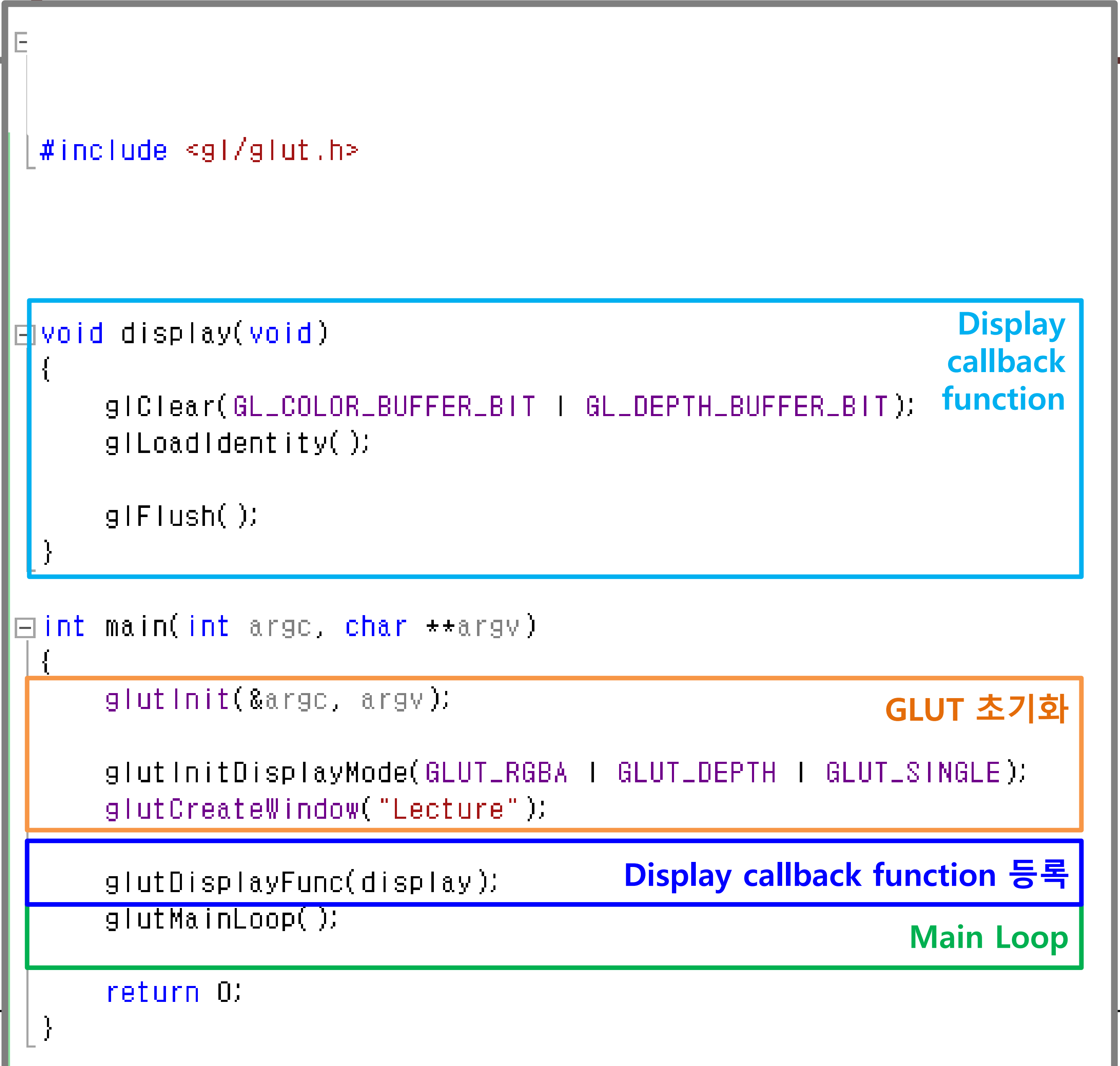


<실행 화면>



```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity( );

    glFlush( );
}

int main( int argc, char **argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow( "Lecture" );

    glutDisplayFunc(display);
    glutMainLoop( );

    return 0;
}
```

OpenGL Basic Code



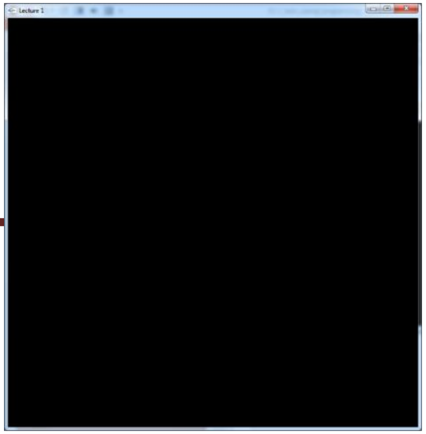
```
#include <gl/glut.h>

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow("Lecture");

    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

Display callback function

GLUT 초기화

Display callback function 등록

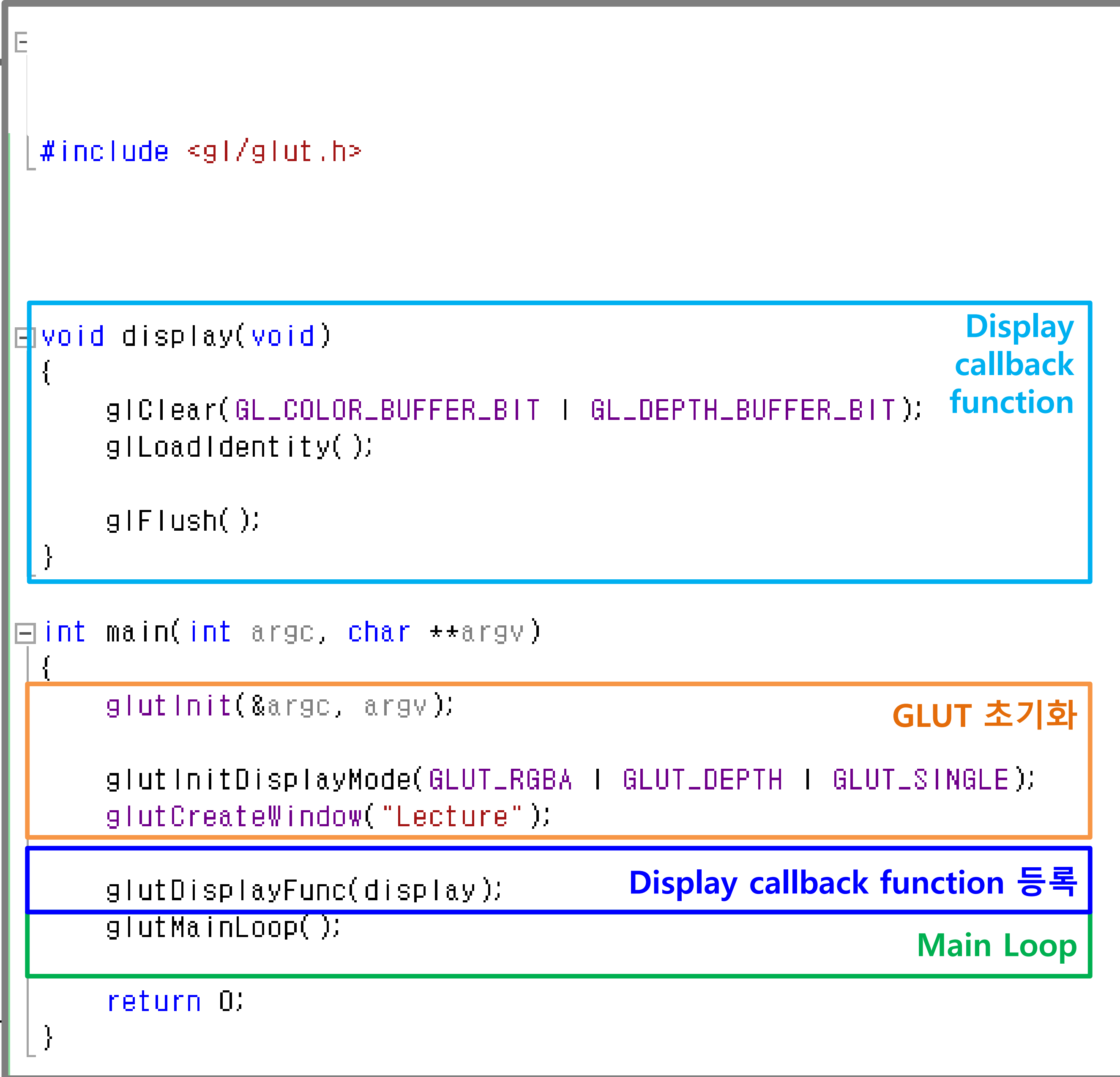
Main Loop



<실행 화면>

Mission : OpenGL 코드 이해

- 파트너에게 OpenGL 코드 설명하기



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

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity( );

    glFlush( );
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow("Lecture");

    glutDisplayFunc(display);
    glutMainLoop( );

    return 0;
}
```

Display callback function

GLUT 초기화

Display callback function 등록

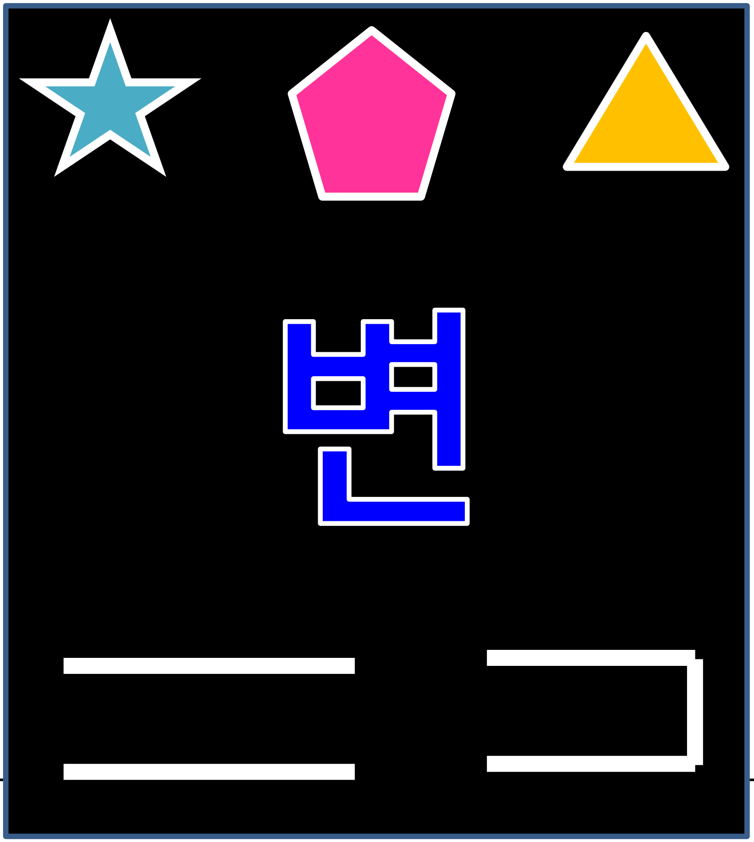
Main Loop

Goals

- OpenGL Basic Programming
- OpenGL Drawing Function
- Transformation Matrix
- Coding OpenGL Transformation
- Order matters in Transformations

OpenGL

- 학습목표
  - OpenGL을 이용하여 점, 선, 면을 그리는 프로그램을 구현한다.

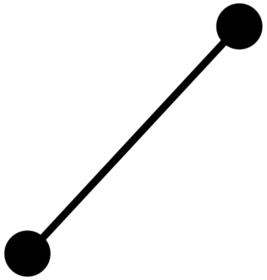


OpenGL Drawing Function

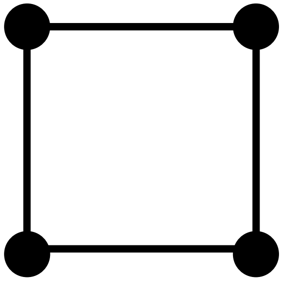
OpenGL Basic Primitive

- 그림을 그리기 위한 기본 요소

점



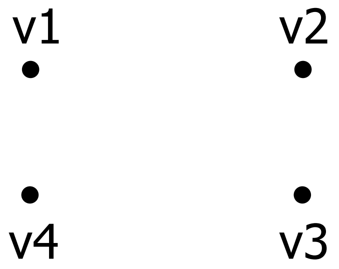
선



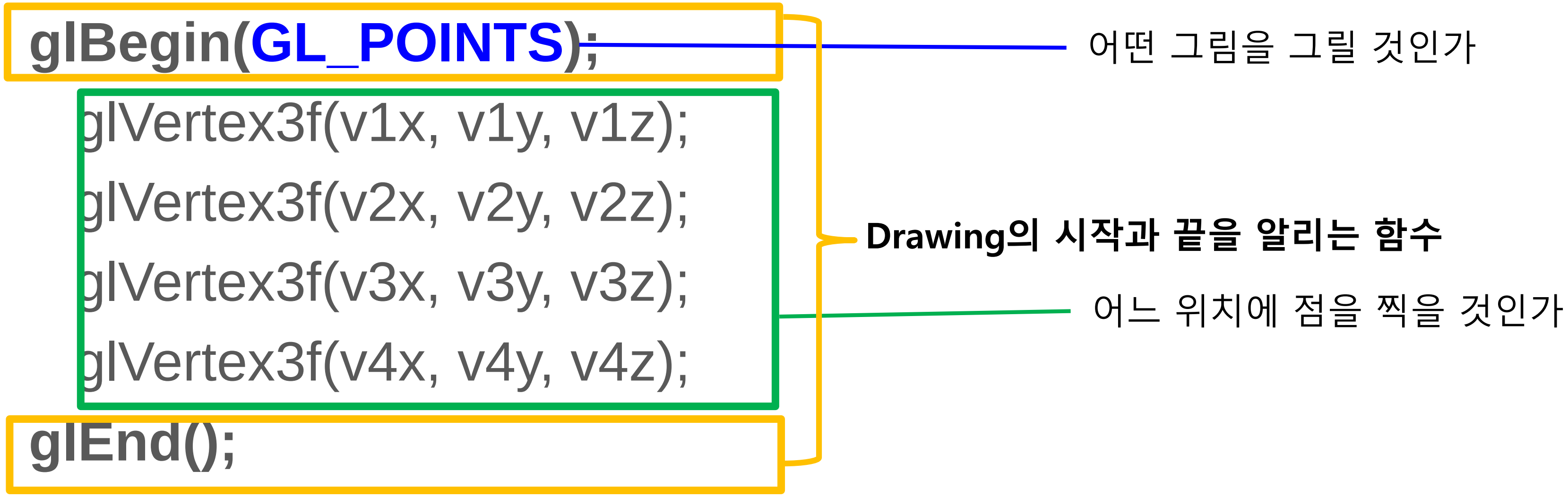
면

Point

- 4개의 점을 찍는 OpenGL 코드



- GL_POINTS



```
glBegin(GL_POINTS);
    glVertex3f(v1x, v1y, v1z);
    glVertex3f(v2x, v2y, v2z);
    glVertex3f(v3x, v3y, v3z);
    glVertex3f(v4x, v4y, v4z);
glEnd();
```

어떤 그림을 그릴 것인가

Drawing의 시작과 끝을 알리는 함수

어느 위치에 점을 찍을 것인가

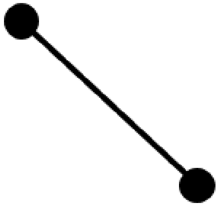
OpenGL Primitives

- 점, 선, 면
- Point
  - GL_POINTS

```
glBegin(GL_POINTS);
    glVertex3f(v1x, v1y, v1z);
    glVertex3f(v2x, v2y, v2z);
    glVertex3f(v3x, v3y, v3z);
    glVertex3f(v4x, v4y, v4z);
glEnd();
```

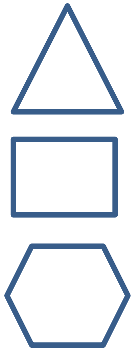
- Line
  - GL_LINES | GL_LINE_STRIP | GL_LINE_LOOP

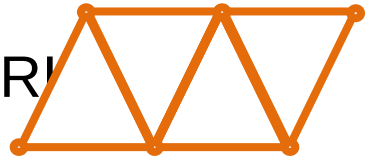
Line을 여러 개 이어서 그리는 방법

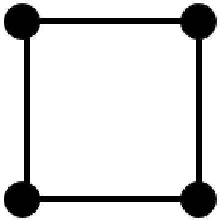


- Polygon
  - GL_TRIANGLES | GL_TRIANGLE_STRIP | GL_TRIANGLE_FAN
  - GL_QUADS | GL_QUAD_STRIP
  - GL_POLYGON

Triangle을 여러 개 이어서 그리는 방법





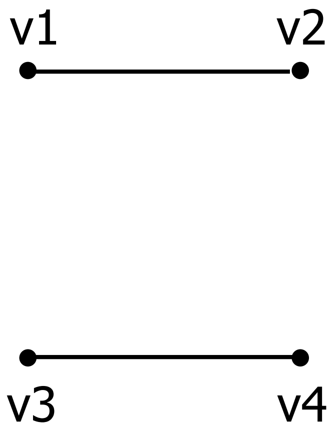


Line

- 선을 그리는 OpenGL 코드

- **GL_LINES**

```
glBegin(GL_LINES);
   glVertex3f(v1x, v1y, v1z);
   glVertex3f(v2x, v2y, v2z);
   glVertex3f(v3x, v3y, v3z);
   glVertex3f(v4x, v4y, v4z);
glEnd();
```

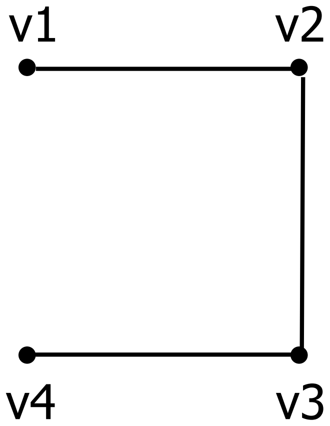


3D Connected Lines

- 여러 개의 선을 **이어서** 그리는 2가지 방법
- **GL_LINE_STRIP**

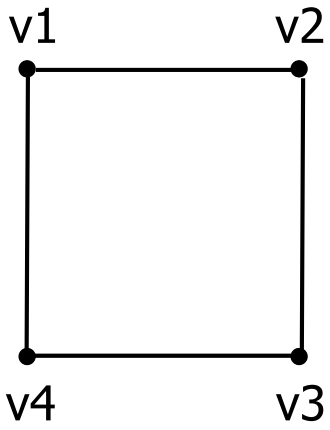
```
glBegin(GL_LINE_STRIP);
    glVertex3f(v1x, v1y, v1z);
    glVertex3f(v2x, v2y, v2z);
    glVertex3f(v3x, v3y, v3z);
    glVertex3f(v4x, v4y, v4z);
glEnd();
```

** vertices 순서 중요함.



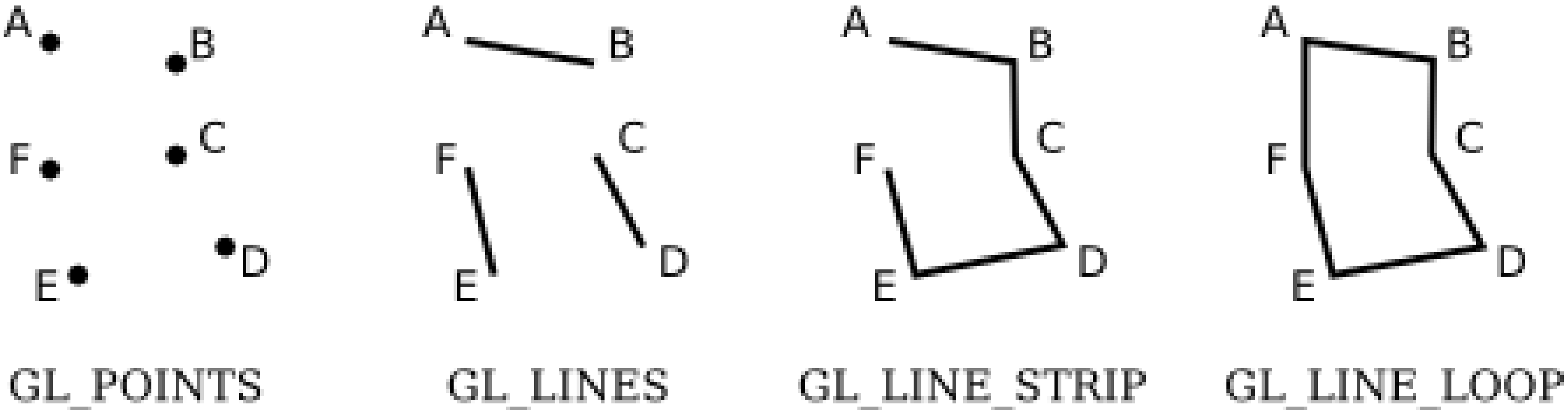
- **GL_LINE_LOOP**

```
glBegin(GL_LINE_LOOP);
    glVertex3f(v1x, v1y, v1z);
    glVertex3f(v2x, v2y, v2z);
    glVertex3f(v3x, v3y, v3z);
    glVertex3f(v4x, v4y, v4z);
glEnd();
```



GL 점, 선 그리기 summary

- 점 찍기
- 선 그리기 세가지



GL_POINTS GL_LINES GL_LINE_STRIP GL_LINE_LOOP

3D Triangle

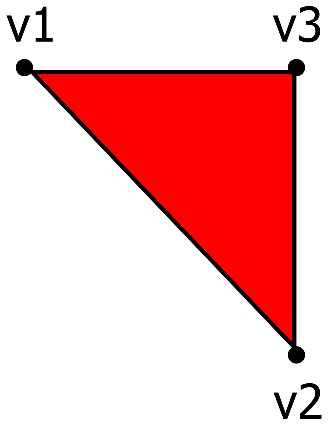
- 삼각형을 그리는 OpenGL 코드

- GL_TRIANGLES

```
glBegin(GL_TRIANGLES);
   glVertex3f(v1x, v1y, v1z);
   glVertex3f(v2x, v2y, v2z);
   glVertex3f(v3x, v3y, v3z);
glEnd();
```

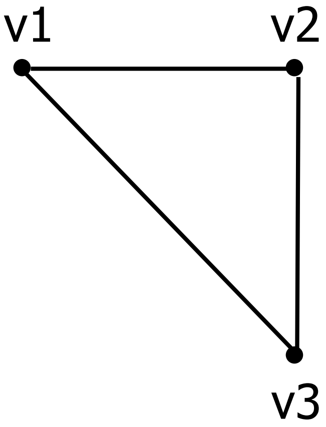
GL_TRIANGLES

** 내부가 채워져 있음.



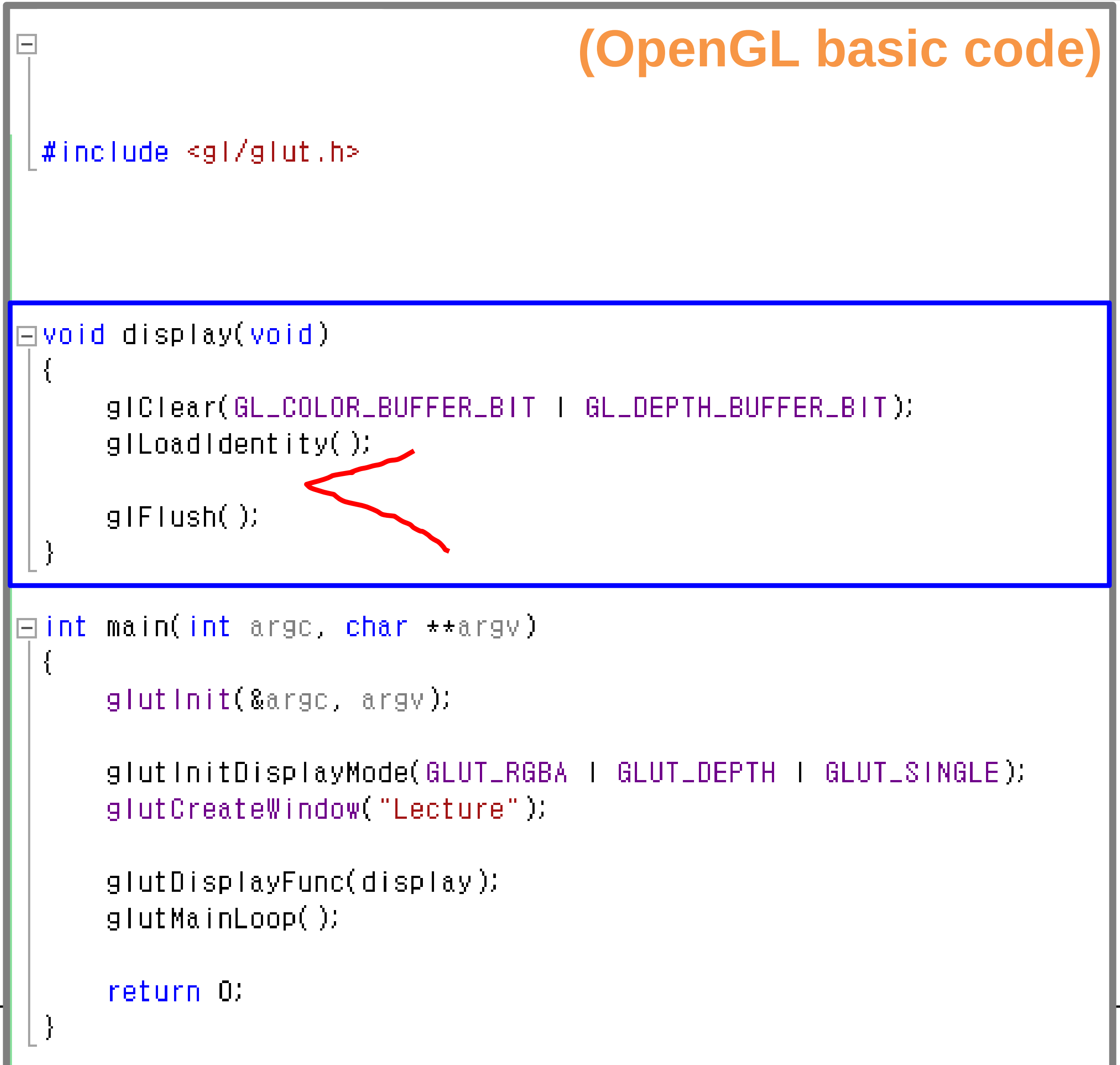
GL_LINE_LOOP

** 라인으로 그린 다각형은 단지 선들의 연결,
내부 개념 없음.



OpenGL Drawing Function

Drawing Triangle



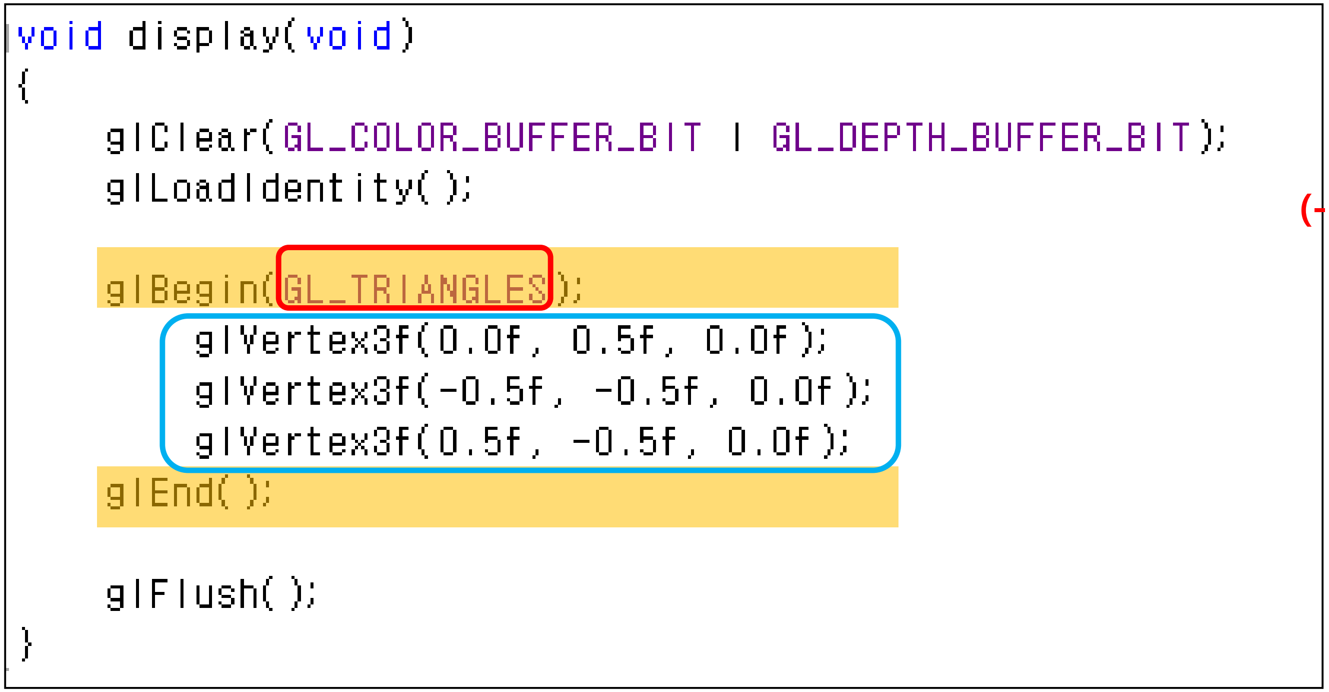
(OpenGL basic code)

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

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
    glutCreateWindow("Lecture");

    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

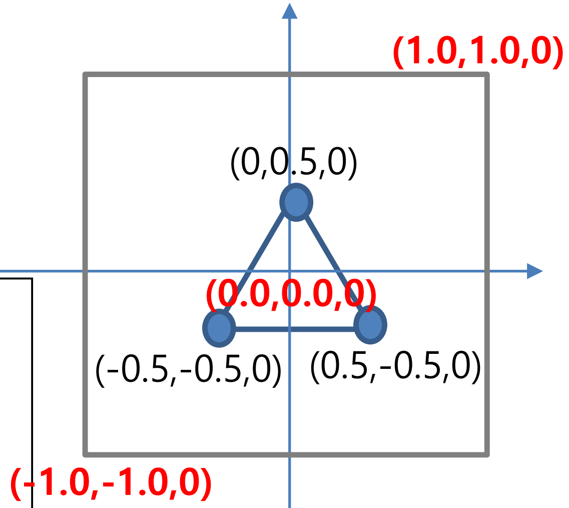
OpenGL Drawing Function

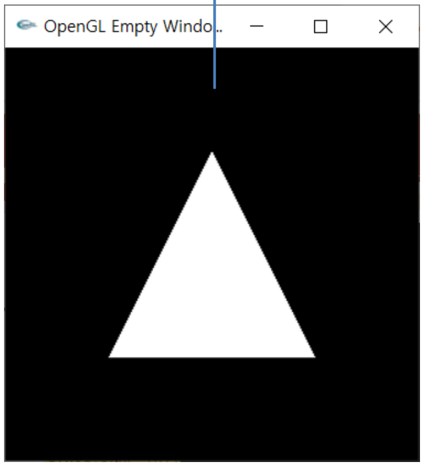
Triangle Code

- 윈도우 왼쪽 하단 (-1,-1)
- 윈도우 오른쪽 상단 (1,1)



```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glBegin(GL_TRIANGLES);
        glVertex3f(0.0f, 0.5f, 0.0f);
        glVertex3f(-0.5f, -0.5f, 0.0f);
        glVertex3f(0.5f, -0.5f, 0.0f);
    glEnd();

    glFlush();
}
```



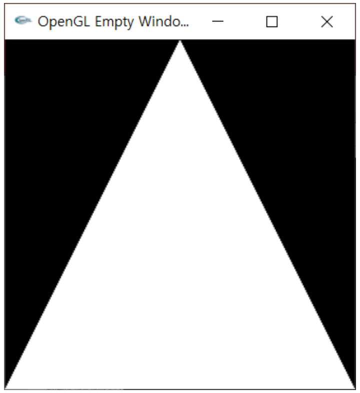


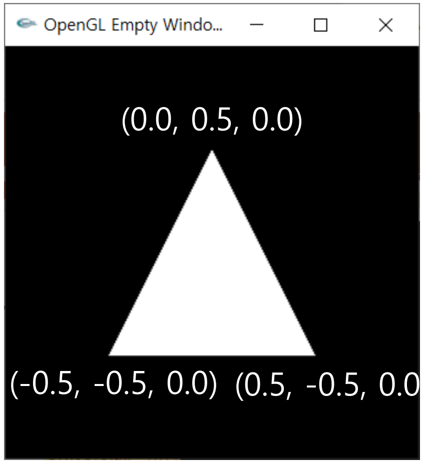
Mission

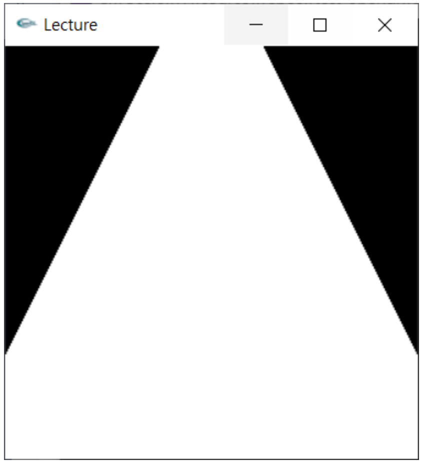
OpenGL Drawing Function

Triangle Code

- 다양한 크기의 삼각형 그리기

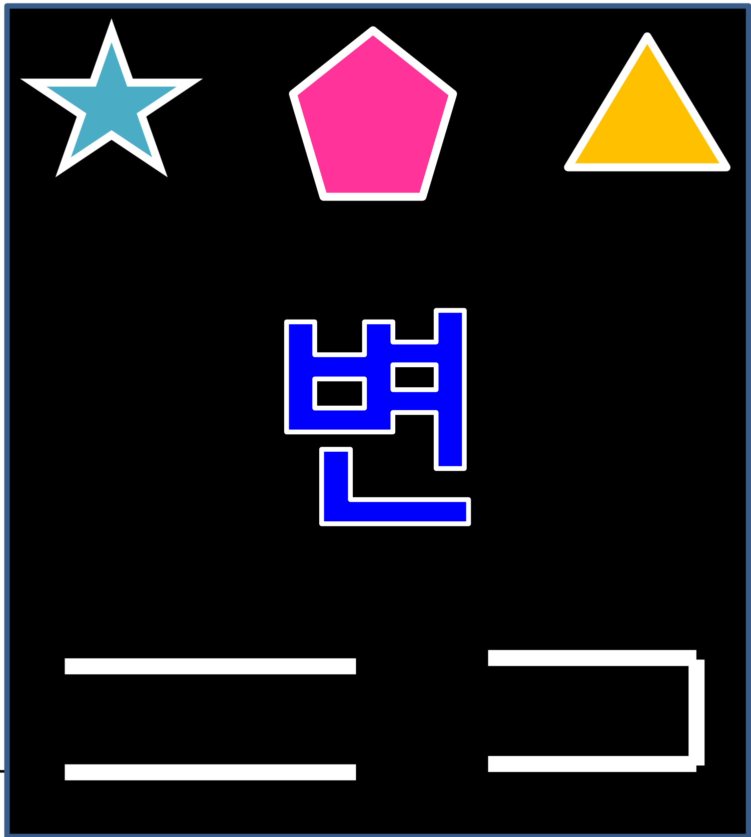






과제 1. Drawing

- **OpenGL 윈도우에 다음 그림을 그리세요. 가운데는 학생 이름의 성**
  - GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES 익히기
  - glVertex3f(….) 파라메터에 들어가는 좌표 익히기



Goals

- OpenGL Basic Programming
- OpenGL Drawing Function
- Transformation Matrix
- Coding OpenGL Transformation
- Order matters in Transformations

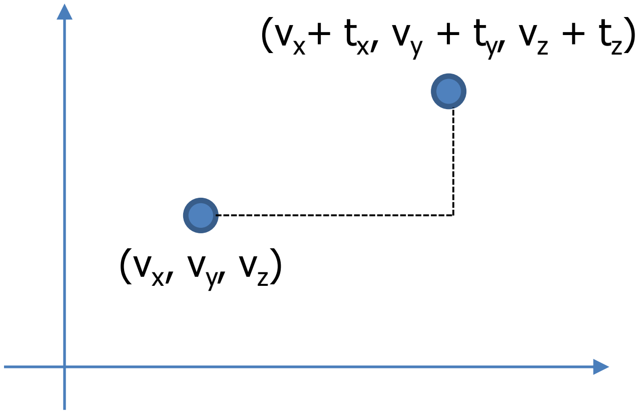
Matrix Multiplication

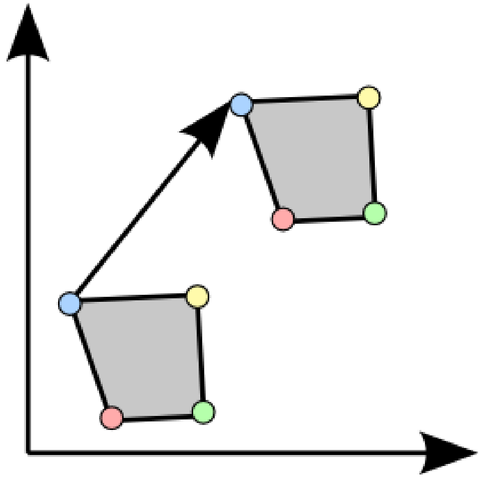
- **Translation matrix**

2D

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate





3D

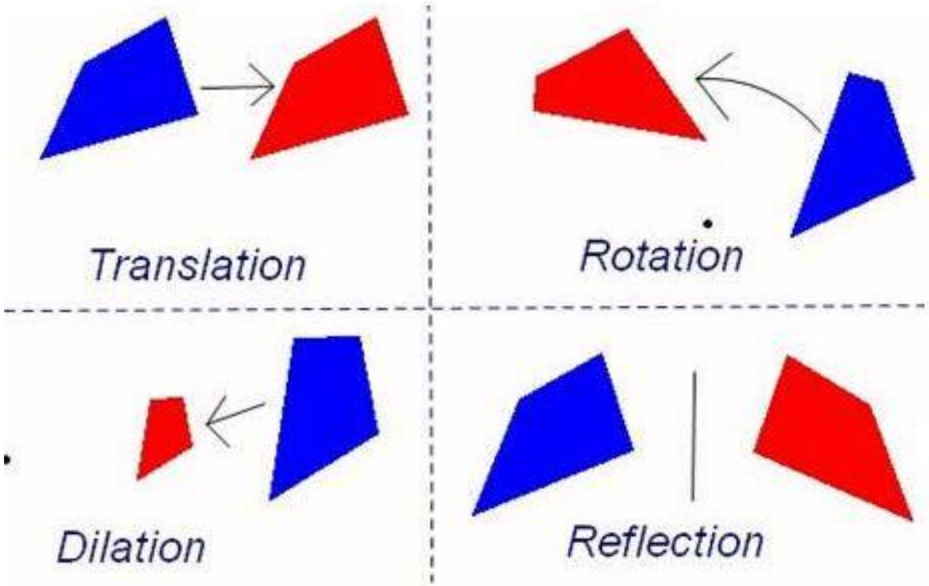
$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix} = \begin{pmatrix} v_x + t_x \\ v_y + t_y \\ v_z + t_z \\ 1 \end{pmatrix}$$

Geometric Transformation

- 기하 변환

Transformation

- glRotatef
- glTranslatef
- glScalef



Transformations can be represented by matrices.

Transformation

- Translation matrix

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix} = \begin{pmatrix} v_x + t_x \\ v_y + t_y \\ v_z + t_z \\ 1 \end{pmatrix}$$

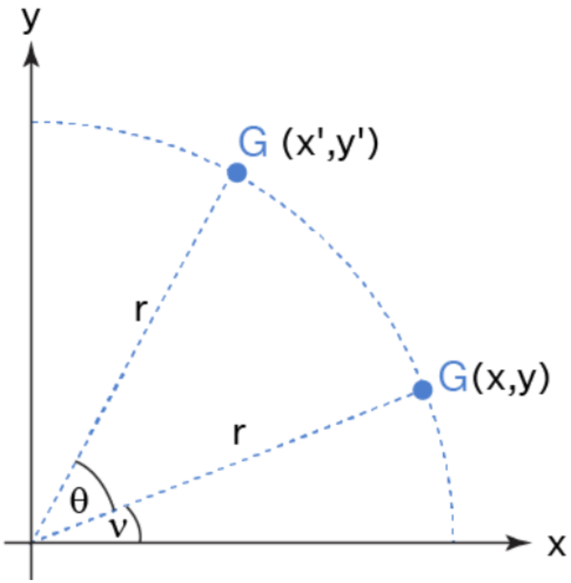
- Scaling matrix

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix} = \begin{pmatrix} s_x v_x \\ s_y v_y \\ s_z v_z \\ 1 \end{pmatrix}$$

Transformation

- **Rotation Matrix**

$$\mathbf{R}_z(\theta)=\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Transformation : Rotation

$$\mathbf{R}_Z(\theta)=\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Derivation of Rotation Matrix

$x = r \cos v$ -- (1)

$y = r \sin v$ -- (2)

Similarly, expressing (x', y') in polar form

$x' = r \boxed{\cos(v + \theta)}$

$y' = r \boxed{\sin(v + \theta)}$

Expanding the brackets using trigonometric identities we get,

$x' = r \boxed{(\cos v.\cos\theta - \sin v.\sin\theta)}$

$= \underbrace{r \cos v}_{\mathbf{x}}.\cos\theta - \underbrace{r \sin v}_{\mathbf{y}}.\sin\theta$

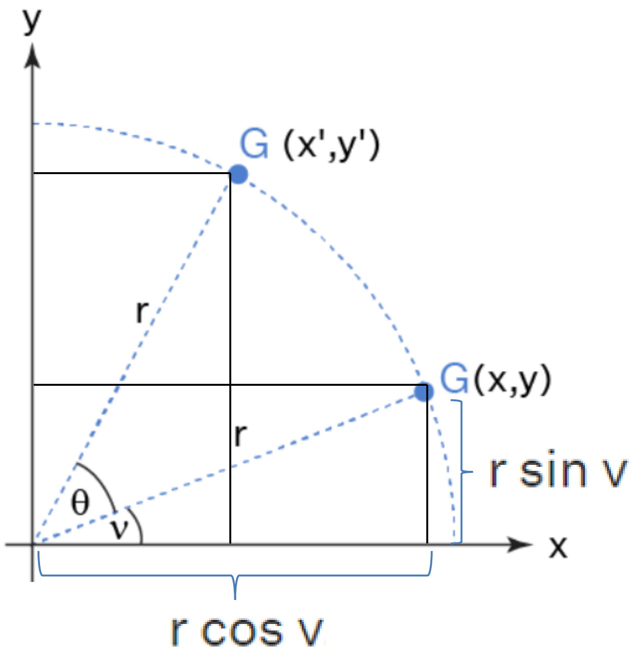
From (1) and (2) we have,

$\boxed{x' = x \cos\theta - y \sin\theta}$ -- (3)

$y' = r \boxed{(\sin v.\cos\theta + \cos v.\sin\theta)}$

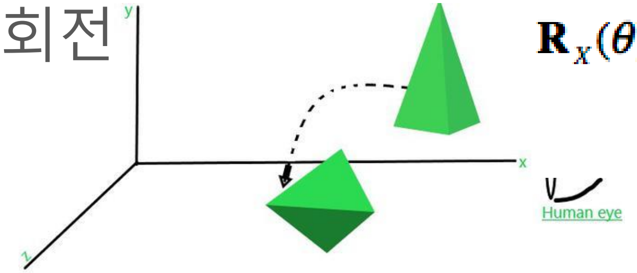
$= \underbrace{r \sin v}_{\mathbf{y}}.\cos\theta + \underbrace{r \cos v}_{\mathbf{x}}.\sin\theta$

$\boxed{y' = y \cos\theta + x \sin\theta}$ -- (4)



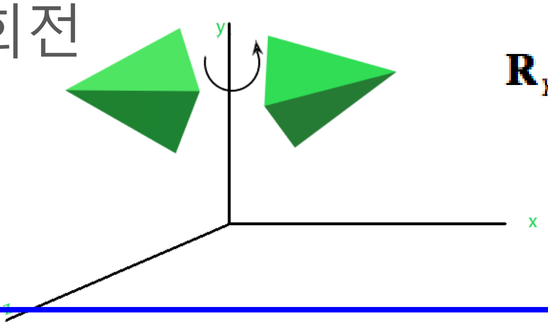
Transformation

- **Rotation matrix**
  - X축 기준 회전



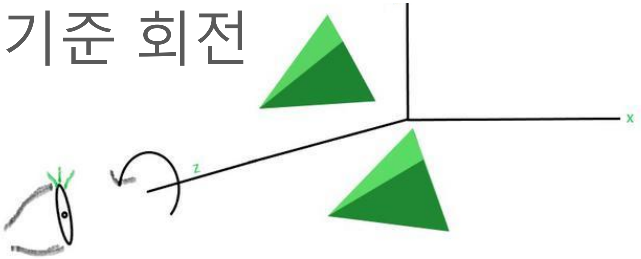
$$\mathbf{R}_X(\theta)=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - Y축 기준 회전



$$\mathbf{R}_Y(\theta)=\begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - Z축 기준 회전



$$\mathbf{R}_Z(\theta)=\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Goals

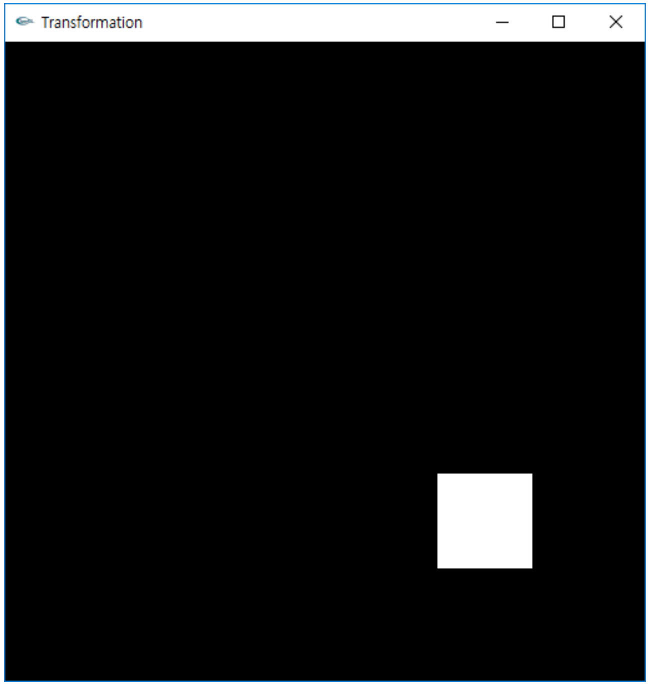
- OpenGL Basic Programming
- OpenGL Drawing Function
- Transformation Matrix
- Coding OpenGL Transformation
- Order matters in Transformations

Transformations: Translation

- void **glTranslatef**(GLfloat dx, GLfloat dy, GLfloat dz)
  - dx, dy, dz : distance of translate

Translation matrix

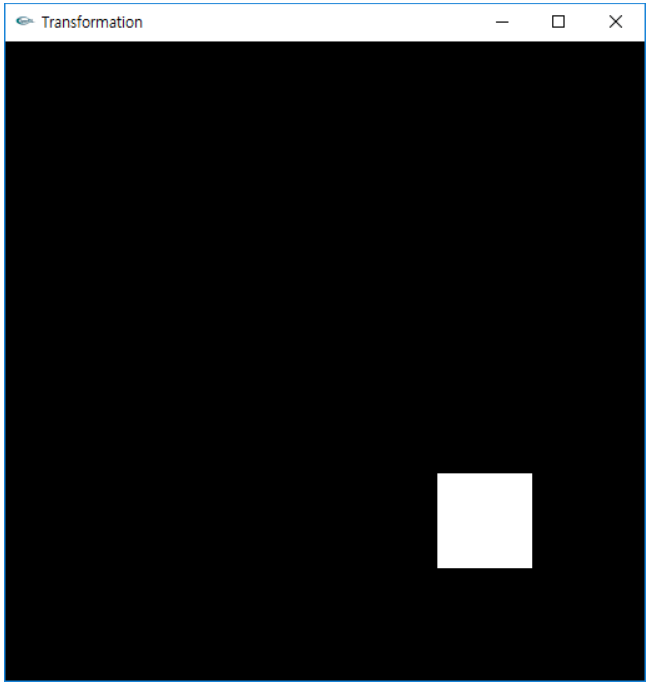
$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + dx \\ y + dy \\ z + dz \\ 1 \end{bmatrix}$$



Transformations: Translation

- void **glTranslatef**(GLfloat dx, GLfloat dy, GLfloat dz)
  - dx, dy, dz : distance of translate

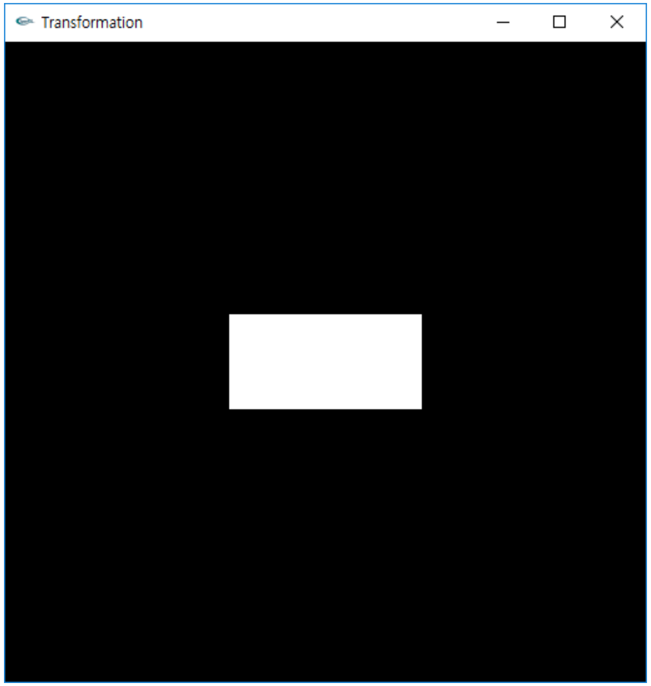
```
// Translate the objects by (0.5, -0.5, 0.0)
glTranslatef(0.5, -0.5, 0.0);

// Draw the solid cube
glutSolidCube(0.3);
```



Transformations: Scaling

- void **glScalef**(GLfloat sx, GLfloat sy, GLfloat sz)
  - sx, sy, sz : scale factor

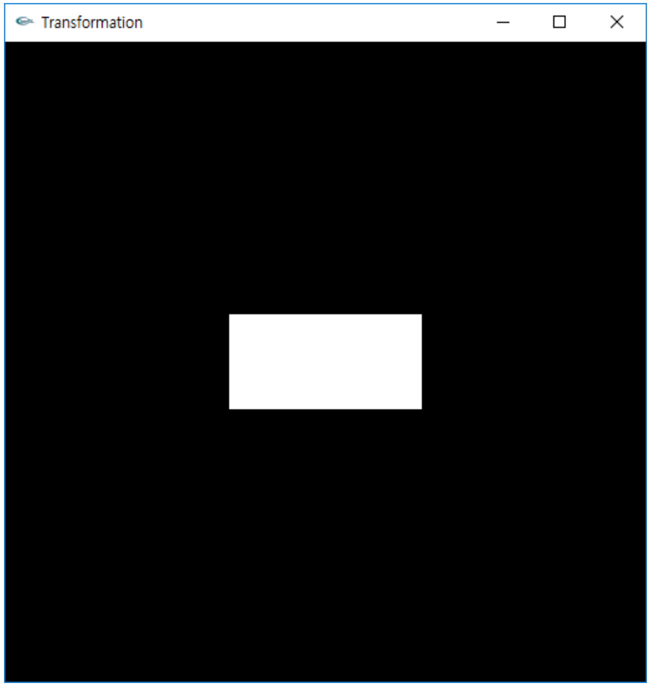
$$\begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x * sx \\ y * sy \\ z * sz \\ w \end{bmatrix}$$



Transformations: Scaling

- void **glScalef**(GLfloat sx, GLfloat sy, GLfloat sz)
  - sx, sy, sz : scale factor

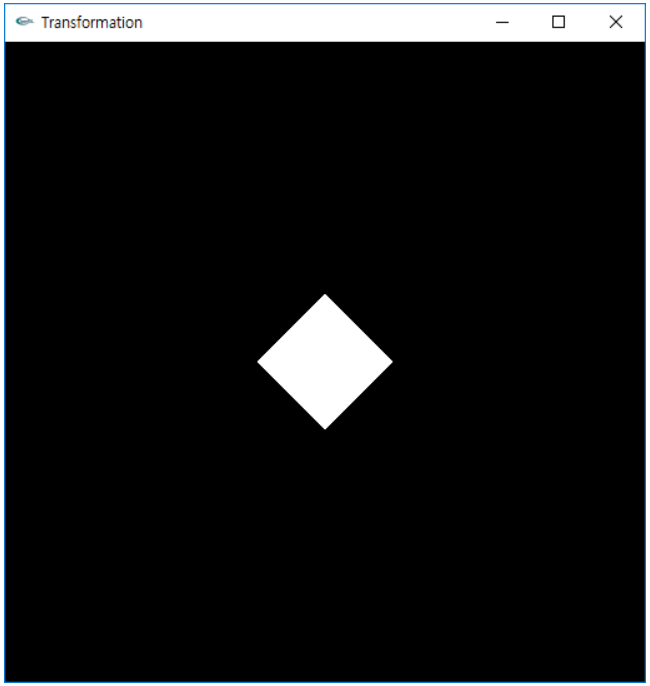
```
// Scale the objects by (2.0, 1.0, 1.0)
glScalef(2.0, 1.0, 1.0);

// Draw the solid cube
glutSolidCube(0.3);
```



Transformations: Rotation

- **void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)**
  - Angle : Rotation angle, by *degree*
  - x, y, z : Rotation axis

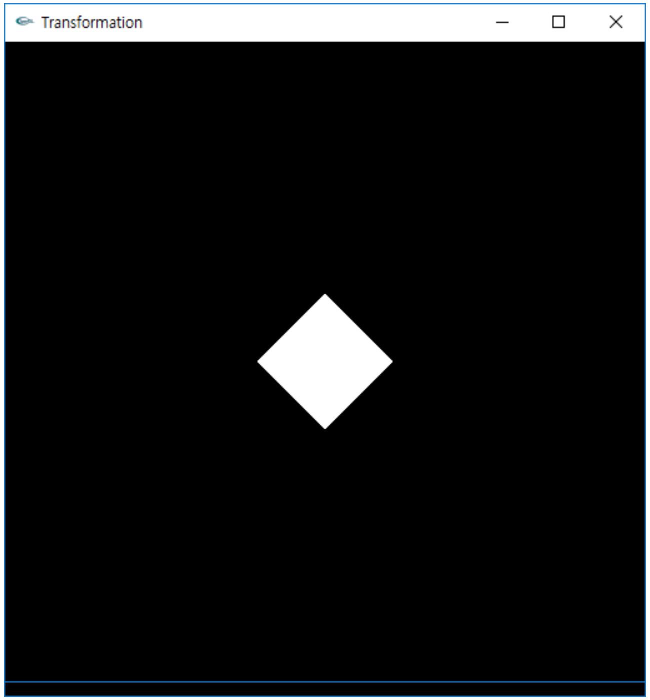
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Transformations: Rotation

- void **glRotatef**(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)
  - Angle : Rotation angle, by *degree*
  - x, y, z : Rotation axis

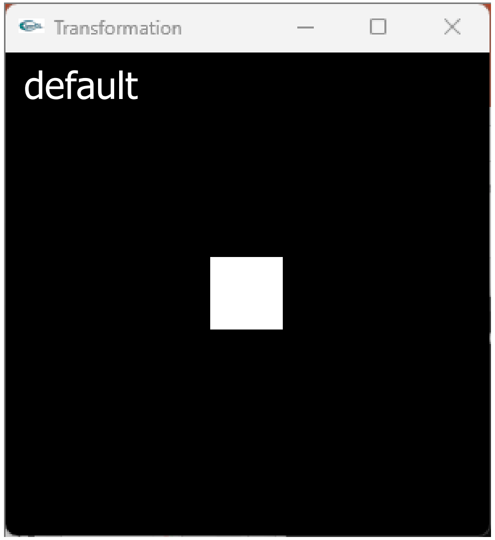
```
// z-Rotate the objects by 45 degree
glRotatef(45.0, 0.0, 0.0, 1.0);

// Draw the solid cube
glutSolidCube(0.3);
```



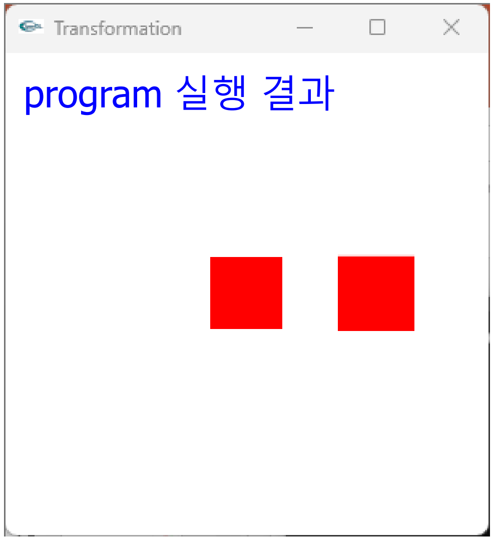
Drawing a Red Rectangle with white bg

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

void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
                                    glTranslatef(0.5, 0.0, 0.0);
        // Set Color
        glColor3f(1.0, 0.0, 0.0);          // red (R, G, B)

        // Draw the solid cube
        glutSolidCube(0.3);
        glFlush();
}

int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
        glutCreateWindow("Transformation");

        // Set Background Color
        glClearColor(1.0, 1.0, 1.0, 0.0);      // white (R, G, B, Alpha)

        glutDisplayFunc(display);
        glutMainLoop();

        return 0;
}
```

Display() 함수에 있는 것이 아니라 main()에 있어야 하는 이유?



default



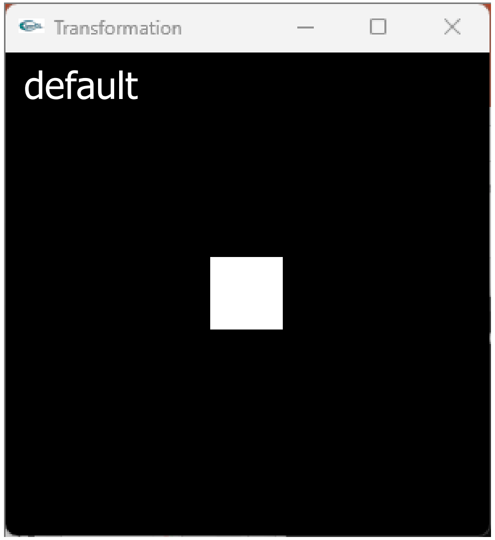
program 실행 결과

Mission

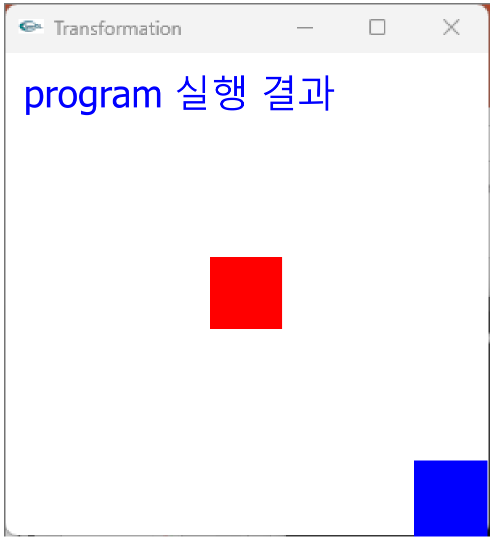
Drawing a **Blue** Rectangle with transformation

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

void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
                              glTranslatef(0.5, 0.0, 0.0);
        // Set Color
        glColor3f(1.0, 0.0, 0.0);          // red (R, G, B)

        // Draw the solid cube
        glutSolidCube(0.3);
        glFlush();
}

int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_SINGLE);
        glutCreateWindow("Transformation");

        // Set Background Color
        glClearColor(1.0, 1.0, 1.0, 0.0);     // white (R, G, B, Alpha)

        glutDisplayFunc(display);
        glutMainLoop();

        return 0;
}
```

Display() 함수에 있는 것이 아니라 main()에 있어야 하는 이유?



default



program 실행 결과

Goals

- OpenGL Basic Programming
- OpenGL Drawing Function
- Transformation Matrix
- Coding OpenGL Transformation
- Order matters in Transformations

Matrix Multiplication(4주차 Review)

- 다수의 행렬과 벡터의 곱은 **연속적인 Transformation**을 의미함

Transformation2

$$(\boldsymbol{M_1} * \boldsymbol{M_2}) * \vec{v} = M_1 * (M_2 * \vec{v})$$

Transformation1

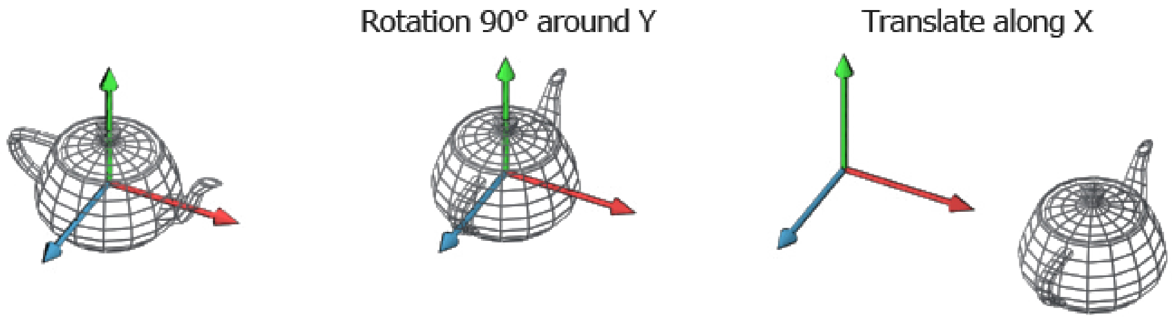
- Transformation은 제일 오른쪽 행렬에서 왼쪽으로 진행하면서 계산

Transformation Ordering

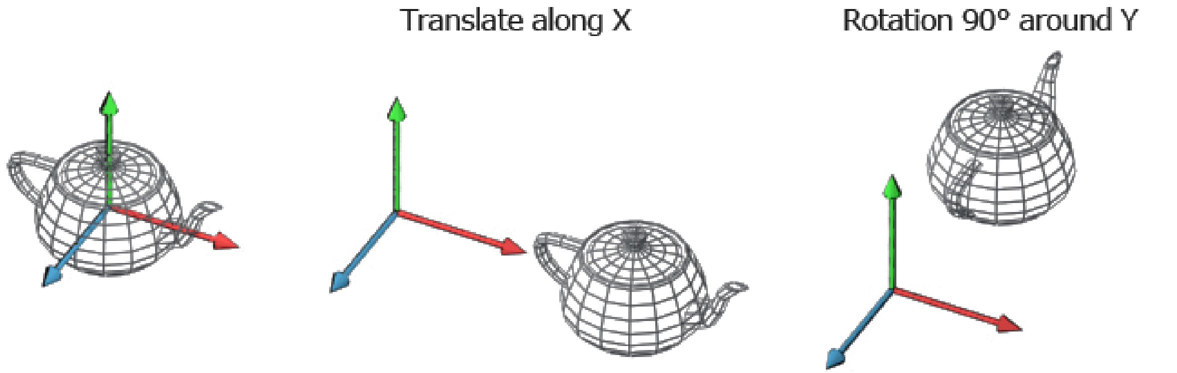
M = TRSv

Transformation 순서에 따라 결과가 달라진다 !!

M = TR

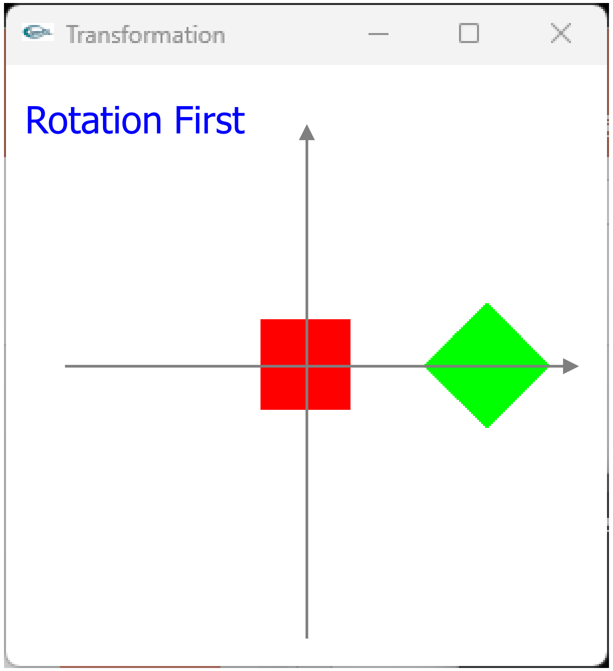


M = RT



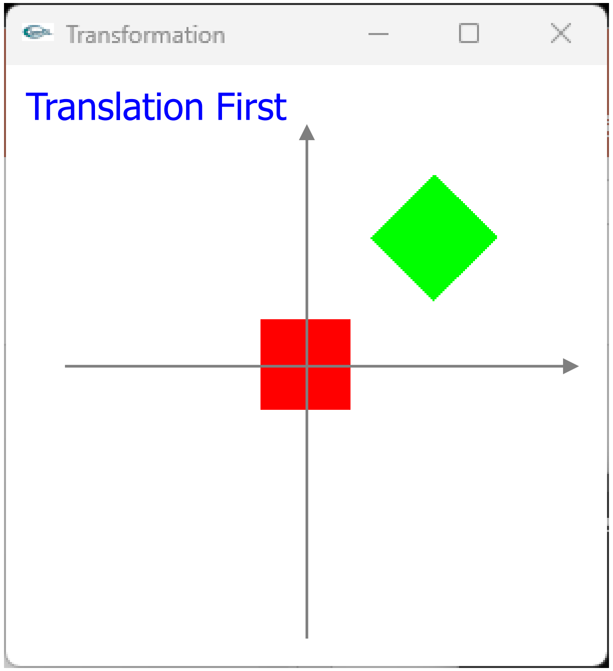
Rotation & Translation

```
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        // Red Cube
        glColor3f(1.0, 0.0, 0.0);
        glutSolidCube(0.3);

        // Translation & Rotation - 다각형에 가까운 변환 먼저 수행
        glTranslatef(0.6, 0.0, 0.0);     ----- (2)
        glRotatef(45.0, 0.0, 0.0, 1.0); ----- (1)

        // Green Cube
        glColor3f(0.0, 1.0, 0.0);
        glutSolidCube(0.3);

        glFlush();
}
```



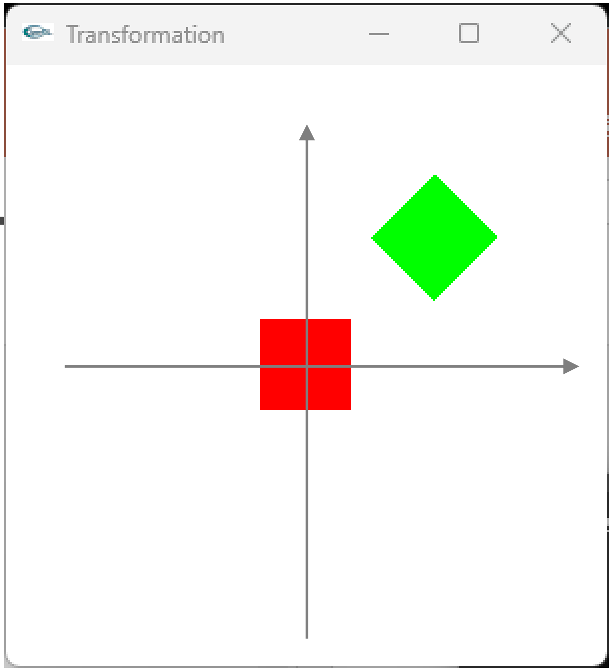
Translation & Rotation

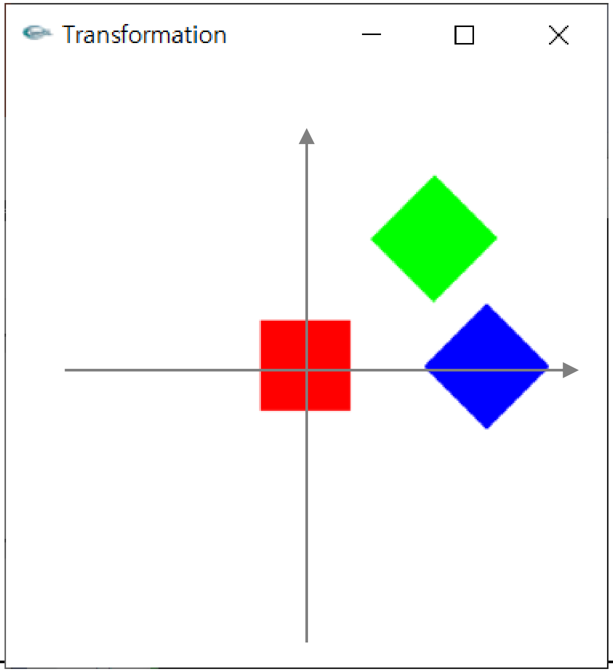
```
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        // Red Cube
        glColor3f(1.0, 0.0, 0.0);
        glutSolidCube(0.3);

        // Translation & Rotation - 다각형에 가까운 변환 먼저 수행
        glRotatef(45.0, 0.0, 0.0, 1.0); ----- (2)
        glTranslatef(0.6, 0.0, 0.0);    ----- (1)

        // Green Cube
        glColor3f(0.0, 1.0, 0.0);
        glutSolidCube(0.3);

        glFlush();
}
```



Transformation

```
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        // Red Cube
        glColor3f(1.0, 0.0, 0.0);
        glutSolidCube(0.3);

        // Translation & Rotation
        glRotatef(45.0, 0.0, 0.0, 1.0);
        glTranslatef(0.6, 0.0, 0.0);

        // Green Cube
        glColor3f(0.0, 1.0, 0.0);
        glutSolidCube(0.3);

        // Rotation & Translation
        glTranslatef(0.6, 0.0, 0.0);
        glRotatef(45.0, 0.0, 0.0, 1.0);

        // Blue Cube
        glColor3f(0.0, 0.0, 1.0);
        glutSolidCube(0.3);

        glFlush();
}
```

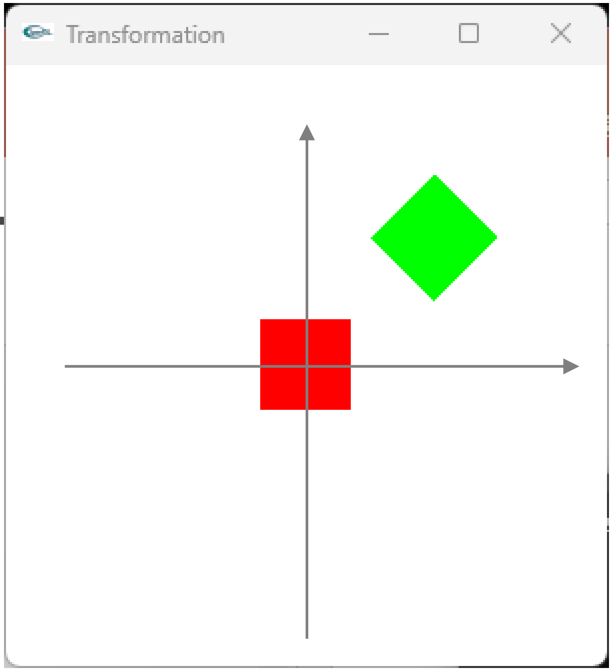


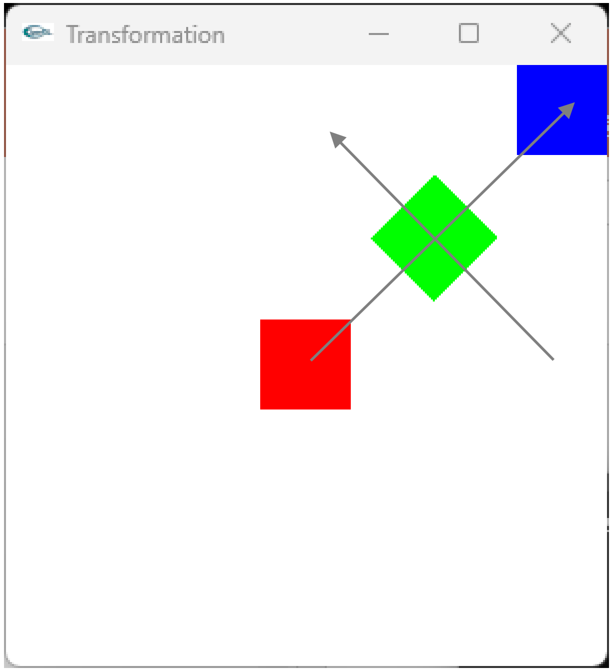


Transformation

```
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        // Red Cube
        glColor3f(1.0, 0.0, 0.0);
        glutSolidCube(0.3);

        // Translation & Rotation
        glRotatef(45.0, 0.0, 0.0, 1.0);
        glTranslatef(0.6, 0.0, 0.0);

        // Green Cube
        glColor3f(0.0, 1.0, 0.0);
        glutSolidCube(0.3);

        // Rotation & Translation
        glTranslatef(0.6, 0.0, 0.0);
        glRotatef(45.0, 0.0, 0.0, 1.0);

        // Blue Cube
        glColor3f(0.0, 0.0, 1.0);
        glutSolidCube(0.3);

        glFlush();
}
```

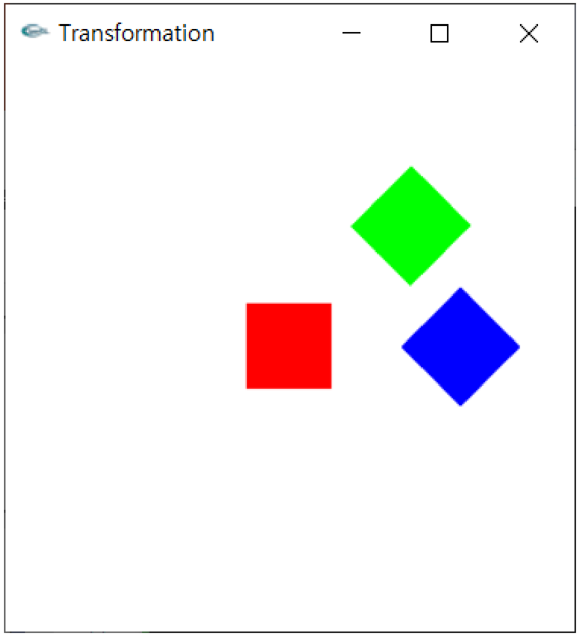
Transformation 누적





How to transform the cubes?

- Order matters in transformation



?

Inverse Transformation

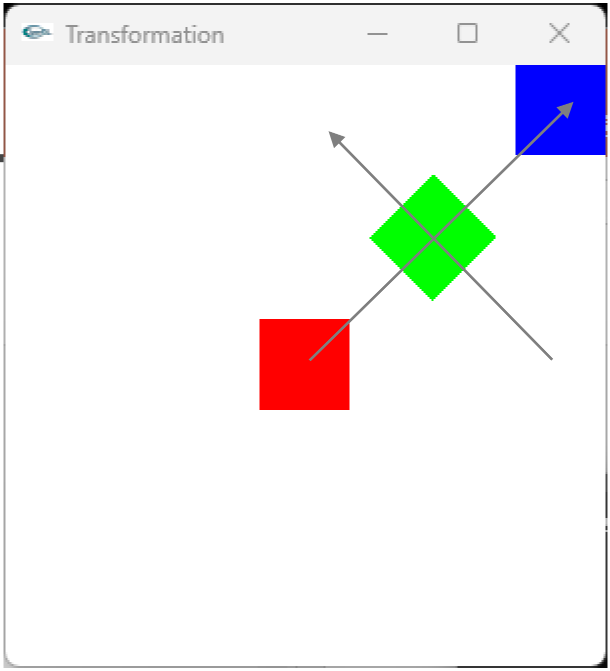
```
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        // Red Cube
        glColor3f(1.0, 0.0, 0.0);
        glutSolidCube(0.3);

        // Translation & Rotation
        glRotatef(45.0, 0.0, 0.0, 1.0);
        glTranslatef(0.6, 0.0, 0.0);

        // Green Cube
        glColor3f(0.0, 1.0, 0.0);
        glutSolidCube(0.3);

        // Inverse Transformation
        // Inverse Rotation, Inverse Translation
        glTranslatef(-0.6, 0.0, 0.0);
        glRotatef(-45.0, 0.0, 0.0, 1.0);

        // Rotation & Translation
        glTranslatef(0.6, 0.0, 0.0);
        glRotatef(45.0, 0.0, 0.0, 1.0);

        // Blue Cube
        glColor3f(0.0, 0.0, 1.0);
        glutSolidCube(0.3);

        glFlush();
}
```

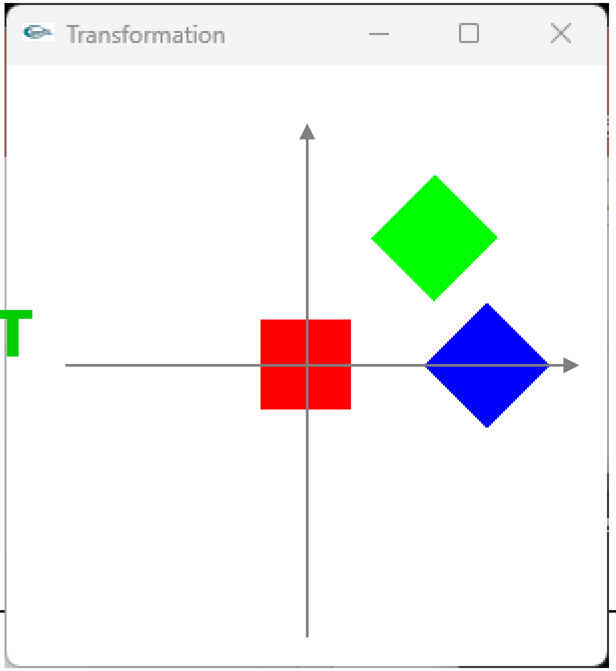
*순서 중요해요 !

$T^{-1}R^{-1}$ RT

원점으로 이동 이전 Transformation의 반대 순서로

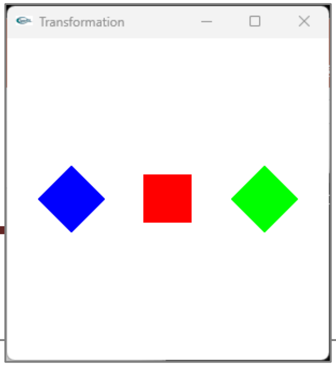
TR$T^{-1}R^{-1}$ RT





Mission

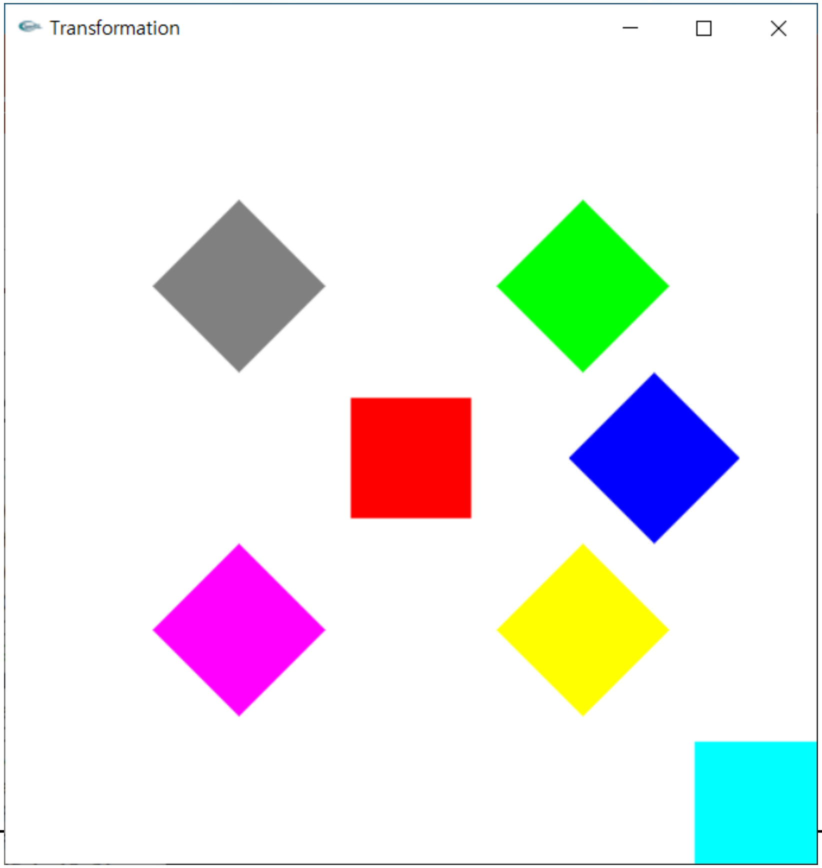
Exercise.



```
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();

        // Red Cube
        glColor3f(1.0, 0.0, 0.0);
        glutSolidCube(0.3);

        // Transformation



        // Green Cube
        glColor3f(0.0, 1.0, 0.0);
        glutSolidCube(0.3);

        // Transformation



        // Blue Cube
        glColor3f(0.0, 0.0, 1.0);
        glutSolidCube(0.3);

        glFlush();

}
```

과제 2. Transformation

- 오늘 배운 Transformation을 활용하여 다음 화면을 렌더링하세요. (red를 제외한 모든 사각형은 transformation을 사용하여 위치시킬 것)



Thank you !