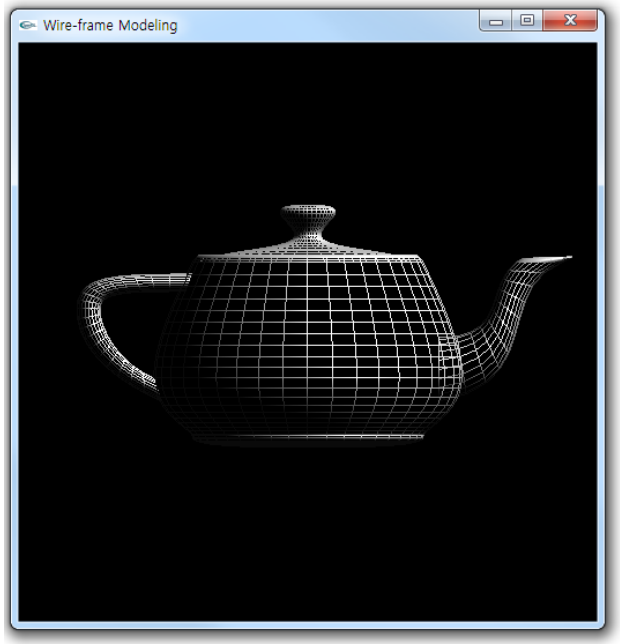
OpenGL Modeling
Primitive Drawing

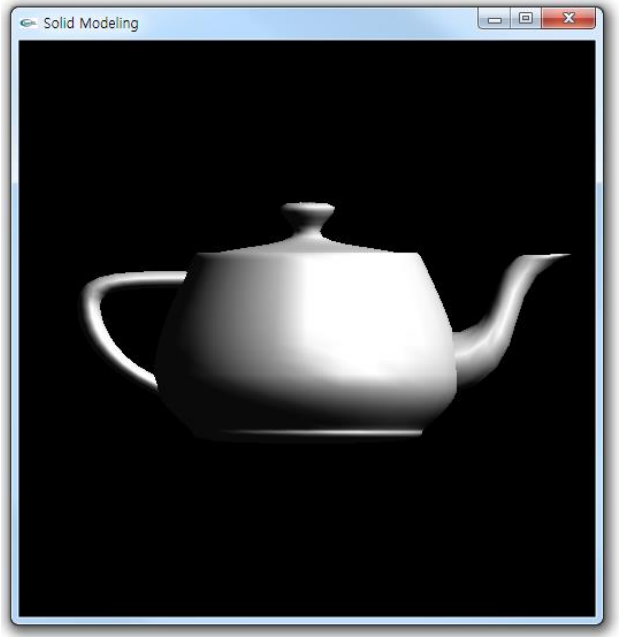
동아대학교 컴퓨터AI공학부

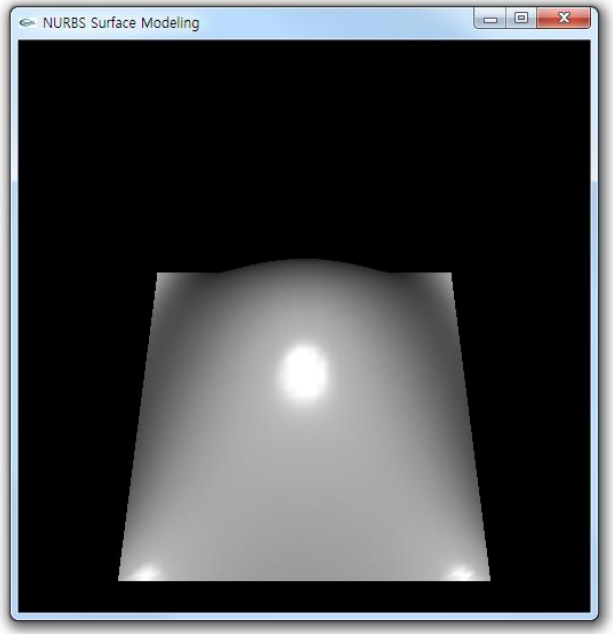
박영진

Modeling

- **컴퓨터그래픽스 기술을 이용해 3D 모델(모형)을 만드는 것**
  - Wireframe modeling
  - Solid modeling
  - NURBS surface modeling







GL Modeling

- **점을 그리는 코드**

glBegin(**GL_POINTS**);

glVertex2f(0, 0);

glEnd();

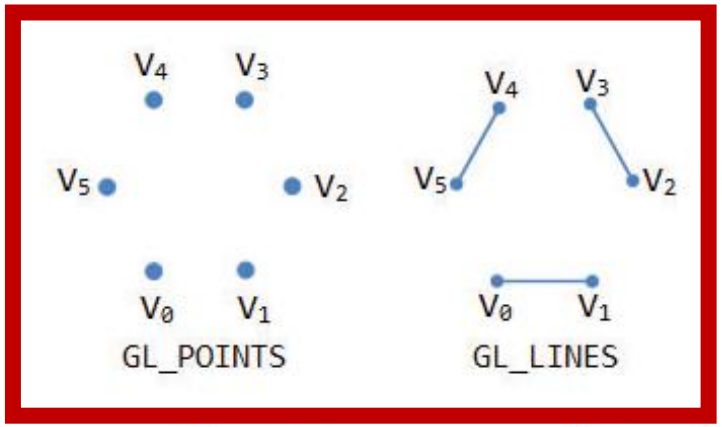
- **선(라인)을 그리는 코드**

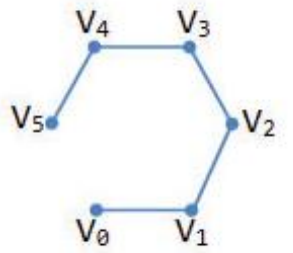
glBegin(**GL_LINES**);

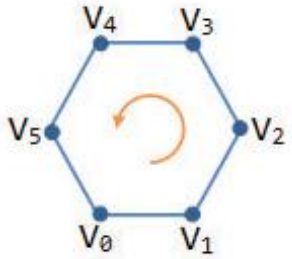
glVertex2f(0, 0);

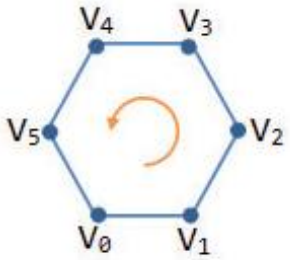
glVertex2f(1, 0);

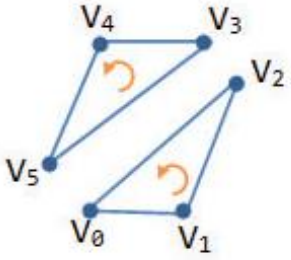
glEnd();

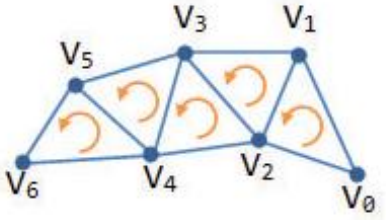


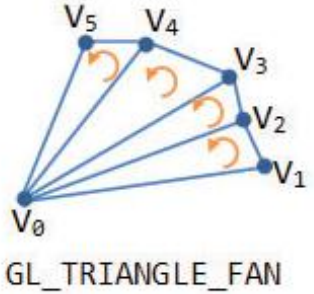


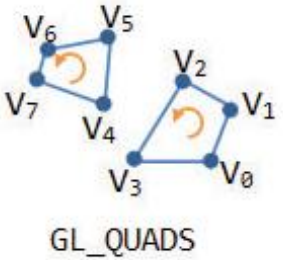


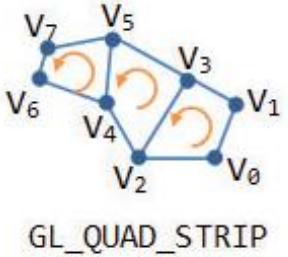










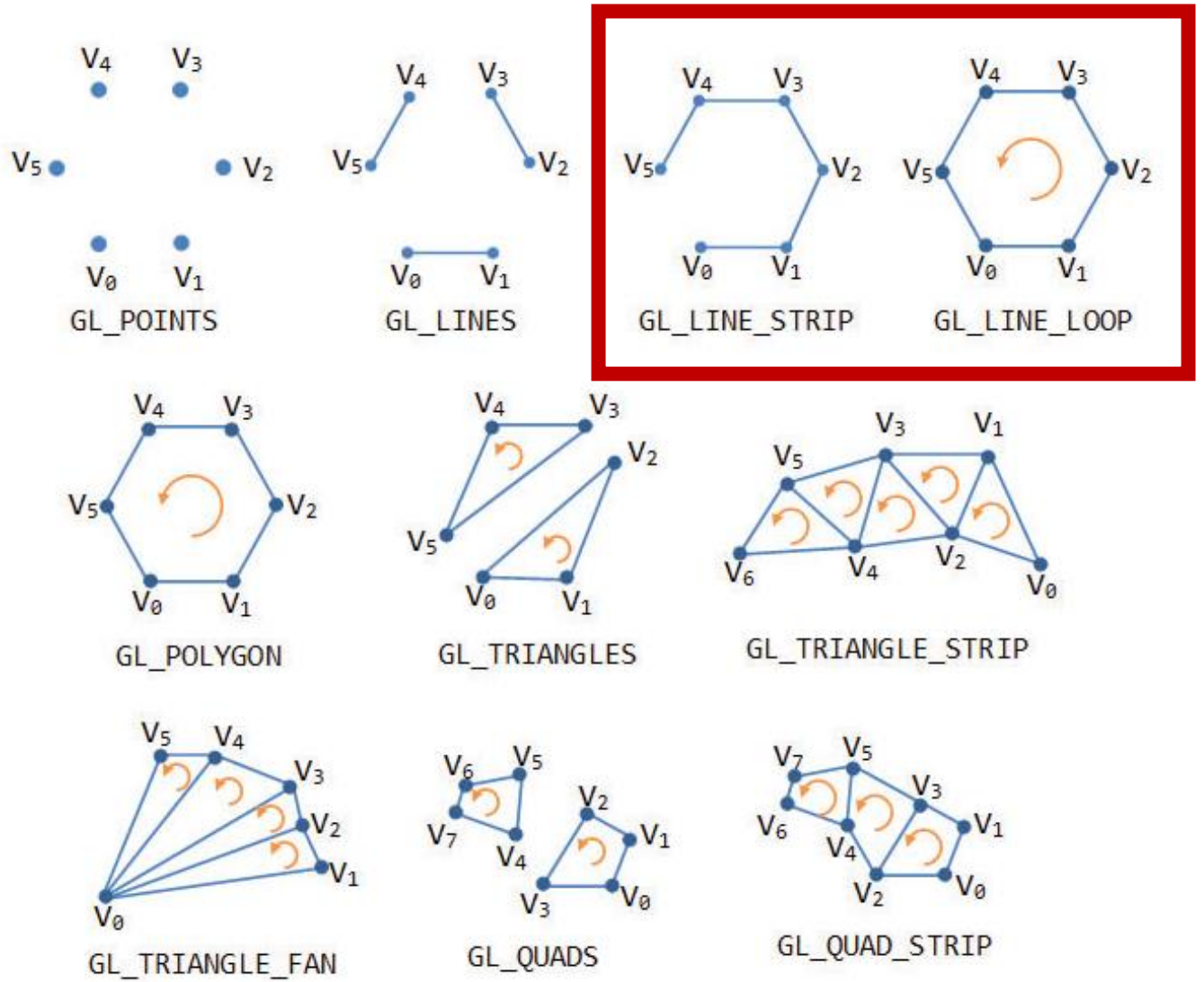


- **선을 이어서 그리는 코드**

```
glBegin(GL_LINE_STRIP);
glVertex2f(0, 0);
glVertex2f(1, 0);
glVertex2f(1, 1);
glEnd();
```

- **선을 그리고 순환(loop)을 그리는 코드**

```
glBegin(GL_LINE_LOOP);
glVertex2f(0, 0);
glVertex2f(1, 0);
glVertex2f(1, 1);
glEnd();
```



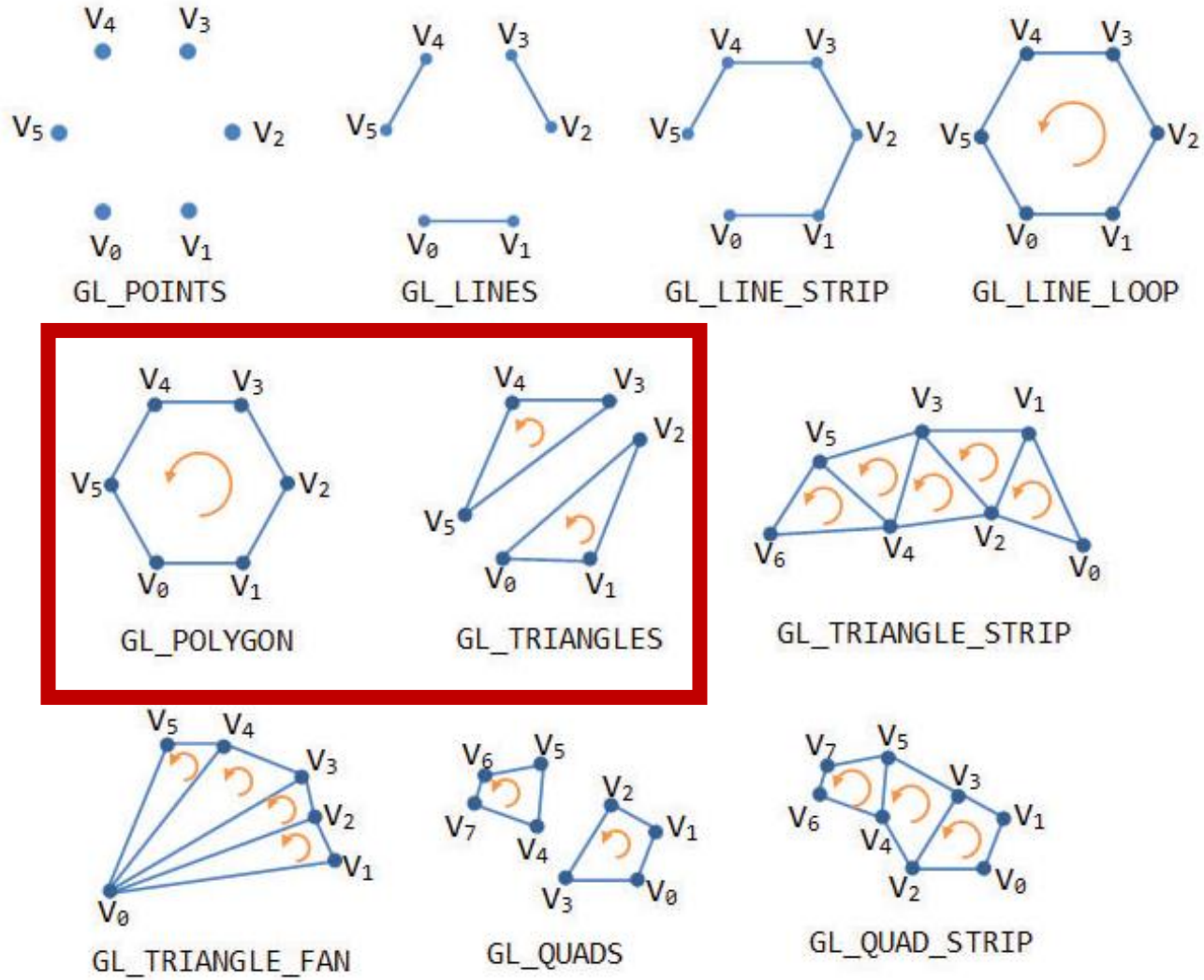
GL Modeling

- **다각형을 그리는 코드**

```
glBegin(GL_POLYGON);
glVertex2f(0, 0);
glVertex2f(1, 0);
glVertex2f(1, 1);
glEnd();
```

- **삼각형을 그리는 코드**

```
glBegin(GL_TRIANGLES);
glVertex2f(0, 0);
glVertex2f(1, 0);
glVertex2f(1, 1);
glEnd();
```

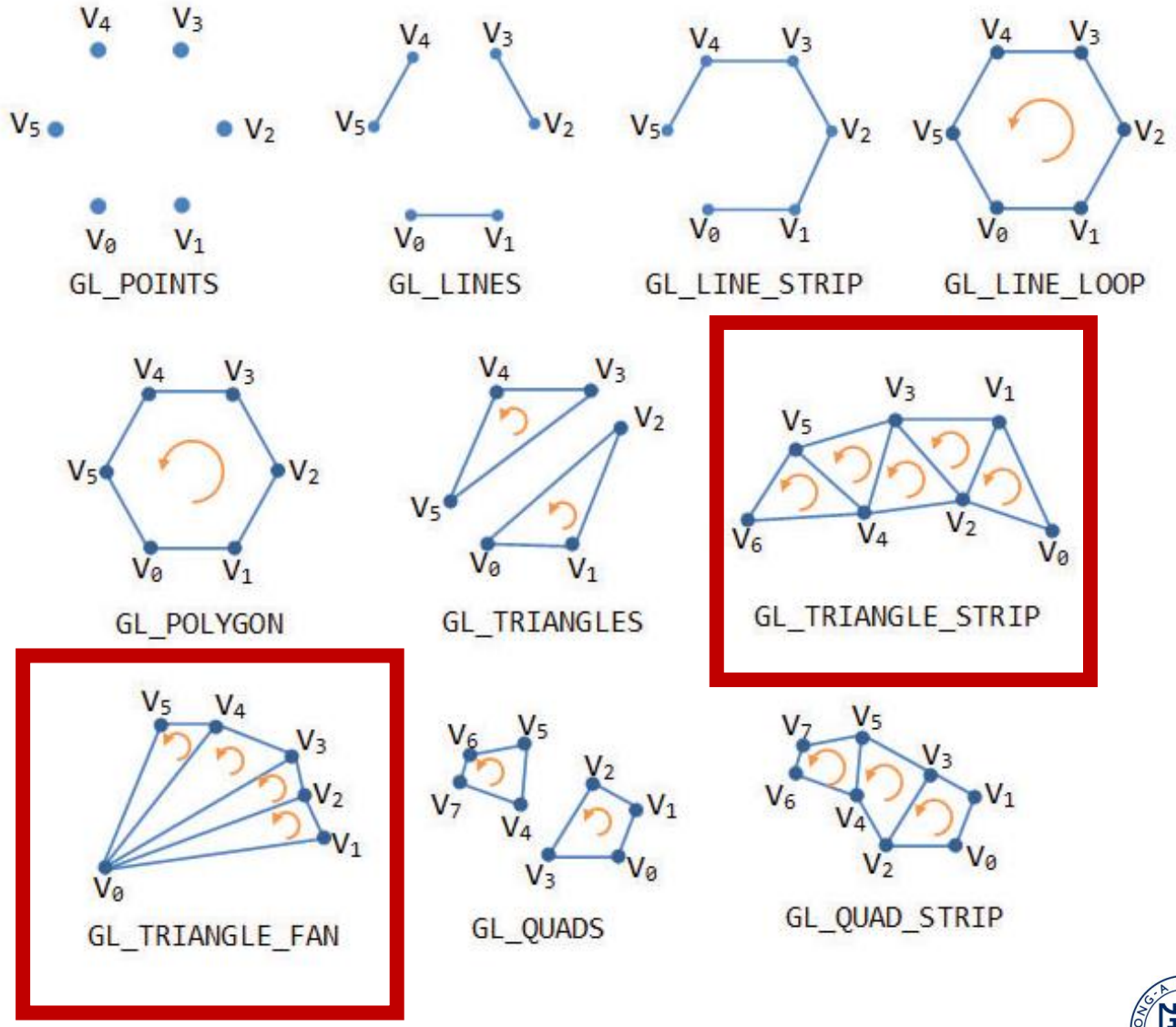


- **삼각형을 이어서 그리는 코드**

glBegin(**GL_TRIANGLE_FAN**);
glVertex2f(0, 0);
glVertex2f(1, 0);
glVertex2f(1, 1);
glEnd();

- **삼각형을 이어서 그리는 또다른 코드**

glBegin(**GL_TRIANGLE_FAN**);
glVertex2f(0, 0);
glVertex2f(1, 0);
glVertex2f(1, 1);
glEnd();



GL Modeling

Value	Description
GL_POINTS	각 Vertex들을 하나의 Point로 표현한다.
GL_LINES	Vertex들을 두 개씩 묶어서 Line을 만든다.
GL_LINE_STRIP	Line들을 한 줄로 연결한다.
GL_LINE_LOOP	마지막 Vertex와 첫 번째 Vertex를 연결하는 Line이 추가된 GL_LINE_STRIP
GL_TRIANGLES	세 개의 Vertex를 묶어서 삼각형을 만든다.
GL_TRIANGLE_STRIP	삼각형들을 길게 연결한다.
GL_TRIANGLE_FAN	삼각형들을 부채 모양으로 연결한다.
GL_QUADS	Vertex들을 네 개씩 묶어서 만든 네 개의 모서리를 가진 Polygon을 만든다.
GL_QUAD_STRIP	사각형들을 길게 연결한다.
GL_POLYGON	단순, 볼록 Polygon

Callback Programming

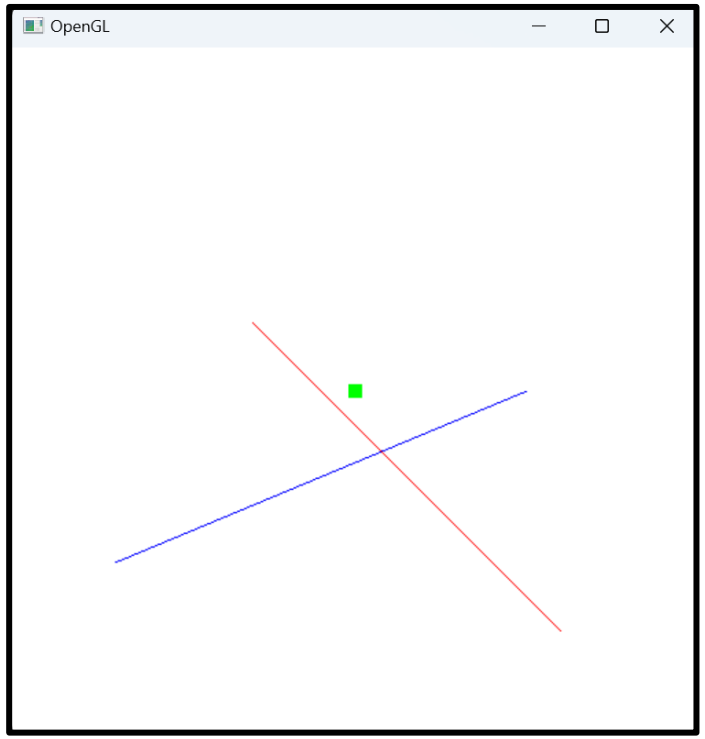
- Display Callback
- Keyboard Callback
- Reshape Callback
- Mouse Callback
- Menu Callback
- Idle Callback & Double Buffering
- Timer Callback

[실습] 선분과 선분의 교차

- 2차원 점을 저장하기 위한 Vec2 구조체, 선분을 정의하기 위한 linePt

```
struct Vec2 {
    float x, y;
};

Vec2 linePt[4] = {
    {-0.3f, 0.2f},
    {0.6f, -0.7f},
    {-0.7f, -0.5f},
    {0.5f, 0.0f}
};
```



[실습] 선분과 선분의 교차

- Display callback

```
void display() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2f(linePt[0].x, linePt[0].y);
    glVertex2f(linePt[1].x, linePt[1].y);
    glEnd();

    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINES);
    glVertex2f(linePt[2].x, linePt[2].y);
    glVertex2f(linePt[3].x, linePt[3].y);
    glEnd();

    /* Calculate the interseciton point! */
    glColor3f(0.0, 1.0, 0.0);
    glPointSize(10.0);
    glBegin(GL_POINTS);
    glVertex2f(0, 0); /* Need to change the value. */
    glEnd();

    glutSwapBuffers();
}
```

```
Vec2 linePt[4] = {
    {-0.3f, 0.2f},
    {0.6f, -0.7f},
    {-0.7f, -0.5f},
    {0.5f, 0.0f}
};
```

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(1480, 100);


    glutCreateWindow("OpenGL");
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();

    return 0;
}
```

[실습] 선분과 선분의 교차

- Keyboard callback

```
void keyboard(unsigned char key, int x, int y) {

    switch (key) {
    case 'w':
        linePt[0].y += 0.1f;
        linePt[1].y += 0.1f;
        break;
    case 's':
        linePt[0].y -= 0.1f;
        linePt[1].y -= 0.1f;
        break;
    case 'a':
        linePt[0].x -= 0.1f;
        linePt[1].x -= 0.1f;
        break;
    case 'd':
        linePt[0].x += 0.1f;
        linePt[1].x += 0.1f;
        break;
    case 27: // ESC
        exit(0);
        break;
    }
    glutPostRedisplay();
}
```

```
Vec2 linePt[4] = {
    {-0.3f, 0.2f},
    {0.6f, -0.7f},
    {-0.7f, -0.5f},
    {0.5f, 0.0f}
};
```

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(1480, 100);


    glutCreateWindow("OpenGL");
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();

    return 0;
}
```

[실습] 선분과 선분의 교차 – 수행과제

기본과제 (2점)

- 선분이 교차하는 지점에 초록색 점을 (프로그래밍으로 계산하여) 그리기
- 키보드 WSAD에 따라 선분이 이동, 실시간으로 교차점이 곧바로 계산되어 다시 그려져야 함

도전과제 (1점)

- 기본과제를 수행하며 발생 가능한 예외의 경우를 모두 나열하고 프로그래밍으로 예외처리를 보이시오.
  (단, 선분을 구성하는 점의 위치는 더 이상 키보드 콜백에 제한받지 않고, 다양한 위치로 이동 가능하다고 가정)
  - 발생 가능한 예외 사례 : 선분을 직선으로 간주하여 교차점 계산을 구현하는 경우, 선분 범위 밖의 영역에서 교차가 발생할 수 있음

제출물 PDF 보고서, 표지 X, 3장 이내로 제출, 개인정보 뒤 도전과제 수행여부 표시해야 함.

마감기한 3/15(금) 23:59

Any Questions?