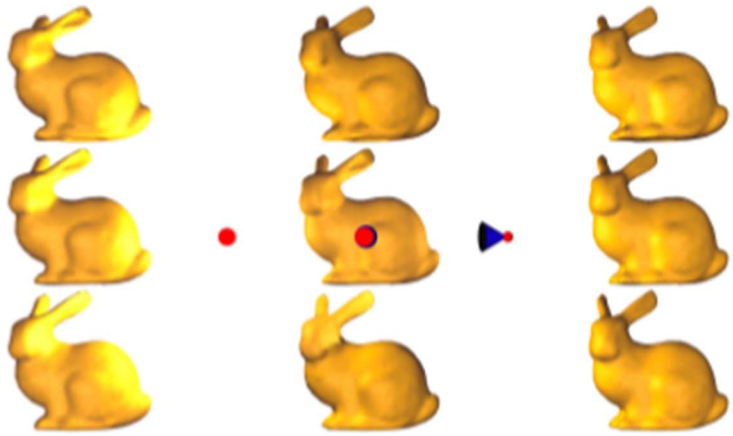
Computer Graphics



Interaction and Animation

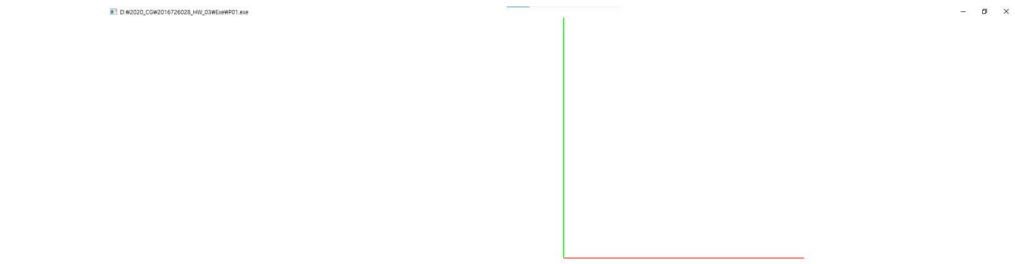
P03

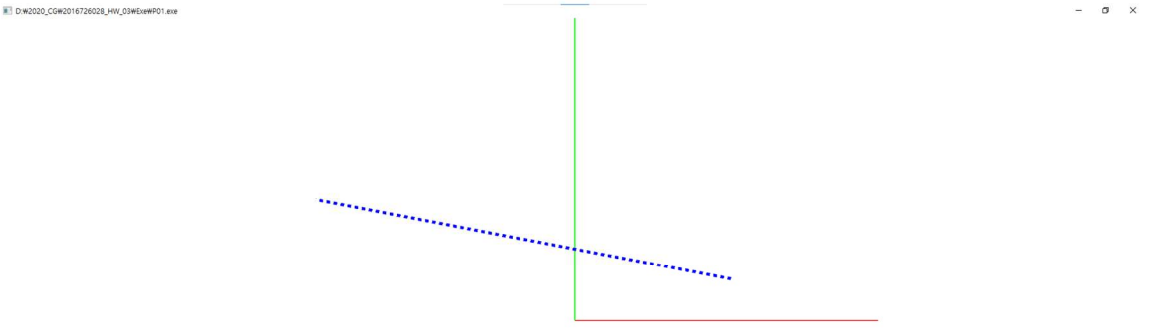
이민재 | Computer Graphics [심화전공실습 1] | 2020/09/20

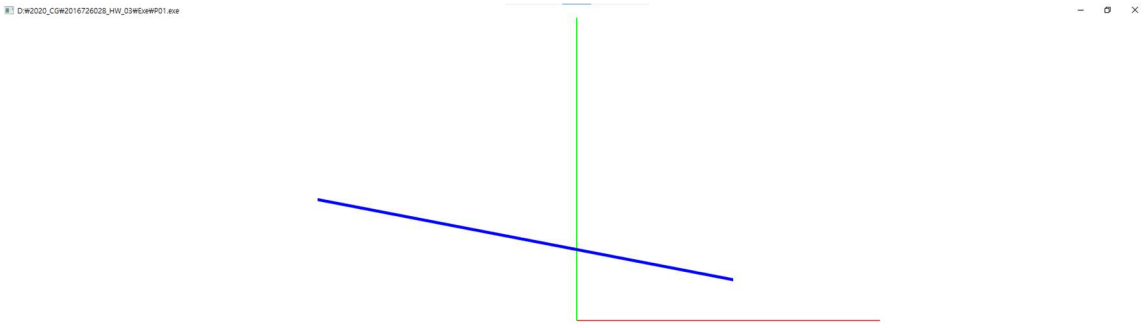
	P01	P02	E01	E02	E03	TOTAL
SCORE	1	1	1	1	1	5

P01 (Mouse input for drawing a line segment)

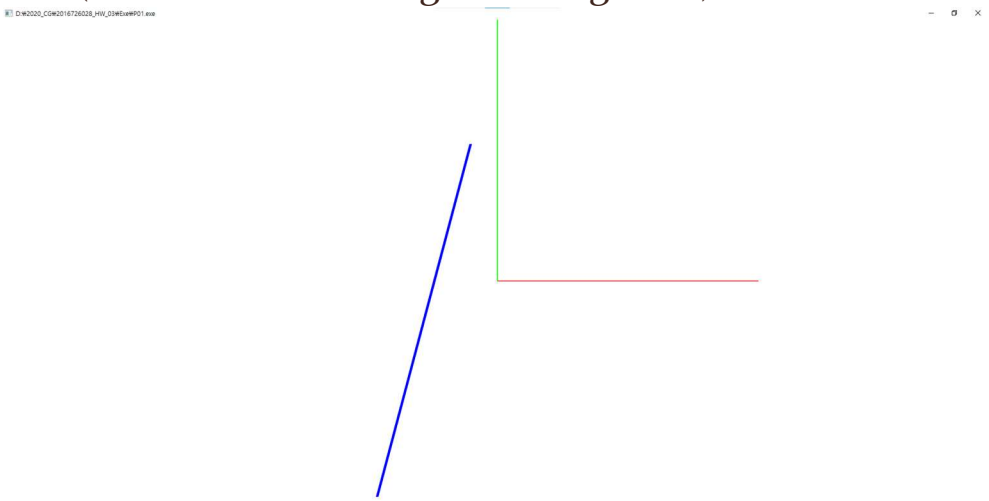
<SNAPSHOT>

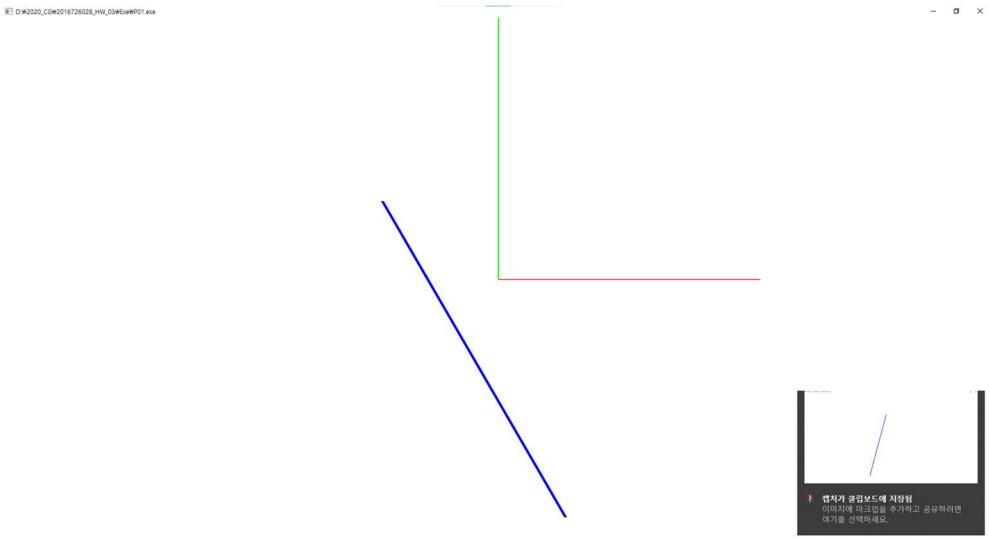


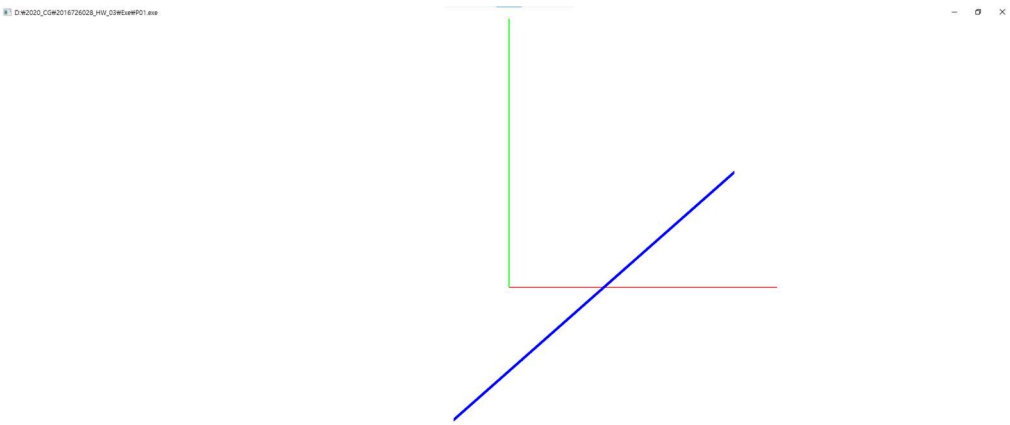




P02 (Timer for animating a line segment)

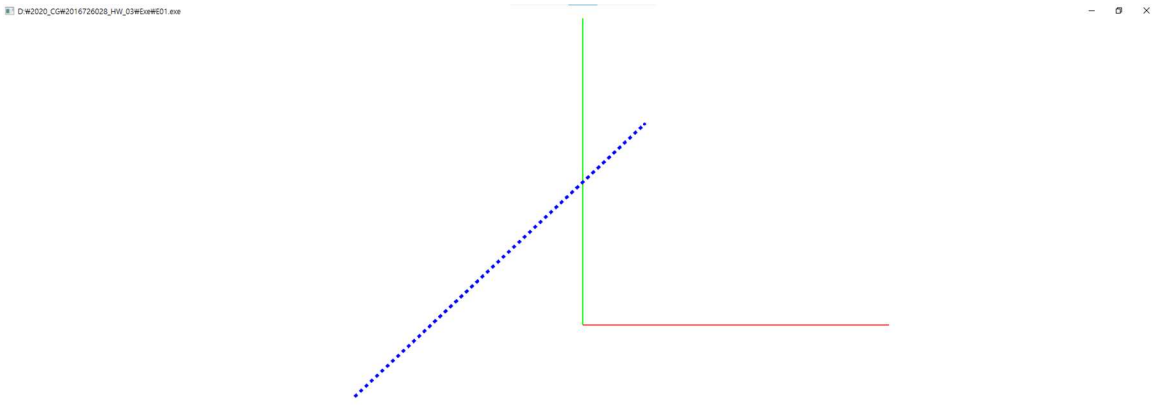


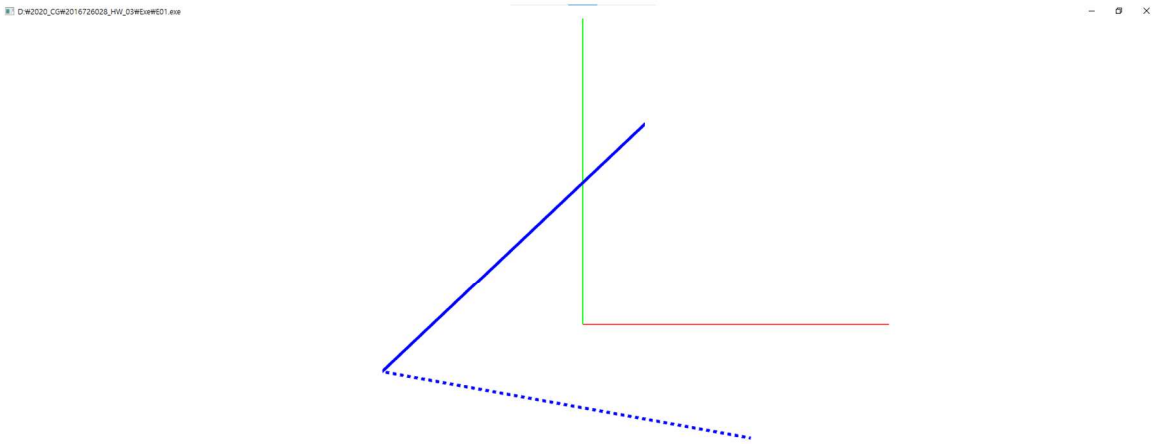


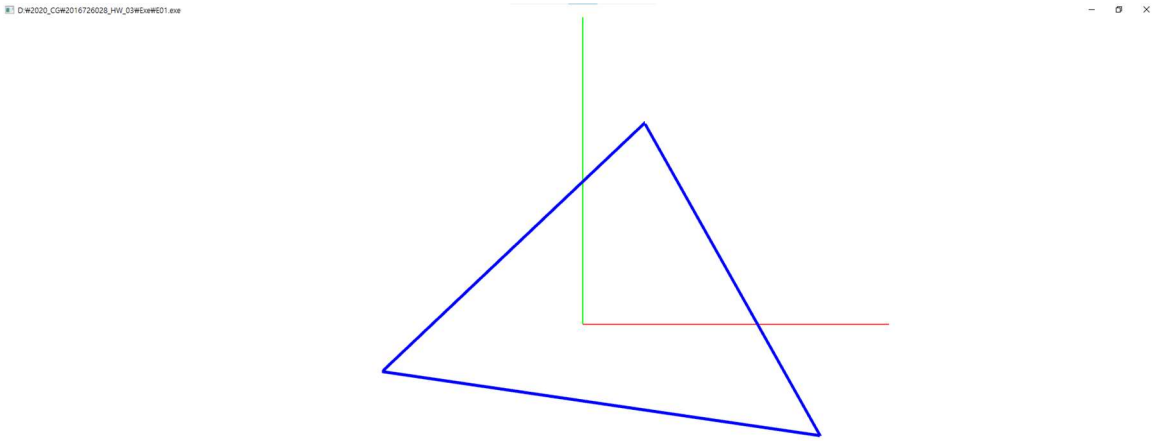


E01 (Draw a triangle by clicking the left mouse button 3 times)

<SNAPSHOT>







<EXPLANATION>

3 개의 vertex 를 이용하여 단계적으로 삼각형을 그리는 과제이기 때문에, 다음과 같은 순서로 구성하였다.

1. 첫번째 클릭의 경우 점을 생성한다. (GL_POINTS 이용)
2. 두번째 클릭의 경우, 첫번째 클릭한 위치 (p[0])와 두번째 클릭한 위치(p[1])를 시작점과 끝점으로 하는 선분을 그린다. (GL_LINES 이용)
3. 세번째 클릭의 경우, 앞의 두 위치와 세번째 클릭한 위치의 좌표를 이용하여 삼각형을 그린다. (GL_TRIANGLES)

이를 위해 기존의 예시 코드에서 전역 변수 float point 를 point[3][2] = {{0,0}, {0,0}, {0,0}} 으로 수정하였다. 또한 클릭의 횟수에 따른 동작을 구분하기 위하여 클릭 횟수를 저장하는 변수 c_count 를 선언하였다.

```
//Endpoints of the line segment
float point[3][2] = { {0,0}, {0,0}, {0,0} };
//Max Click count
int c_count = 0;
```

그 다음 마우스버튼 이벤트 **함수를 c_count 값에 동기화되어 작동**하도록 다음과 같은 조건문을 추가하였다.

```
switch (c_count)
	{
	case 3:
	case 0:
		// the left button of the mouse is pressed
		if (action == GLFW_PRESS && button == GLFW_MOUSE_BUTTON_LEFT)
		{
			inputMode = InputMode::CLICK; // click starts
		}
		// The left button of the mouse is released;
		if (action == GLFW_RELEASE && button == GLFW_MOUSE_BUTTON_LEFT)
		{
			inputMode = InputMode::COMPLETE; //click ends
			point[0][0] = xw; point[0][1] = yw; // 1st point
			c_count = 1; //increase count count 3 -> count 1
		}
		break;
	case 1:
		... (생략)
			c_count++; //increase count
		}
		break;
	case 2:
		... (생략)
		c_count++; //increase count
		}
		break;
```

즉, 마우스 클릭 횟수가 올라갈수록 case 별로 일어나는 이벤트를 정의하였으며 **클릭 횟수가 3 번이 넘어간다면 처음상태 (case 1)로 돌아가게 된다.**

또, 클릭을 **press** 와 **release** 로 구분하여 E02 에서 활용하는 COMPLETE 상태와 CLICK 상태를 구분하였다.

마우스 왼쪽 버튼이 올라갔을 때 **커서의 좌표를 각 case 별로 각 3 개의 vertex** 에 저장한다.

이렇게 저장한 vertex 의 좌표들은 render 함수에서 다음과 같이 활용한다.

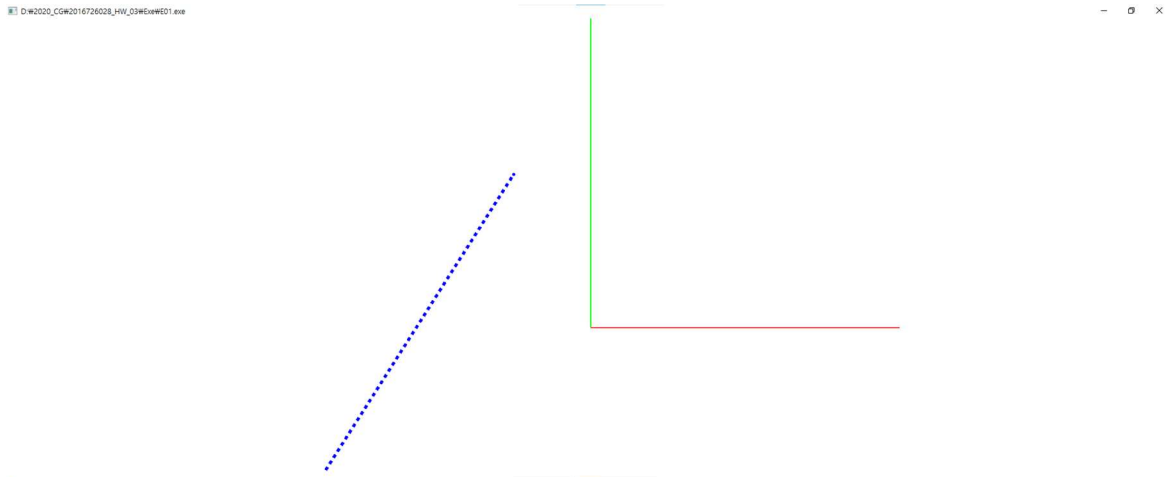
```
if (inputMode > InputMode::NONE && c_count == 1)
	{
		glBegin(GL_POINTS);
		glVertex2f(point[0][0], point[0][1]);
		glEnd();
	}
	else if (inputMode > InputMode::NONE && c_count == 2)
	{
		glDisable(GL_LINE_STIPPLE);
		glBegin(GL_LINES);
		glVertex2f(point[0][0], point[0][1]);
		glVertex2f(point[1][0], point[1][1]);
		glEnd();
	}
	else if (inputMode > InputMode::NONE && c_count == 3)
	{
		glBegin(GL_TRIANGLES);
		glVertex2f(point[0][0], point[0][1]);
		glVertex2f(point[1][0], point[1][1]);
		glVertex2f(point[2][0], point[2][1]);
		glEnd();

	}
```

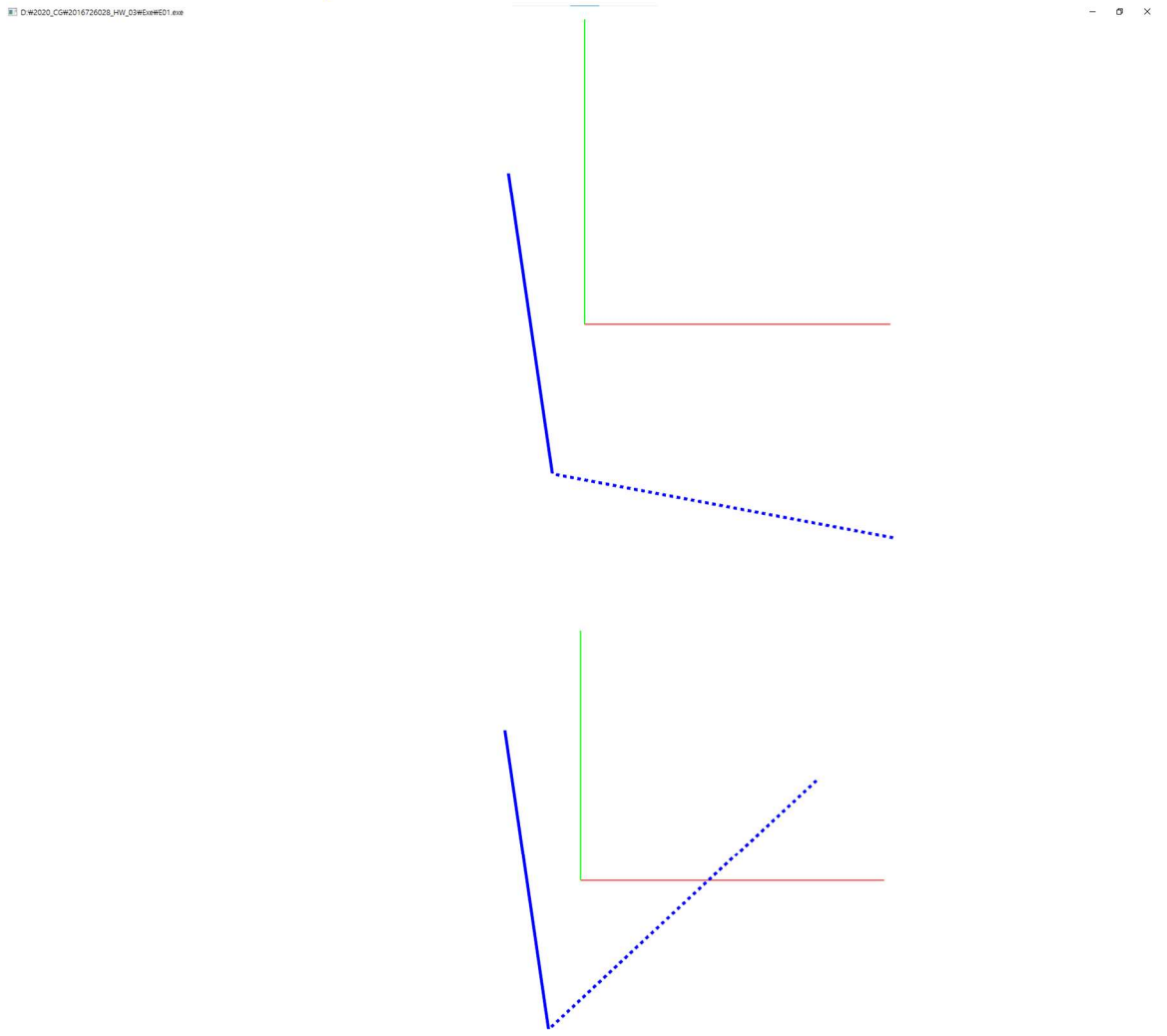
c_count 의 값에 따라 각각 point[0]으로 점 , point[0],point[1]로 선 , point[0],point[1],point[2]를 사용한 삼각형을 화면에 출력한다.

두번째 else if 문의 glDisable(GL_LINE_STIPPLE)은 E02 의 **점선이 클릭 후에는 선분으로 보이게 하도록** 하기 위함이다.

E02 (Draw dashed lines while moving the mouse pointer after the first click)

<SNAPSHOT>





<EXPLANATION>

첫번째 클릭 을 시작점으로 ,그 다음 클릭 이전의 마우스 위치를 끝점으로 하는 점선(두번째 클릭도 동일 과정) 을 생성하기 위해서 다음과 같이 MouseMove 함수를 구성하였다.

```
void mouseMove(GLFWwindow* window, double x, double y)
{
	IsMouseMove = true;
	if (inputMode == InputMode::COMPLETE && c_count == 1)
	{
		//Screen coordinate 2 world coordinate conversion
		screen2world((float)x, (float)y, point[1][0], point[1][1]);
	}
	else if (inputMode == InputMode::COMPLETE && c_count == 2)
	{
		//Screen coordinate 2 world coordinate conversion
		screen2world((float)x, (float)y, point[2][0], point[2][1]);
	}
}
```

버튼 이벤트와 동일하게 c_count 값을 조건으로 구분되며, inputmode 가 COMPLETE 상태일경우 (**버튼 릴리스 이벤트 일어난 이후**) **현 마우스 커서의 위치를 점선 선분의 끝점으로** 각 버텍스에 대입하여 준다.

render 함수에서는 다음과 같은 내용을 추가하였다.

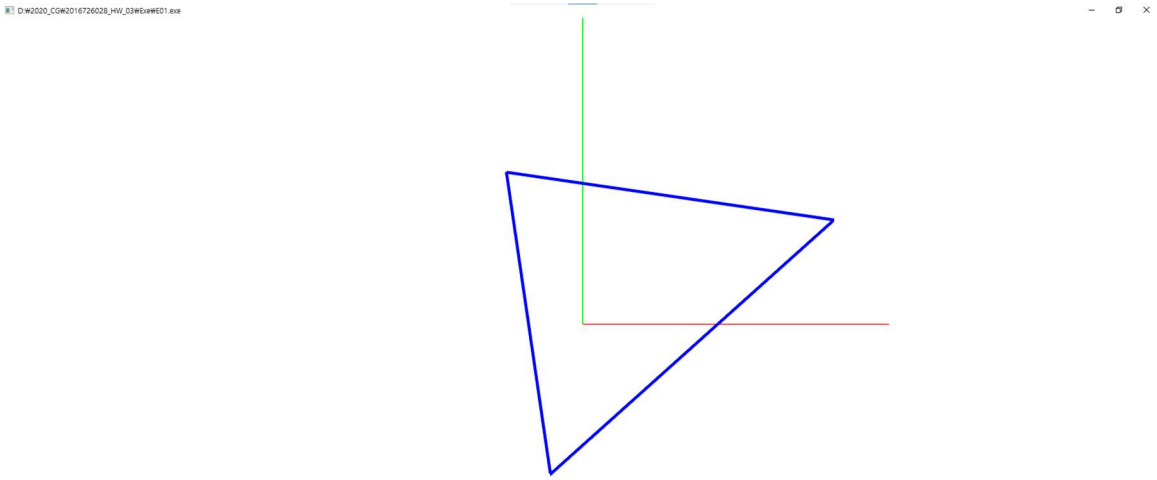
```
if (inputMode == InputMode::COMPLETE && c_count == 1 && IsMouseMove == true)
	{
		glEnable(GL_LINE_STIPPLE);
		glLineStipple(int(3 * dpiScaling), 0xcccc);
		glBegin(GL_LINES);
		glVertex2f(point[0][0], point[0][1]);
		glVertex2f(point[1][0], point[1][1]);
		glEnd();
	}
… (생략)
```

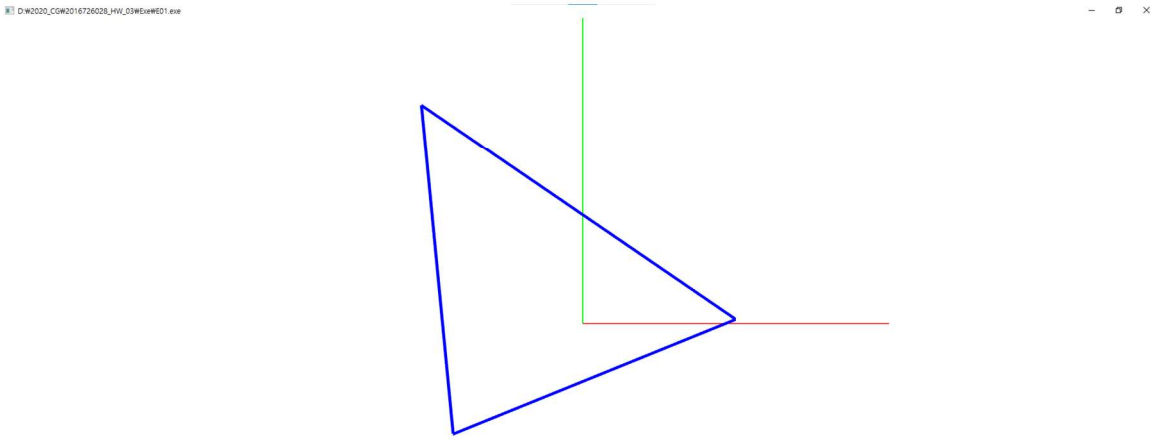
IsMouseMove 는 전역변수인 트리거로써 MouseMove 이벤트가 발생하면 true 값으로 바뀐다. 마우스 이동 이벤트가 일어나기 전에 점선이 랜더링되면, 초기값(0,0)이나 이전 사이클에서의 좌표값이 일시적으로 보이는 현상 때문에, 트리거를 둠으로써 이동 이후에 랜더링이 발생하도록 제한하였다.

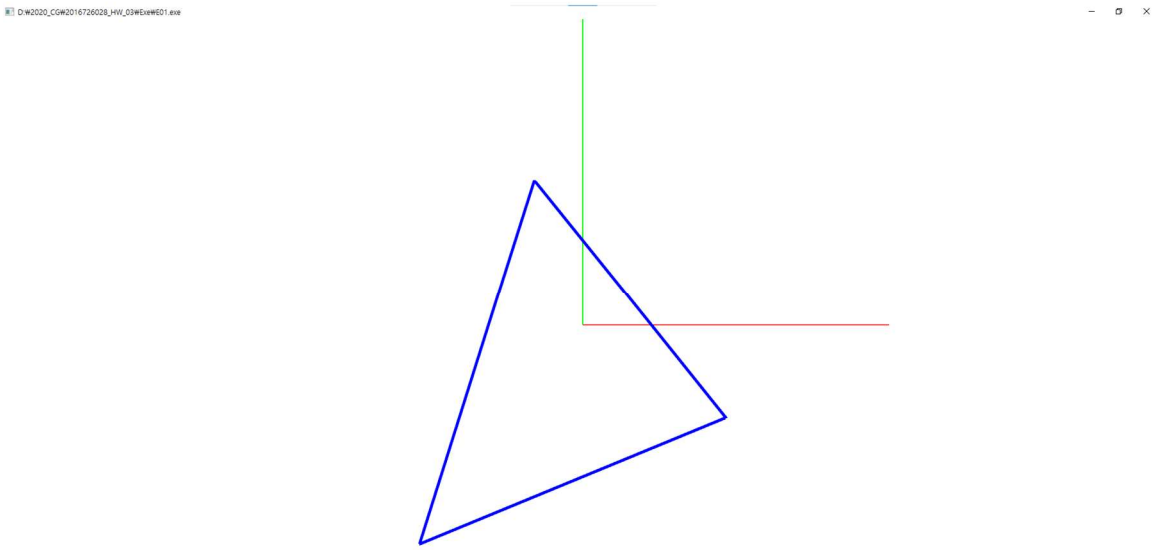
마우스 무브 함수에서 받아온 버텍스를 끝점으로 GL_LINES 를 GL_LINE_STIPPLE 을 enable 하여 선분을 출력하여준다.

E03 (Rotate the triangle)

<SNAPSHOT>







<EXPLANATION>

점이 하나 추가된 것 뿐이므로 기존 예제에서 rotate 시킬 점을 하나 추가하였다.

```
if (!pause && inputMode == InputMode::COMPLETE)
                        {
                                //One rotation per period
                                float delta_theta = float(2.0 * M_PI) / period * elapsed;

                                rotate(point[0], delta_theta);
                                rotate(point[1], delta_theta);
                                rotate(point[2], delta_theta);
```

(point[2])