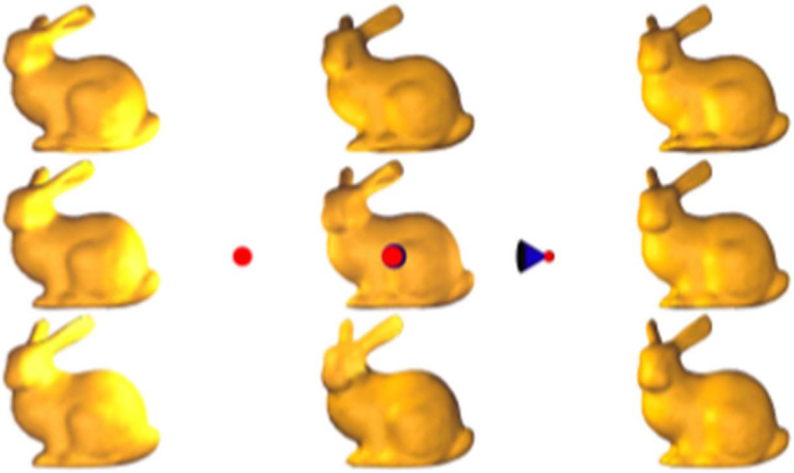
Computer Graphics



Picking

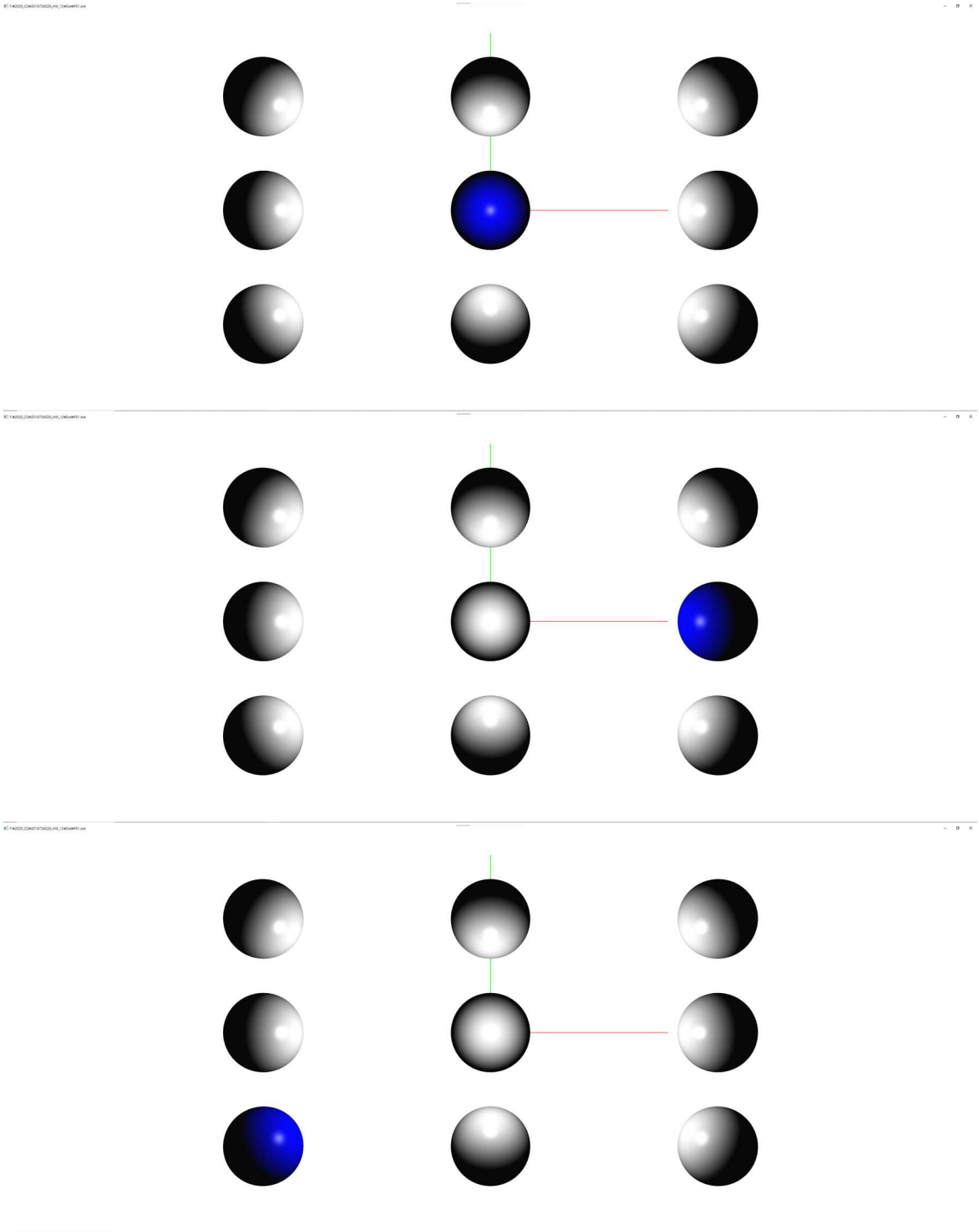
HW12

이민재 | Computer Graphics [심화전공실습 1] | 2020/11/10

	P01	E01	E02	Total
SCORE	1	1	1	3

P01 (Picking an object using selection mode)

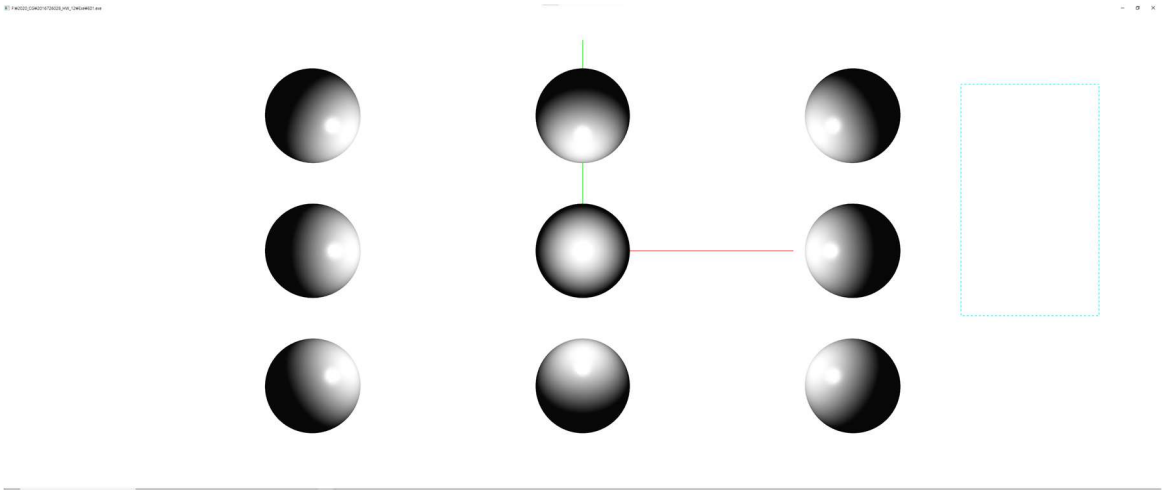
<SNAPSHOT>

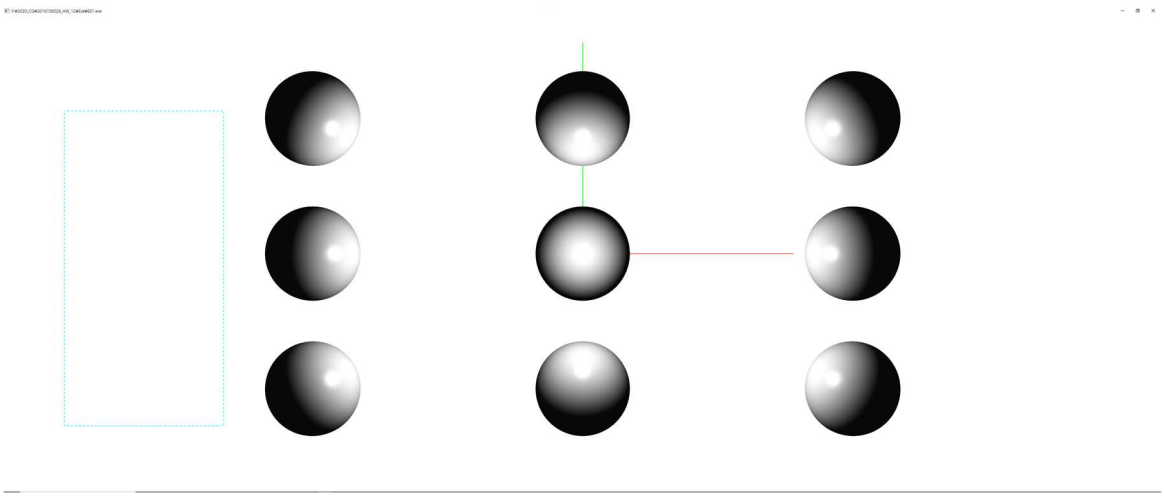


E01 (Implement click-and-drag using stippling)

<SNAPSHOT>

(하늘색 점선 사각형이라 잘 안보임)





<EXPLANATION>

화면에 점선으로 이루어진 사각형을 마우스 좌표와 연동하여 출력하기 위해, 먼저 클릭과 드래그, 릴리스 의 3 단계로 마우스 조작을 나누고 콜백함수를 설정하였다.

클릭된 순간의 좌표가 사각형의 시작점이 되고, 드래그시의 마우스 좌표가 사각형의 대각선 반대편의 좌표가 된다. 최종적으로, 출력 window 와 사각형의 출력 평면의 비가 일치하도록 z = 7.5 인 평면에 출력하기 위해서 다음과 같은 함수를 사용하였다.

```
void unProject(double xCursor, double yCursor, double* wx, double* wy, double* wz)
{
```

```
if (gluUnProject(winX, winY, zCursor, modelView, projection, viewPort, wx, wy, wz) == GLU_FALSE) {
    cout << "failed" << endl;
}
```

unProject 함수를 거친 좌표들은 point 구조체에 저장되어 drawBox 함수로 넘겨진다.

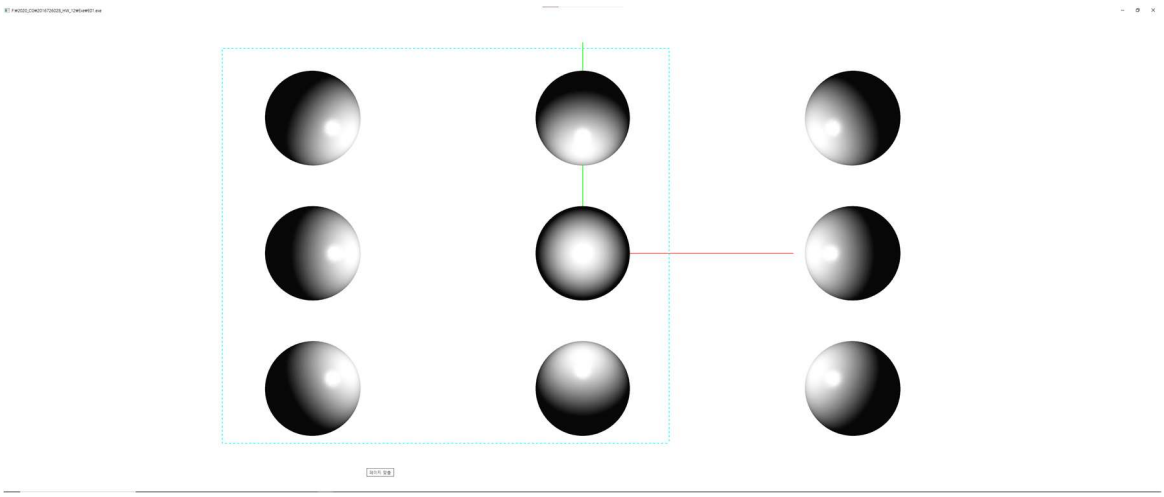
```
        unProject(BRx, BRy, &wx, &wy, &wz);

        point[1][0] = wx;
        point[1][1] = wy;

    }
}
```

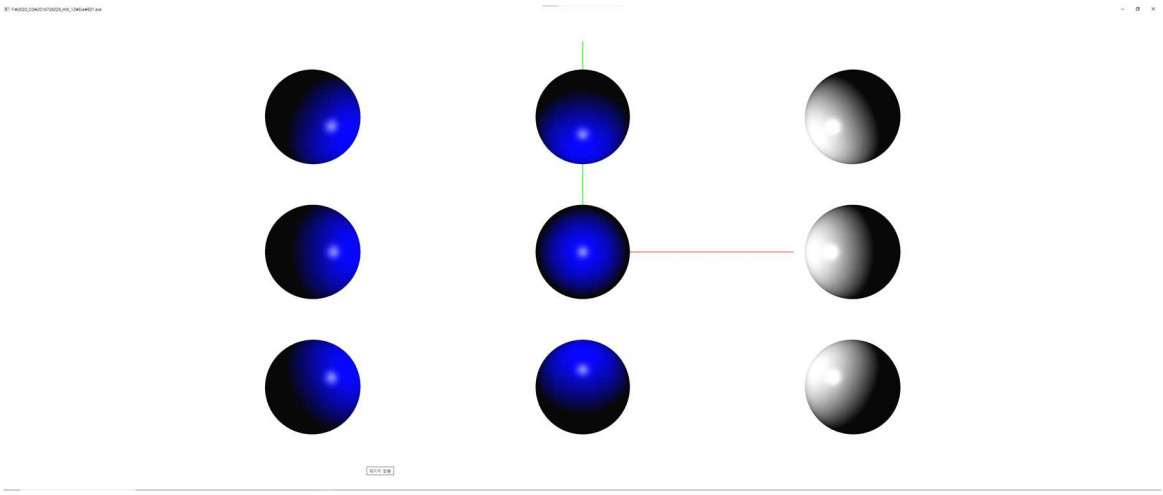
winX 와 winY 값은 getCursorPos 와 마우스 이동 콜백함수 리턴값로 넘겨받은 좌표값을 dpiScaling 해준 값이고, zCursor 의 경우 nearplane 일때를 상정하여 0.0f 로 설정하였다. modelView, projection 은 opengl 의 glGetDoublev 함수를 통해서, viewport 는 setObject 함수와 동일하게 glGetIntegerv 를 사용하여 인자로 넘겨주었다. object 좌표계의 값이 wx, wy, wz 로 저장되어, 이 값들을 render 함수부에서 gl_Quads 와 stipple 옵션을 주어 사각형을 출력하였다.

```
if (inputMode == InputMode::CLICK && IsMouseMove == true)
{
    glDisable(GL_LIGHTING);
    drawBox(AXIS_LENGTH, AXIS_LINE_WIDTH * dpiScaling);
}
```

E02 (Select all objects inside a click-and-drag rectangle)

<SNAPSHOT>





동영상 촬영에서는 뒤 axes 를 제거하고 진행하였다. 뒤의 축 또한 object 로 판정되어 hit 의개수에 포함되기 때문에 9 개로 설정한 kname 의 사이즈를 벗어나는 오류가 생긴다.

<EXPLANATION>

범위 내의 구들을 선택하기위해, 기존의 selectObject 함수의 delX 와 delY 를 사각형 의 가로,세로 값으로 설정하였다. 또한 전달되는 x,y 값을 사각형의 중심으로 설정하기 위해,

스케일링된 각 좌표들을 더하고 반으로 나누어 인자로 전달하였다.

```
inputMode = InputMode::NONE;
selectObject(window, (TLx + BRx) / 2.0f, (TLy + BRy) / 2.0f);
/*for(int i=0; i<9; i++)
    cout << "names : " << names[i] << endl;*/
IsMouseMove = false;
```

```
        delX = abs(BRx - TLx);
        delY = abs(BRy - TLy);
```

selectBuffer 에서 여러 개의 해당되는 요소들을 추출하기 위해서 selectObject 와 findNearestHits 함수를 리턴값이 없는 void 형으로 바꾸고 findNearestHits 함수를 다음과같이 변경하였다. (용도에 따라 이름도 변경하는 것이 맞으나 시간관계상 생략하였다.)

```
if (diagnosis)
{
    cout << "₩t# of names = " << n << endl;
    cout << "₩tz1 = " << z1 << endl;
    cout << "₩tz2 = " << z2 << endl;
    cout << "₩tnames: ";
    for (int j = 0; j < n; j++)
    {
        cout << selectBuffer[index + 3 + j] << " ";
        kname[i] = selectBuffer[index + 3 + j];
    }

    cout << endl;
```

사각형 안에 hit 된 모든 구들의 인덱스를 추출하여 최종적으로는 render 함수부에서 해당인덱스가 추출된 적이 있으면 파란색으로 칠하는 과정을 거쳤다.

```
if (kname[0] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));
if (kname[1] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));
if (kname[2] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));
if (kname[3] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));
if (kname[4] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));
if (kname[5] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));
if (kname[6] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));
if (kname[7] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));
if (kname[8] == (3 + i + j)) setupColoredMaterial(vec3(0, 0, 1));

if(kname[0] != (3 + i + j) && kname[1] != (3+i+j) && kname[2] != (3 + i + j) && kname[3] !=
//else setupColoredMaterial(vec3(1, 1, 1));
```