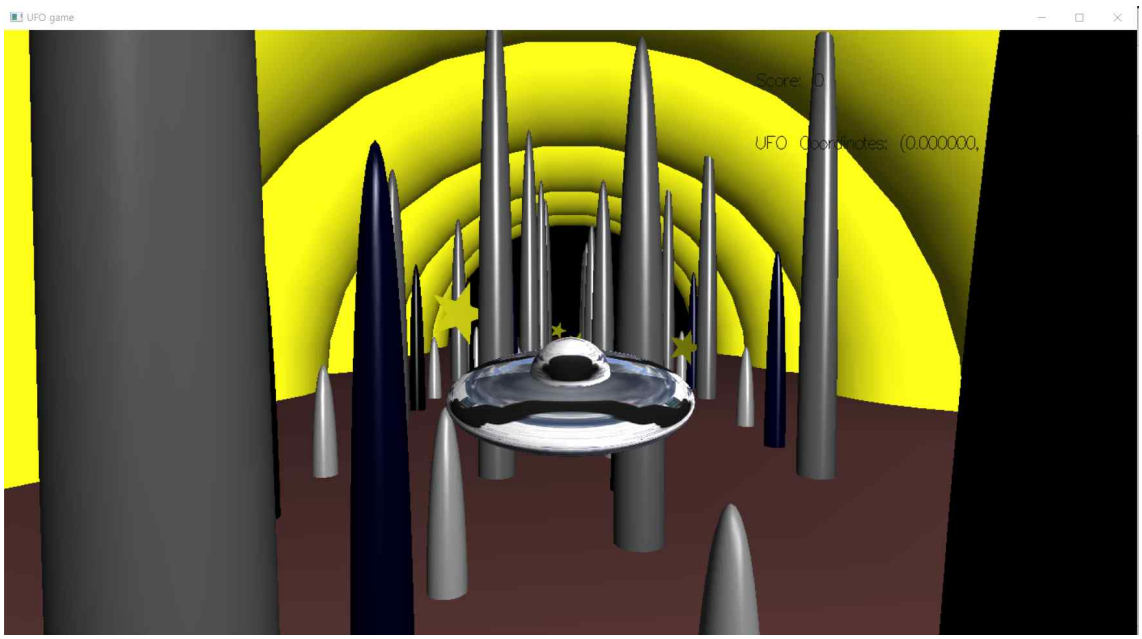
3D Video Game 최종보고서

201902703 설윤환

1. 전체 모델 구현 결과



2. 전체 프로그램 구조

키 입력 함수와 UFO, 장애물, 별모양 함수의 모델링 함수는 둘째주 보고서에 작성한 내용 그대로 만들었다.

draw함수

```
void draw() {
    /// UFO
    if (incX > 25) { //기울기가 일정수준을 넘으면 다시 돌려줌
        incX = 25;
    }
    if (incX < -25) {
        incX = -25;
    }
    if (incZ > 25) {
        incZ = 25;
    }
    if (incZ < -25) {
        incZ = -25;
    }
    glPushMatrix(); //UFO 그리기
    glTranslated(0, 1, 0);
    glRotated(90, 0, 1, 0);
    glRotated(5, 0, 0, 1);
    glRotated(incX, 1, 0, 0);
    glRotated(incY, 0, 1, 0);
    glRotated(incZ, 0, 0, 1);
    glScaled(0.45, 0.45, 0.45);
    UFO();
    glPopMatrix();

    if (uX >= 10)uX = 10; //일정공간을 못넘어가도록 함
    if (uX <= -10)uX = -10;
    if (uY > 0.1)uY = 0.1;
    if (uY < -15)uY = -15;
```

draw함수에 UFO의 기울기 및 상하좌우 이동에 제한을 거는 코드를 추가하였다.
기울기가 일정 기울기가 넘으면 넘지 않도록 하였고 x좌표와 y좌표도 제한을 넘지 못하게 하였다.

```
uZ += speed; //UFO가 아닌 배경이 뒤로간다
for (int i = 0; i < 6; i++)
{
    Z[i] += speed;
}
if (uZ >= 20)uZ = -110; //일정 범위를 넘으면 반복해준다
for (int i = 0; i < 6; i++)
{
    if (Z[i] >= 20) {
        Z[i] = -110;
    }
}
if (incX > 0)incX -= incBackFrac; //기울기를 자동으로 돌려준다
if (incX < 0)incX += incBackFrac;
if (incY > 0)incY -= incBackFrac;
if (incY < 0)incY += incBackFrac;
if (incZ > 0)incZ -= incBackFrac;
if (incZ < 0)incZ += incBackFrac;

speed += 0.0002;
if (speed >= 1)speed = 1;
// 별 충돌탐지, 점수 올리기
if (checkCollision()) {
    if (temp - uZ > 5 || uZ -temp >5) {
        score++;
        temp = uZ;
    }
}
```

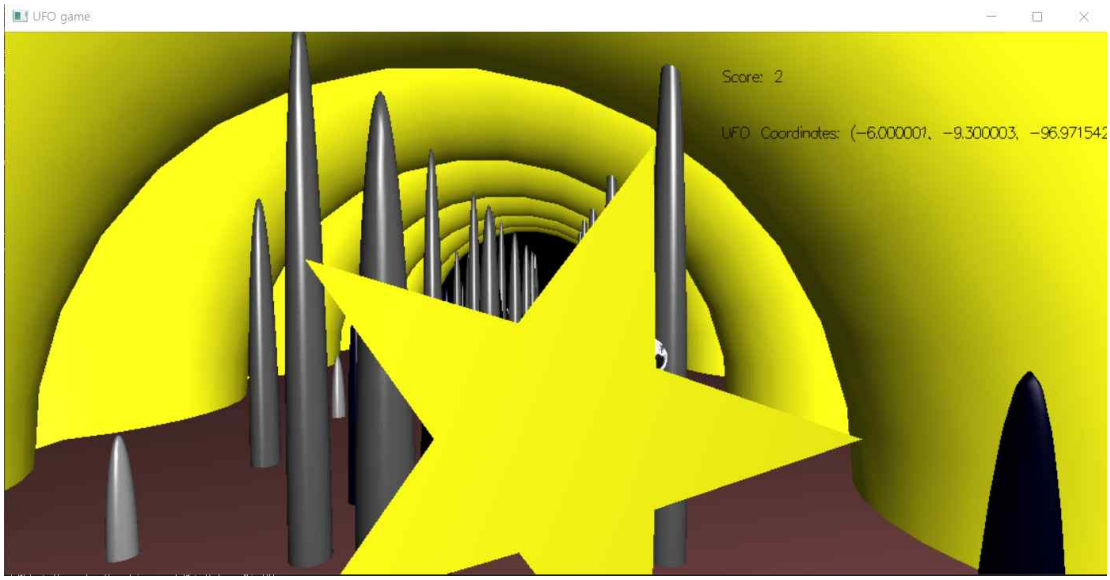
UFO를 이동하려 하였는데 시점이동에 어려움이 있어 그것보다 배경을 뒤로하는게 쉬울것같아 배경을 뒤로 가게 하였다. 배경은 speed만큼 뒤로간다. 배경을 반복하였다.
속도는 시간이 지날수록 점점 증가하도록 하여 난이도가 올라가도록 하였다.
1이 넘으면 너무 빨라지지 않게 1로 고정하였다.
checkCollision함수로 별 충돌을 감지하고 별에 닿으면 점수가 오르도록 설정하였다.

```
bool checkCollision() { //아이템 충돌 감지 함수
    for (int i = 0; i < 7; i++)
    {
        float distance = sqrt(pow(uX + itemPosX[i], 2) + pow(uY + itemPosY[i], 2)) + pow(uZ - starZ[i], 2); //아이템과 UFO의 거리 계산
        if (distance < 1.3) { //거리가 조건보다 작으면 충돌 탐지
            return true;
        }
    }

    return false;
}
```

checkCollision함수는 아이템의 절대좌표를 배열에 저장한 후 배열만큼 반복해서

UFO의 좌표와 비교해 거리를 계산해 1.3보다 작으면 별을 먹었다고 가정하고 true를 반환한다.



별을 지나가면 score가 증가하는 것을 볼 수 있다.

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

    glLoadIdentity();
    gluLookAt(0.0, 4.5, 10.0,
        0, 4, 0,
        0, 1.0f, 0.0f);

    if (start) { //시작하면
        glPushMatrix();
        glTranslated(0, 0, 0);
        glScaled(3, 3, 3);
        draw();
        glPopMatrix();
        string t = to_string(score);
        drawStrokeText(("Score: " + t).c_str(), 3, 0, 0); //점수 표시
        string ufoCoordText = "UFO Coordinates: (" + to_string(uX) + ", " + to_string(uY) + ", " + to_string(uZ) + ")";
        drawStrokeText(ufoCoordText.c_str(), 3, -1, 0); //UFO 좌표 표시
    }
    else { //메뉴화면
        glClearColor(0, 0, 0, 1);
        titleText("The UFO game", -1, -1, 0); //제목
        startText("let's start with 'q'", -1, -1, 0); //q를 누르면 시작
    }
    glColor3d(0, 0, 0);

    glutSwapBuffers();
}
```

display함수는 gluLookAt으로 시점을 설정하였다.
start가 true이면 게임이 시작된 것이다.
start가 true일 때 draw함수를 호출하고 화면에 점수, UFO의 좌표를 표시하였다.
start가 false일 때에는 메뉴화면으로 제목과 안내글자가 출력되도록 하였다.

```
const GLfloat l_ambient[] = { 0, 0, 0, 1.0 }; //빛광원
const GLfloat l_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
const GLfloat l_specular[] = { 1, 1, 1, 1.0 };
const GLfloat l_position[] = { 5, 5, 5, 0.2 };

const GLfloat m_ambient[] = { 0.6, 0.6, 0.6, 1.0 }; //물체표면
const GLfloat m_diffuse[] = { 0.8, 0.8, 0.8, 1.0 };
const GLfloat m_specular[] = { 1, 1, 1, 1.0 };
const GLfloat shininess[] = { 100.0 };
```

빛과 물체의 빛속성은 다음과 같이 설정하였다.

```
int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(1400, 750);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);

    glutCreateWindow("UFO game");

    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutKeyboardFunc(key);
    glutIdleFunc(idle);

    glClearColor(0.8, 0.8, 1, 1);
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);

    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);

    glEnable(GL_LIGHT0);
    glEnable(GL_NORMALIZE);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING);

    glLightfv(GL_LIGHT0, GL_AMBIENT, l_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, l_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, l_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, l_position);

    glMaterialfv(GL_FRONT, GL_AMBIENT, m_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, m_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, m_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, shininess);

    glutMainLoop( );

    return 0;
}
```

마지막으로 main을 만들어 정상적으로 실행할 수 있었다.

3. 후기

이때까지 배운 내용을 바탕으로 게임을 만들 수 있다는 사실이 흥미로웠고 처음에는 막막했는데 하나씩 해결해가며 기능을 추가해가는게 재밌고 이때까지 배운 내용을 정리하는 느낌이라 좋았다.

장애물에 부딪히면 게임이 종료되도록 하는 기능도 추가하고 싶었는데 시간이 부족해 기능을 넣지 못했다. 별을 먹으면 점수가 오르도록 하는 기능과 동일한 로직으로 하되 y축 범위를 늘리면 될 것 같아 텀프가 끝난 후에라도 추가해볼 예정이다.

그래픽스라는 분야가 내가 만든 코드를 통해 결과를 직관적으로 직접 볼 수 있다는 점이 흥미로운 것 같아 앞으로도 더 공부해보고 싶다.