**기초 컴퓨터 그래픽스**

**HW3 README**

20191571 김세영

**1. [환경 명세]**

1) 본인 프로그램의 실제 구동 환경을 명시 할 것 (OS, CPU, GPU, Compiler 등등)

window10 64bit, i5-10400F, gtx 1660 SUPER,

visual studio 2019 – debug win32

**2. [요구사항]**

   1. Modeling Transformation

1) 먼저 가상의 3차원 세상의 바닥과 좌표의 기준이 되는 세상 좌표계를 그려라 (최대 10점)

- 확인 방법:

프로그램 실행 시 확인할 수 있다. 가상의 3차원 세상의 바닥은 background와 색이 구분되어 있으며, 원점을 기준으로 각 좌표축의 방향을 확인할 수 있다. 

2) 최대 5개까지의 서로 다른 정적인 물체를 서로 다른 모델링 변환을 사용하여 가상의 세상에 배치하라 (물체 당 5점 최대 25점)

a) 사용한 물체: ironman

- 부여한 **서로 다른** 모델링 변환: 크기를 0.5배로 축소한 후 이동, 회전하였다.

버스의 위에 위치할 수 있도록 하였다.

- 확인 방법: 프로그램 실행시 (1번카메라 기준) 화면의 왼쪽에서 확인할 수 있다.

b) 사용한 물체: tank

- 부여한 **서로 다른** 모델링 변환: 축소 후 평면과 평행하도록 회전한 후 y축 방향으로 70도 회전한 후 이동하였다. bike의 뒤에 있도록 배치하였다.

- 확인 방법: 프로그램 실행시 (1번카메라 기준) 화면의 오른쪽에서 확인할 수 있다.

c) 사용한 물체: bike

- 부여한 **서로 다른** 모델링 변환: 축소 후 bike의 앞쪽이 떠있는 것을 의도하기 위해 y축 방향으로 회전 후 z축 방향으로 회전 후 이동하였다.

- 확인 방법: 프로그램 실행시 (1번카메라 기준) 화면의 중앙에서 확인할 수 있다.

d) 사용한 물체: bus

- 부여한 **서로 다른** 모델링 변환: 축소 후 y축방향으로 시계방향 90도 회전 후 이동, 회전하였다. ironman의 모델링 변환과 비슷하지만 이동할 때의 z좌표가 다르기 때문에 ironman의 아래에 위치한다.

- 확인 방법: 프로그램 실행시 (1번카메라 기준) 화면의 왼쪽에서 확인할 수 있다.

e) 사용한 물체: optimus

- 부여한 **서로 다른** 모델링 변환: 축소 후 회전 변환 2번 후 이동하였다. y좌표가 0보다 크게 하여 화면에 떠있도록 하였다.

- 확인 방법: 프로그램 실행시 (1번카메라 기준) 화면의 오른쪽에서 확인할 수 있다.

3) 최대 4개까지의 서로 다른 동적인 물체를 가상의 세상에 배치하라 (물체 당 10점 최대 40점). 각 동적 물체는 이동 변환, 크기 변환, 그리고 회전 변환 등의 기본 기하 변환 중 최소한 두 개 이상을 사용하여 서로 다른 움직임을 표현해야 하며, 각 동적 물체는 키보드 또는 마우스 동작을 통하여 움직임과 멈춤을 조절할 수가 있었야 한다 (자신이 선택한 최대 네 개의 동적인 물체에 대해 이 기능이 구현이 안되어 있으면 물체 당 4점 감점). 요구 사항은 아니나 동적인 물체들 중 최소한 1개의 물체에 대해서는 뉴턴의 운동의 법칙과 같이 물리적으로 충실한 방법을 사용하여 움직임을 표현해볼 것.

a) 사용한 물체: tiger

- 부여한 **서로 다른** 모델링 변환: 축소 후 회전하여 화면의 바닥 위에 배치하였고, 바닥의 가장자리 부분을 따라 걷는다. 이동한 이후 다음 모서리를 이동하기 전 90도씩 회전한다.

멈춤/이동은 키보드 ‘t’를 이용하여 조절할 수 있다. default는 움직이는 것이다.

- 확인 방법: 바닥의 모서리 부분에서 확인할 수 있다.

b) 사용한 물체: spider

- 부여한 **서로 다른** 모델링 변환: 바닥 위에 배치한 다음 바닥의 중앙을 원점으로 하고 5.0f를 반지름으로 하는 원을 따라 시계방향으로 걷는다.

멈춤/이동은 키보드 ‘s’를 이용하여 조절할 수 있다. default는 움직이는 것이다

- 확인 방법: 화면의 바닥에서 확인할 수 있다.

c) 사용한 물체: ben

- 부여한 **서로 다른** 모델링 변환: 원점에서 바닥의 한 꼭짓점까지 z좌표가 증가하며 걷고 다시 원점으로 돌아오기를 반복한다. 꼭짓점으로 이동할수록 크기가 작아진다.

멈춤/이동은 키보드 ‘b’를 이용하여 조절할 수 있다. default는 움직이는 것이다

- 확인 방법: 1번 카메라 기준 화면의 오른쪽에서 확인할 수 있다.

d) 사용한 물체: wolf

- 부여한 **서로 다른** 모델링 변환: 1.5배 확대한 후 바닥의 대각선 방향으로 포물선 운동을 한다.

포물선 운동의 각도는 45도로 하여 cos와 sin이 같고, 시간과 높이를 적절하게 조절하기 위해 중력가속도 g=0.0024로 설정하였다.

멈춤/이동은 키보드 ‘w’를 이용하여 조절할 수 있다. default는 움직이는 것이다

- 확인 방법: 1번 카메라 기준 화면의 중앙에서 확인할 수 있다.

2. Viewing Transformation

1) 1번부터 4번까지의 카메라는 CCTV 카메라와 같이 주어진 위치에 고정하여 세상을 바라보는 카메라이다. 적절한 사용자 인터페이스 동작을 통하여 원하는 카메라에서 세상을 바라볼 수 있도록 하라

a) 1번 카메라 확인 방법: 키보드 ‘1’을 누른다

b) 2번 카메라 확인 방법: 키보드 ‘2’를 누른다

c) 3번 카메라 확인 방법: 키보드 ‘3’을 누른다

d) 4번 카메라 확인 방법: 키보드 ‘4’를 누른다

2) 5번 카메라는 동적인 카메라로서 사용자 인터페이스 동작을 통하여 다음과 같이 움직일 수 있 도록 하라

a) 5번 카메라 translation 확인 방법: 키보드로 p를 누르면 translation이 활성화 되고 rotate 기능이 꺼진다. 이때 마우스를 클릭하면서 위아래로 움직이면 카메라가 translate된다.

키보드로 각각 x,y,z를 누르면 해당 축을 기준으로 translate되도록 변경할 수 있다.

b) 5번 카메라 rotation 확인 방법: q를 누르면 rotate가 활성화 되고 translation이 꺼진다. 때 마우스를 클릭하면서 좌우로 움직이면 카메라가 rotate된다.

키보드로 각각 x,y,z를 누르면 해당 축을 기준으로 rotate되도록 변경할 수 있다.

3) (추가) 1번 카메라에 대하여 고정된 위치를 중심으로 시선의 방향을 바꿀 수 있도록 하라. 어떠한 방식으로 구현할 지는 본인이 결정할 것.

a) 추가 구현 확인 방법: 1번 카메라인 상태에서 ( 1번 카메라가 아니라면 키보드

‘1’을 눌러 변경) 키보드 ‘6’ ‘7’ ‘8’ ‘9’ ‘0’ 을 눌러서 확인할 수 있다.

‘6’- ironman을 바라봄

‘7’-optimus를 바라봄

‘8’-spider를 바라봄. spider는 움직이는 물체이기 때문에 키보드를 누를 당시의 spider의 위치를 기준으로 함.

‘9’- wolf를 바라봄. wolf는 움직이는 물체이기 때문에 키보드를 누를 당시의 wolf의 위치를 기준으로 함.

‘0’-원점을 바라봄

3. Projection Transformation

1) 적절한 사용자 인터페이스 동작을 통하여 5번 카메라에 대하여 줌 인/줌 아웃 기능을 구현하라. 이때, 최대로 줌 인/줌 아웃할 수 있도록 적절히 범위를 설정하라.

- 확인 방법: 5번 카메라인 상태에서 키보드 왼/오른쪽 버튼을 누르면 zoom in/zoom out 된다.