**기초 컴퓨터 그래픽스**

**HW3 README**

학번 이름

**1. [환경 명세]**

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

- ex 1) window10 64bit, i9-10980xe, rtx 3090, visual studio 2019 – win 64, release

- ex 2) window10 64bit (mac-vm), i5-8210Y, intel UHD graphic 617,

visual studio 2019 – win 64, release

window10 – 64bit intel CPU 3825U

visual studio 2019 – win32, Debug

**2. [요구사항]**

   1. Modeling Transformation

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

- 확인 방법:

처음 프로그램이 실행되면 중간에 작게 세상 좌표계가 그려질 것이다.

바닥은 그려진 거미가 움직이는 거미줄 형태로 구현하였다.

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

a) 사용한 물체 아이언맨

- 부여한 **서로 다른** 모델링 변환: translation으로 원점에서 이동시킨 후에, scale로 크기를 조정해서 정적으로 만들어준다.

- 확인 방법: 화면이 실행되면 나온다.

b) 사용한 물체: 드래곤

- 부여한 **서로 다른** 모델링 변환: 드래곤을 그려논 거미줄과 붙어있게 이동시킨 후, 크기가 크므로 조정하여 정적으로 나타내었다.

- 확인 방법: 화면이 처음 실행되면 나온다.

c) 사용한 물체: 소

- 부여한 **서로 다른** 모델링 변환: 소를 좌표계와 함께 나타내며 시간에 따라 색 이 변하게 만들었고, translate과 scale을 이용하여 나타내었다.

- 확인 방법: 화면이 처음 실행되면 나오며, 색은 시간에 따라 자연스럽게 바뀐다.

d) 사용한 물체: 박스

- 부여한 **서로 다른** 모델링 변환: 화면의 중앙에 박스를 구현하였고, B버튼을 누르면 박스가 좌우로 움직이게 하였다. B버튼을 누르면 진동운동을 하며 속도를 늦추어서 음수가 되면 물체가 뒤로 움직이기 시작하고, 이러다가 보면 출발하는 부분을 지난다. 이 때부터 반대방향으로 가속하기 시작한다. 이러면 같은 궤도를 반복하며 가속운동을 하는 물체를 얻을 수가 있다.

- 확인 방법: 화면이 처음 실행되면 박스가 있으며 이때, B를 누르면 박스가 좌 우로 움직인다.

e) 사용한 물체: 거미줄

- 부여한 **서로 다른** 모델링 변환: 거미가 움직이는 거미줄 형태의 큰 선들을 구현하였다. 12개의 점만을 사용해 만들며, 이 점은 반복하며 멀어질 수록 더 크게 그리도록 하며 거미줄을 만듭니다. 1.05배씩 바퀴가 커진다.

- 확인 방법: 화면이 처음 실행되면, 움직이는 거미 밑에 거미줄이 구현되어있다.

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: 거미

- 부여한 **서로 다른** 모델링 변환: 거미가 시간이 지남에 따라 프레임을 달리하며, 함수에서 프레임을 받아와서 돈다. 이 거미는 한 바퀴를 돌고 나서는 반대 방향 으로 돕니다. 이때 거미는 현재 돌고있는 각도를 저장하여 이것이 360도가 넘으 면, 반대 방향으로 돕니다. 이때 거미는 항상 중심을 오른쪽에 두고 돌기에 현재 돌고 있는 각도 만큼 거미 자체를 돌린 후, 원의 좌표에 맞게 위치시킨다.

- 확인 방법: 처음 화면이 실행되면 거미가 거미줄 위를 돌아다닌다.

b) 사용한 물체: 자동차

- 부여한 **서로 다른** 모델링 변환: 자동차는 원 궤도를 따라 위성운동을 하며 자 동차에 또한 바퀴도 구현하여 사이클로이드를 사용하여 원을 굴린다. 바퀴의 방 향전환을 자연스럽기 위해 rotate로 자연스럽게 굴립니다. 만든 자동차와 그 바 퀴, 못은 각각의 위치와 회전한 각도를 계층적으로 가지며 거리, 진행방향을 고 려하여 자동차와 바퀴의 상태를 조절한다. 각각의 바퀴는 몸체에, 못은 바퀴에 소속되어 움직인다.

- 확인 방법: 자동차는 처음 프로그램이 실행되면 자동으로 특정 궤도를 따라

운동합니다.

c) 사용한 물체: 호랑이

- 부여한 **서로 다른** 모델링 변환: 호랑이는 제자리에서 움직이며, C버튼을 누르 면 위아래로 호랑이가 점프를 한다. Rotate와 scale, translate를 통해 WC에 배치 되어있고, C를 누를 경우에 flag가 1로 설정되며, 원래 1이었다면 0으로 설 정된다. 이러한 flag가 1로 설정되어 있으면 jump함수를 호출하여 속도를 프레임 별로 줄이고, 높이가 0이되면 속도를 초기화시키는 탄성운동을 한다.

- 확인 방법: 처음 프로그램이 실행이 되면, 호랑이가 제자리에서 걸어다니고, C 버튼을 누르면 호랑이가 위아래로 점프를 하고, 다시 버튼을 누르면 호랑이가 그 자리에서 멈춘다.

d) 사용한 물체: 주전자

- 부여한 **서로 다른** 모델링 변환: 주전자는 버튼을 누르면 원점을 기준으로 특정 거리를 두고 원 궤도 운동을 하며, 주전자를 원래 각도로 유지하면서 회전시키 기 위하여 translate을 하고 rotate를 사용하였습니다.

- 확인 방법: 처음 프로그램이 실행이 되면, 주전자가 가만히 있다가, P버튼을 누 르면, 색이 변화하면서 원점을 중심으로 원 궤도 운동을 합니다.

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: world 화면 일때, 아랫쪽 화살표를 누르면 밑에서 물체들을 바라본다.

b) 2번 카메라 확인 방법: R버튼을 누르면, 자동차 안에 있는 사람을 기준으로 디스플레이 모드를 전환한다.

c) 3번 카메라 확인 방법: G버튼을 누르면, 호랑이 기준으로 디스플레이 모드를 전환한다.

d) 4번 카메라 확인 방법:world 화면 일때, 위쪽 화살표를 누르면 위에서 물체들을 바라본다.

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

a) 5번 카메라 translation 확인 방법:

world 화면일 경우(w) 및 다른 화면일 경우 w버튼을 눌러 world화면에 온 뒤에 상 하키를 누르면 z축을 기준으로 +,- 방향으로 볼 수가 있으나, 이 때에는 좌우키를 사용하여 움직였던 각도가 0도로 초기화됩니다.

b) 5번 카메라 rotation 확인 방법:

world 화면일 경우(w) 및 다른 화면일 경우 w버튼을 눌러 world화면에 온 뒤에 좌 우키를 누르면 해당 방향으로 30도씩 회전한 극좌표로 카메라의 위치를 옮기며 바라보는 방향은 원점으로 같습니다.

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:

world 화면 상태에서 D버튼을 누르면 자동차의 시점을 화면 오른쪽 하단에 추가해주고,

T버튼을 누르면 화면 오른쪽 하단에 나타내어준다.

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

World 화면인 경우나 모든 카메라에서 Shift를 누르고 드래그하면 좌우로 움직임에 따라 줌인/줌아웃이 가능합니다. (다른 관점의 경우에도 적용이 가능합니다.)

추가적인 버튼 설명

(다 소문자임)

w : worldview 상태

p : teapot 움직임(원점을 중심으로 돈다.)

b : box 움직임( 가로로 왔다갔다 운동)

c : tiger 움직임(위아래로 점프)

r : Driver 시점 디스플레이

g : tiger 시점 디스플레이

f : 물체들 칸 색으로 칠함

l : 물체들 칸 다시 비게 함

d : 차 기준 뷰포트를 화면 오른쪽 하단에 추가

t : 호랑이 기준 뷰포트를 화면 오른쪽 하단에 추가