**기초 컴퓨터 그래픽스**

**HW3 README**

20171646 박태윤

**1. [환경 명세]**

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

- window10 64bit, i5-9300H, NVIDIA GeForce GTX 1050, visual studio 2019 – win 64, release

**2. [요구사항]**

   1. Modeling Transformation

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

- 확인 방법: 물처럼 보이는 바닥과 화면 중앙(고질라)에 좌표계가 그려져 있는 것을 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: godzilla

- 부여한 **서로 다른** 모델링 변환:

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(0.0f, -300.0f, 0.0f));

ModelViewMatrix = glm::scale(ModelViewMatrix, glm::vec3(3.0f, 3.0f, 3.0f));

- 확인 방법: 화면 중앙에 godzilla를 배치하였다.

b) 사용한 물체: ironman

- 부여한 **서로 다른** 모델링 변환:

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(-400.0f, 200.0f, -400.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, -90.0f \* TO\_RADIAN, glm::vec3(-1.0f, 0.0f, 0.0f));

ModelViewMatrix = glm::scale(ModelViewMatrix, glm::vec3(30.0f, 30.0f, 30.0f));

- 확인 방법: 화면 중앙의 godzilla가 바라보고 있는 방향을 12시로 기준을 두었을 때, 5시 방향에 위치함.

c) 사용한 물체: bus

- 부여한 **서로 다른** 모델링 변환:

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(100.0f, 300.0f, -100.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, -30.0f \* TO\_RADIAN, glm::vec3(-1.0f, 0.0f, 0.0f));

ModelViewMatrix = glm::scale(ModelViewMatrix, glm::vec3(3.0f, 3.0f, 10.0f));

- 확인 방법: 화면 중앙의 godzilla가 바라보고 있는 방향을 12시로 기준을 두었을 때, 7시 방향에 위치함.

d) 사용한 물체: tank

- 부여한 **서로 다른** 모델링 변환:

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(200.0f, 50.0f, 200.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, -150.0f \* TO\_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, -90.0f \* TO\_RADIAN, glm::vec3(1.0f, 0.0f, 0.0f));

ModelViewMatrix = glm::scale(ModelViewMatrix, glm::vec3(10.0f, 10.0f, 10.0f));

- 확인 방법: 화면 중앙의 godzilla가 바라보고 있는 방향을 12시로 기준을 두었을 때, 11시 방향에 위치함.

e) 사용한 물체: bike

- 부여한 **서로 다른** 모델링 변환:

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(-200.0f, -30.0f, 200.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, -90.0f \* TO\_RADIAN, glm::vec3(0.0f, 1.0f, 1.0f));

ModelViewMatrix = glm::scale(ModelViewMatrix, glm::vec3(60.0f, 60.0f, 60.0f));

- 확인 방법: 화면 중앙의 godzilla가 바라보고 있는 방향을 12시로 기준을 두었을 때, 1시 방향에 위치함.

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: spider

- 부여한 **서로 다른** 모델링 변환:

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(-300.0f + clock\_spider, 0.0f, /\*300.0f - spider\_size - clock\_spider\*/0.0f));

if (clock\_spider > 180 && clock\_spider < 360) {

if (clock\_spider < 270) {

ModelViewMatrix = glm::translate(ModelViewMatrix, glm::vec3(1.0f, (clock\_spider - 180.0f) \* 5.0f, 1.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, 30.0f \* TO\_RADIAN, glm::vec3(0.0f, 0.0f, 1.0f));

}

else {

ModelViewMatrix = glm::translate(ModelViewMatrix, glm::vec3(1.0f, (360.0f - clock\_spider) \* 5.0f, 1.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, -30.0f \* TO\_RADIAN, glm::vec3(0.0f, 0.0f, 1.0f));

}

}

ModelViewMatrix = glm::rotate(ModelViewMatrix, 120.0f \* TO\_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));

ModelViewMatrix = glm::scale(ModelViewMatrix, glm::vec3(50.0f + spider\_size, -50.0f - spider\_size, 50.0f + spider\_size));

- 확인 방법:

고질라를 기준으로 거미가 걸어가면서 타고 올라가는 듯한 동작을 보이는 것을 확인할 수 있다. ‘s’버튼을 계속 입력해보면 동작을 멈추거나, 이동을 멈추거나, 둘 다 멈추는 것을 확인할 수 있다.

b) 사용한 물체: tiger

- 부여한 **서로 다른** 모델링 변환:

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(-1000.0f - (clock\_tiger \* tanf(clock\_tiger \* 0.04f \* TO\_RADIAN)), -30.0f, 250.0f \* sinf(clock\_tiger \* TO\_RADIAN)));

ModelViewMatrix = glm::rotate(ModelViewMatrix, (60.0f \* sinf(clock\_tiger \* TO\_RADIAN)) \* TO\_RADIAN - 90.0f \* TO\_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, (240.0f) \* TO\_RADIAN, glm::vec3(1.0f, 0.0f, 0.0f));

- 확인 방법: godzilla기준 오른쪽에서 호랑이가 헤엄을 치는 듯한 동작을 하는 것을 확인할 수 있다. ‘t’버튼을 계속 입력해보면 동작을 멈추거나, 이동을 멈추거나, 둘 다 멈추는 것을 확인할 수 있다.

c) 사용한 물체: ben

- 부여한 **서로 다른** 모델링 변환:

ModelViewMatrix = glm::rotate(ViewMatrix, (-clock\_ben-45.0f) \* TO\_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));

ModelViewMatrix = glm::translate(ModelViewMatrix, glm::vec3(200.0f+clock\_ben, clock\_ben, 200.0f+clock\_ben));

ModelViewMatrix = glm::scale(ModelViewMatrix, glm::vec3(100.0f, -100.0f, -100.0f));

- 확인 방법: 화면 중앙의 godzilla를 기준으로 점점 커지는 나선 모양으로 ben이 걸어가

는 것을 확인할 수 있다. ‘b’버튼을 계속 입력해보면 동작을 멈추거나, 이동을 멈추거나, 둘 다 멈추는 것을 확인할 수 있다.

d) 사용한 물체: wolf

- 부여한 **서로 다른** 모델링 변환:

if (rotation\_angle\_wolf < 360) {

//ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(360.0f-rotation\_angle\_wolf, 500.0f \* sinf(rotation\_angle\_wolf \* TO\_RADIAN), 0.0f));

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(1000.0f, 0.0f, 0.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, -10.0f \* rotation\_angle\_wolf \* TO\_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));

//ModelViewMatrix = glm::rotate(ModelViewMatrix, -1.0f \* rotation\_angle\_wolf \* TO\_RADIAN, glm::vec3(0.0f, 0.0f, 1.0f));

}

else {

ModelViewMatrix = glm::translate(ViewMatrix, glm::vec3(rotation\_angle\_wolf, 500.0f \* sinf(rotation\_angle\_wolf \* TO\_RADIAN), 0.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, 1.0f \* rotation\_angle\_wolf \* TO\_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, 1.0f \* rotation\_angle\_wolf \* TO\_RADIAN, glm::vec3(0.0f, 0.0f, 1.0f));

ModelViewMatrix = glm::translate(ModelViewMatrix, glm::vec3(500.0f, -250.0f, 0.0f));

}

ModelViewMatrix = glm::rotate(ModelViewMatrix, -90.0f \* TO\_RADIAN, glm::vec3(0.0f, 1.0f, 0.0f));

ModelViewMatrix = glm::rotate(ModelViewMatrix, -90.0f \* TO\_RADIAN, glm::vec3(1.0f, 0.0f, 0.0f));

ModelViewMatrix = glm::scale(ModelViewMatrix, glm::vec3((100.0f, 100.0f, 100.0f)));

- 확인 방법: wolf가 godzilla기준 왼쪽 방향에서 동작을 하고 있다. 처음에 바닥에 박힌 모습으로 제자리에서 회전을 하다가 이후 위로 솟구쳐 올랐다가 바닥으로 꺼지는 모습을 확인할 수 있다. ‘w’버튼을 계속 입력해보면 동작을 멈추거나, 이동을 멈추거나, 둘 다 멈추는 것을 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

-처음에 default로 5번 카메라로 설정이 되어 있기 때문에 ‘c’버튼을 통해 5번카메라 설정을 해체해야 합니다. 5번 카메라로 다시 보고 싶다면 ‘c’버튼을 한 번 더 누르면 됩니다. 5번 카메라 모드를 해제한 뒤 왼쪽, 오른쪽 방향키를 누르면 1~4번 카메라로 볼 수 있습니다.

a) 1번 카메라 확인 방법: 콘솔창에 ^^^ Camera Number = 1이 출력이 됐을 때의 모습이 1번 카메라로 바라보고 있는 모습입니다.

b) 2번 카메라 확인 방법: 콘솔창에 ^^^ Camera Number = 2이 출력이 됐을 때의 모습이 1번 카메라로 바라보고 있는 모습입니다.

c) 3번 카메라 확인 방법: 콘솔창에 ^^^ Camera Number = 3이 출력이 됐을 때의 모습이 1번 카메라로 바라보고 있는 모습입니다.

d) 4번 카메라 확인 방법: 콘솔창에 ^^^ Camera Number = 4이 출력이 됐을 때의 모습이 1번 카메라로 바라보고 있는 모습입니다.

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

- 각 translation과 rotation의 기준이 되는 축은 ‘x’, ‘y’, ‘z’로 설정할 수 있습니다. 예를 들어 카메라의 ‘y’을 기준으로 translation과 rotation을 하고 싶다면 5번 카메라 모드를 ‘c’버튼을 적절히 사용하여 세팅한 뒤 y버튼을 누르고 방향키를 통해 동작을 할 수 있습니다.

a) 5번 카메라 translation 확인 방법: 왼쪽, 오른쪽 방향키를 통해 translation을 할 수 있습니다.

b) 5번 카메라 rotation 확인 방법: 위, 아래 방향키를 통해 rotation을 할 수 있습니다.

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 카메라로 세팅을 한 뒤 마우스 왼쪽 버튼 클릭을 통해 시선 이동을 할 수 있습니다. 예를 들어 화면 중앙 오른쪽 위를 클릭을 하면 그 방향대로 카메라의 시선이 따라가는 것을 확인할 수 있습니다.

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 카메라로 세팅을 한 뒤 마우스 휠 동작을 통해 줌 인/줌 아웃을 할 수 있습니다.