Student ID: 2018008395

Name: 박정호

1. **Introduction to the program, a brief description of how to run it, and how to use it.**

   팩맨과 닮은 캐릭터를 조종하면서 아이템을 따라가면 아이템이 움직이는 프로그램이다. 아이템 둘의 모양은 같지만, 하나는 캐릭터를 따라오면서 가까운 거리를 유지하려고 하고, 다른 하나는 캐릭터를 따라오지는 않지만, 일정 거리 이내로 캐릭터가 가까워지면 캐릭터와 멀어지려고 하는 움직임을 가진다. 방향키로 캐릭터를 움직이며, A, S, G, H, R 키를 사용해서 main object의 transformation을 수행할 수 있다. 단, main object의 rotate는 마우스로 수행한다. 1인칭 시점과 3인칭 시점의 toggling은 마우스 좌클릭으로 바꿀 수 있다. obj 파일을 사용하지는 않기 때문에 그냥 프로그램을 실행하기만 하면 된다. 자세한 구현과, 동작 설명은 2번에서 계속하겠다.

2. **Description of the implementation of each requirement:** 1) How you implemented that requirement, 2) How TA can verify that the requirement has been implemented, by performing a program's specific feature.. **Requirement items not implemented should be left blank.**

   A. (15 pts) Draw 3 or more 3D objects.

      이 프로그램의 object는 총 3개가 있다. Packman이라고 불리는 main object와 Coin이라고 불리는 object 2개가 그것이다. 마땅한 obj 파일을 찾을 수 없었기 때문에 기존의 drawCube()를 이용해서 적절한 모양을 그렸다. Hierarchical Model에 대한 이야기는 F. Extra Credit에서 계속하겠다.
      3개의 object가 모두 그려지는지에 대한 확인은 단순히 main object를 움직이면서 확인할 수 있다. 기본적으로 coin은 (5, 0, 5), (-5, 0, 5)에 그려지고, main object는 (0, 0, 0)에 그려진다.

   B. (15 pts) One of the objects should be "main object". The user should be able to transform the main object using the mouse or keyboard.

      main object의 transformation은 gComposedM이라는 전역 변수에 쌓아진다. 이는 transformation matrix 저장하는 변수로, curser_callback이나, key_callback에서 이 전역 변수를 수정한다. 수정 규칙은 다음과 같다.

      1. Translate
         이 프로그램은 y축 방향의 이동은 불가능하게 되어 있다. 모두 땅 위에서 움직이는 설정이기 때문인데, 따라서 전후 좌우의 이동만 구현되어 있다. Main object의 z축 양수 방향을 앞쪽이며, 방향키로 조작할 수 있도록 key_callback을 사용

해서 구현했다. Homogenous transformation을 사용해서 gComposedM에 버튼에 맞는 행렬을 right multiply했다.

2. Rotate

   시중의 게임 "마인크래프트"처럼, main object가 바라보는 시각은 마우스로 조작하도록 했다. 앞에서 전후좌우로 움직이는 것과 같은 이유로, 여기서도 회전축은 y축으로 한정해서 고개를 옆으로 돌리는 회전만 허용했다. 구현은 curser_callback을 사용해서 마우스 포인터의 좌우 방향 offset에 비례하는 각도를 구하고, 이 각도를 이용해서 rotation matrix를 만들어 gComposedM에 곱했다.

   이에 대한 확인 방법은 마우스를 양 옆으로 움직이는 것인데, 마우스의 움직임에 맞게 main object도 돌아가는 것을 볼 수 있다.

3. Shear

   한쪽 방향으로 쏠리는 움직임인데, 단순히 앞뒤로 기울어지도록 구현했다. 즉 main object의 local frame의 z축방향으로 기울어지도록 했다. 여기서 기울어지는 정도는 y축 값이 클수록 더 많이 z축 방향으로 쏠리도록 했다. 구현은 translate와 마찬가지로 key_callback을 사용해서 키보드 입력으로 움직이도록 했고, 입력된 키에 따라 다른 matrix를 gCompsedM을 곱하도록 했다.

   사용 방법은 'a'키를 누르면 z축의 음수 방향으로 기울어지고, 's'를 누르면 z축의 양수 방향으로 기울어진다.

4. Scale

   object의 크기를 늘렸다 줄였다 하는 transformation인데, 본인은 x, y, z 방향으로 동시에 크기가 변하도록 했다. 즉, x축 방향 따로, y축 방향 따로, z축 방향 따로 크기가 변하지는 않는다. 구현은 translate나 shear와 마찬가지로 key_callback을 사용해서 구현했으며, 입력된 키에 따라 다른 matrix를 gComposedM에 곱하도록 했다.

   또한 여기서는 후에 다른 object의 movement를 위해서 A_size라는 전역 변수 하나를 같이 수정한다. 이는 main object의 크기를 나타내는 변수인데, 다른 오브젝트와의 거리를 계산하기 위해서 쓰인다.

   사용 방법은 다음과 같다. 'g'키를 누르면 x, y, z 방향으로 1.1배 커지게 되고, 'h' 키를 누르면 x, y, z 방향으로 0.9배 작아지게 된다. 또한 이 계산은 gComposedM 뿐만 아니라 A_size에도 적용된다.

5. Reflect

   특정 축을 기준으로 object를 뒤집는 transformation이다. 여기서는 xz 평면 기준으로만 뒤집히도록 구현되어 있고, 위의 Translate, Shear, Scale과 마찬가지로 key_callback으로 구현했다. 키가 입력이 되면 gComposedM에 reflection matrix

를 곱하게 된다. 다만, 여기서도 다른 변수의 수정이 가해지는데, camera movement를 위해 up 이라고 하는 전역 변수에 -1을 곱하게 된다. 이는 후에 up vector를 만들기 위해 쓰이는데, main object가 y축 방향으로 뒤집히게 되면 up vector도 반대 방향으로 바뀌어야 하기 때문에 이렇게 구현했다.
사용방법은 다음과 같다. 'r' 키를 누르면 main object가 xz평면을 기준으로 뒤집히게 된다. 실제로 화면에서 보이는 것은 main object이외의 모든 object가 뒤집힌 것으로 보이게 된다. 이는 카메라의 위치도 main object가 뒤집히면서 같이 뒤집히기 때문이다.

C. (15 pts) The rest of the objects should be automatically moved in response to the movement of the main object or other objects around it.

Main object를 제외하면 여기서는 두개의 object가 있다. Coin이라는 이름으로 불리는 팔각 별인데, 각각 빨간색, 파란색으로 object color가 설정되어 있다. 빨간색 Coin을 Coin1, 파란색 Coin을 Coin2라고 칭하겠다.
Coin1은 기본적으로 움직이지 않는다. 하지만 main object가 가까이 가서, 일정 거리 이내로 들어올 경우 main object와 거리를 두도록 움직인다. 일정 거리를 띄웠다면 더 이상 움직이지 않는다. 또한 Coin2와 가까워질 경우에도 거리를 두는데 이때도 일정 거리를 띄웠다면 더 이상 움직이지 않는다.
구현 방식은 다음과 같은데, 매 순간 main object의 위치와 Coin1의 위치를 가지고 둘 사이의 거리를 구한다. 여기서 위치는 두 object가 위치한 x, y, z좌표를 말한다. 여기서 거리가 1이하라면, 너무 가까운 것으로 판단하고, Coin1의 위치를 현재 두 object 사이의 벡터 반대 방향으로 옮긴다. 즉 멀어지도록 하는 것이다. 여기서 A_size가 왜 쓰이는지 나오는데, 실제로 object의 위치로 표시하는 부분은 object의 표면 좌표가 아닌, object의 중심 좌표이다. 따라서 두 object가 겹치는 것을 방지하기 위해 object의 크기까지도 같이 거리에 사용하는 것이다. Coin2와의 거리 유지도 위와 같은 방식을 사용한다.
Coin2는 기본적으로 main object를 따라온다. 하지만 Coin1과 마찬가지로 일정 거리 이내로는 들어오지 않는데, 여기서 main object가 움직여서 일정 거리 이내로 들어오게 되면 오히려 거리를 유지하기 위해 main object로부터 멀어진다.
구현 방식은 다음과 같다. 기본적으로는 Coin1과 마찬가지로 매 순간 main과의 거리를 구한다. 여기서 거리의 허용범위를 두는데 거리가 허용 범위 이내라면 움직이지 않고 멈춘다. 하지만 허용 최대 범위보다 멀 경우 두 object 사이의 벡터 방향으로 위치를 옮겨서 거리를 좁힌다. 만약 반대로 허용 최소 범위보다 가까울 경우 두 object 사이의 벡터 반대 방향으로 위치를 옮겨서 거리를 넓힌다.
이를 확인하려면 우선 빨간색 코인 쪽으로 object를 이동해보고, 빨간색 코인이 움직이는지 보면 된다. 3인칭으로 보는 것이 움직임을 확인하기 좋다. 이 동안 아마 파란색 코인은 main object를 따라왔을텐데, 이때 파란색 코인 쪽으로 main object를 움직이면 파란색 코인도 움직이는 것을 볼 수 있다.

D. (15 pts) Use perspective projection. The camera should be able to be switched between two modes:

사용 방법이 공통되는 사항이 있어 구현은 아래에서 설명하겠다. 우선 사용 방법부터 설명하자면 처음에는 Quarter View로 실행이 된다. 이후 마우스 좌클릭을 하게 되면 First Person View가 되고, 다시 누르게 되면 당연히 Quarter View로 돌아온다. 즉, 마우스 좌클릭이 Camera View 방식을 toggling하게 된다. 이는 당연히 button_callback을 사용해서 구현했다.

i. First Person View

카메라의 위치는 main object의 local frame의 z축 양수 방향 표면으로 했고, 카메라가 바라보는 위치는 여기서 z축 양수 방향으로 main_object의 크기만큼 떨어진 방향을 보도록 했다. 여기서 main object의 local frame을 구하기 위해 앞에서 구한 gComposedM을 처음 (main object의 위치 + main object의 크기)에 곱한다. 이렇게 되면 main object의 local frame 기준 z축 양수 방향 표면을 구할 수 있다. 여기서 카메라가 바라보는 위치를 구하는 것은 앞에서 (main object의 위치 + main object의 크기)에 z축 방향 벡터를 한번 더 더해주고 gComposedM을 곱해주는 것으로 구해진다. Up vector는 전역변수 up을 사용한다. 이제 이렇게 구한 카메라 위치, 바라보는 위치, up vector를 glulookat의 인자로 사용하면 된다.

ii. Quarter View

우선 카메라가 바라보는 위치는 gComposedM이 곱해진 main object의 위치로 둔다. 즉 언제나 카메라는 main object를 바라본다. 카메라의 위치는 여기서 구한 main object의 위치에 (3, 3, 3)을 더한 위치가 된다. 물론 이는 up vector가 (0, -1, 0)일 경우 -1이 곱해진다. (reflect가 되면 카메라 위치도 반전된다. Up Vector는 First Person View와 마찬가지로 up을 사용하고, 이렇게 구한 값을 모두 glulookat의 인자로 사용하면 된다.

E. (15 pts) Use OpenGL lighting / shading.

i. Use different material colors for each object.

우선 main object의 경우 흰색, 즉 (1, 1, 1, 1)을 object color로 설정했다.
Coin1의 경우 (1, 0, 0, 1), Coin2의 경우 (0, 0, 1, 1)을 object color로 두어 각각 빨간색과 파란색을 가지도록 했다.

ii. Use 2 or more light sources. Among them, at least one light should be animated.

조명 두개의 위치는 y축 방향으로 5만큼, 떨어진 다음, 반지름 5의 원을 따라서 빙글빙글 돌도록 설정했다. 여기서 조명 두개는 서로 y축 기준 대칭을 이루고 있다. 각 조명은 빨간색 빛과 초록색 빛을 낸다.

iii. You can use either flat shading or smooth shading for each object.

PyOpenGL에 default인 flat shading을 사용했다. 그래서 object를 그릴 때 일일이 vertex normal vector를 설정했다. 이에 대한 확인은 처음 프로그램을 시작하면 흰색의 main object가 색깔이 계속 바뀌는 것으로 확인할 수 있다.

F. Extra credit

i. Use an animating hierarchical model composed of multiple meshes as an object, for all objects.

우선 main object는 두개의 상자가 위아래로 벌렸다 닫았다를 반복하면서 팩맨 같은 움직임을 가진다. 여기서 조금 밋밋한 감이 있어서 위에 빙글빙글 돌아가는 플레이어 표시를 추가했다. 또한 object 전체가 시간에 따른 sin값의 절댓값만큼 y축방향으로 translate하는데 , 이로 인해 마치 통통 튀는 듯한 움직임을 만들 수 있었다.

Coin1과 Coin2는 같은 mesh를 사용해서 같은 움직임을 한다. 아래의 팔각별은 x축과 z축 기준으로 빙글빙글 돌면서 마치 코인이 옆면으로 튀는 느낌을 구현했고, main과 마찬가지로 시간에 따른 cos값의 절댓값만큼 y축 방향으로 translate하면서 통통 뛰는 움직임을 만들었다.