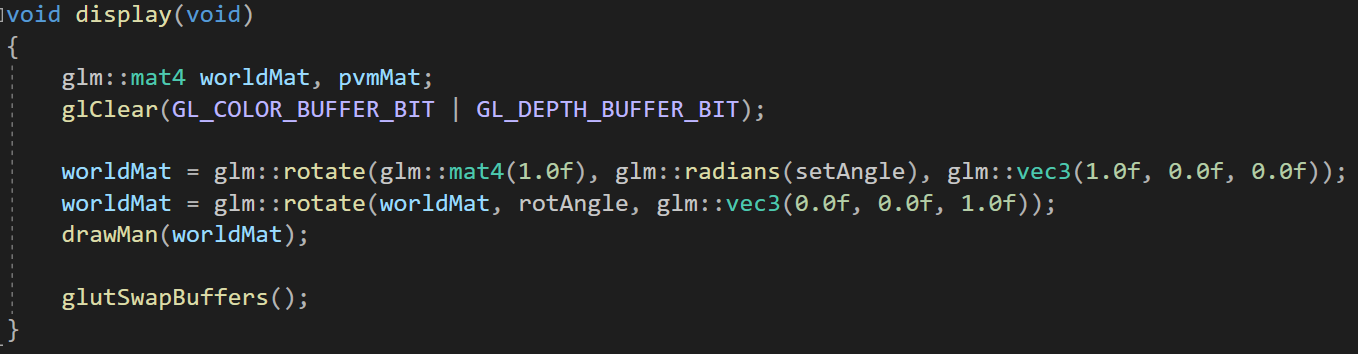
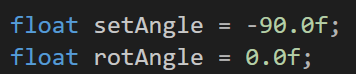
**컴퓨터 그래픽스 프로젝트 1 – Cubeman Walking**

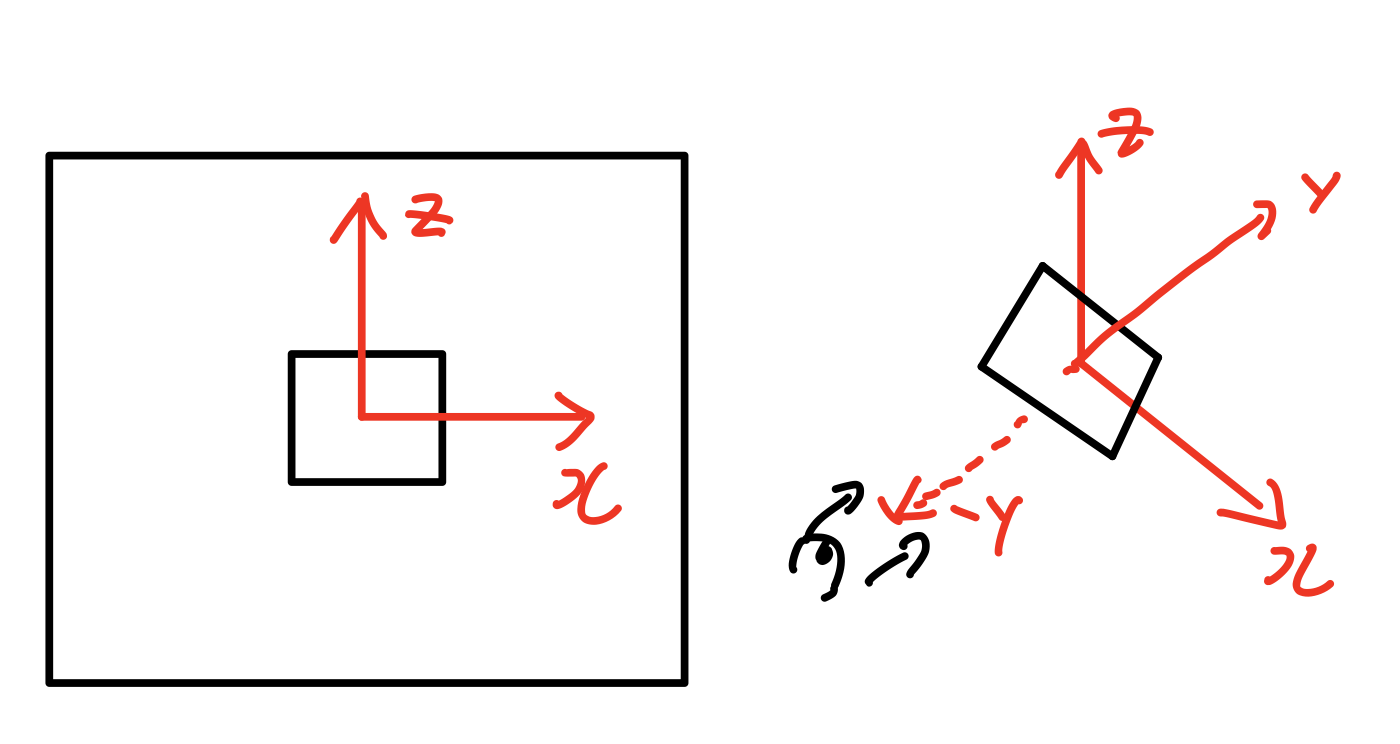
**20182931 신석경**

1. **모델 구성**

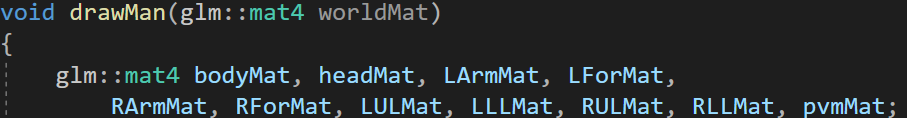




먼저 기본이 되는 worldMat이다. 가로, 세로, 높이가 모두 1이고 setAngel = -90 만큼 x축을 기준으로 회전시킨다. 이후, z축을 기준으로 rotAngle로 회전시킨다. 이 rotAngle은 후에 애니메이션을 설명할 때 설명하겠다. 이 동작 이후의 상황을 그림으로 나타내면,



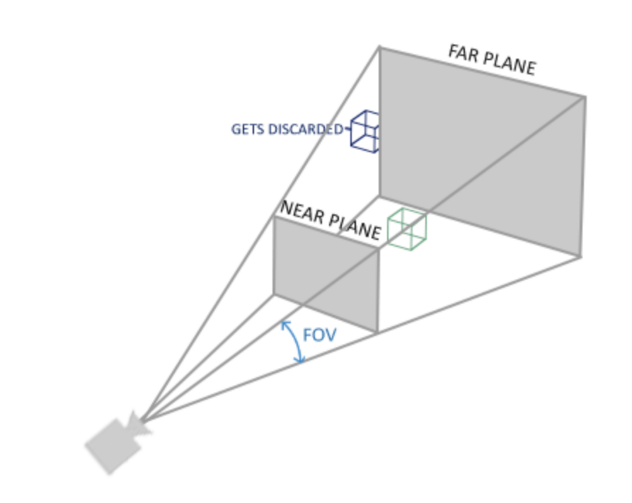
이런 상황이다. 왼쪽은 projection 면에서 보는 상황이고, 오른쪽이 3차원 좌표에서의 객체에 대한 좌표축이 된다. 사용자는 객체의 -y축 방향에서 보고 있는 것이다.



모델을 구성하는데 필요한 변수들이다.

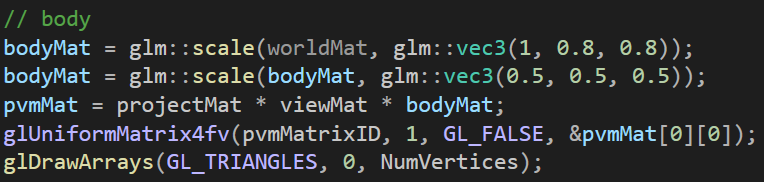


그리고 각각의 모델을 projection하기 위한 매트릭스이다. glm::perspective는

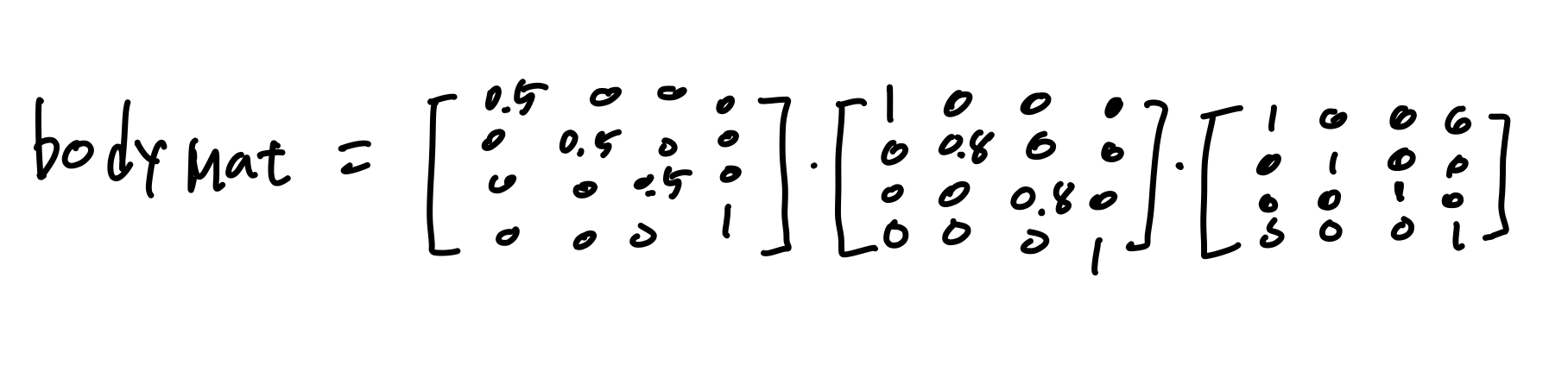
눈에 보이는 공간을 정의하는 큰 절도체(frustum)를 생성하는 함수이다. 첫 번째 파라미터는 FOV 값으로 view space가 얼마나 큰 지 결정한다. 두 번째 파라미터는 화면 비율로 우리의 경우 512/512 = 1 비율이다. 세 번째, 네 번째 파라미터는 가까운 면과 먼 평면의 거리로, 그 내부에 있는 것들이 렌더링된다.

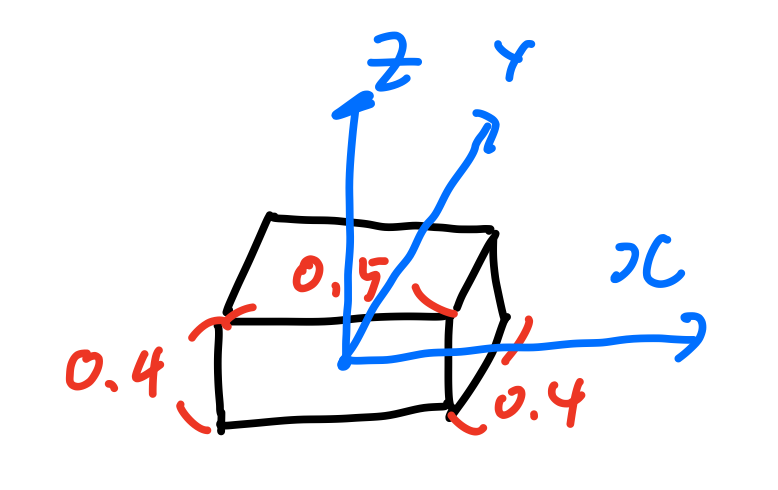
viewMat은 카메라를 정의하는 매트릭스이다. 첫 번째 파라미터는 카메라의 위치이다. x=0, y=0, z=2 위치에 카메라를 정의한다. 두 번째 파라미터는 카메라가 바라볼 점이다. 여기서는 원점을 보도록 하였다. 세 번째 파라미터는 화면 상에서 위쪽을 가리키는 벡터를 의미한다. y축을 설정하므로 2차원 카메라 view에서 가로는 x축 세로는 y축이 된다.

1. Body



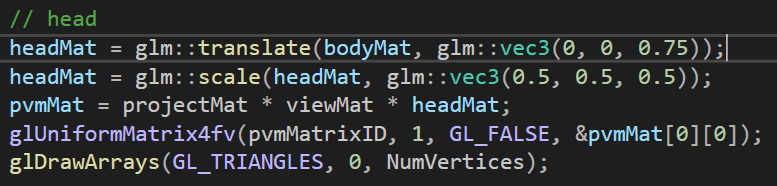
Body 부분이다. 1:1:1의 비율의 네모 박스를 1:0.8:0.8로 바꾼다. 그리고 그 다음 각각의 scale에 0.5씩 곱해 크기를 반절로 줄인다.



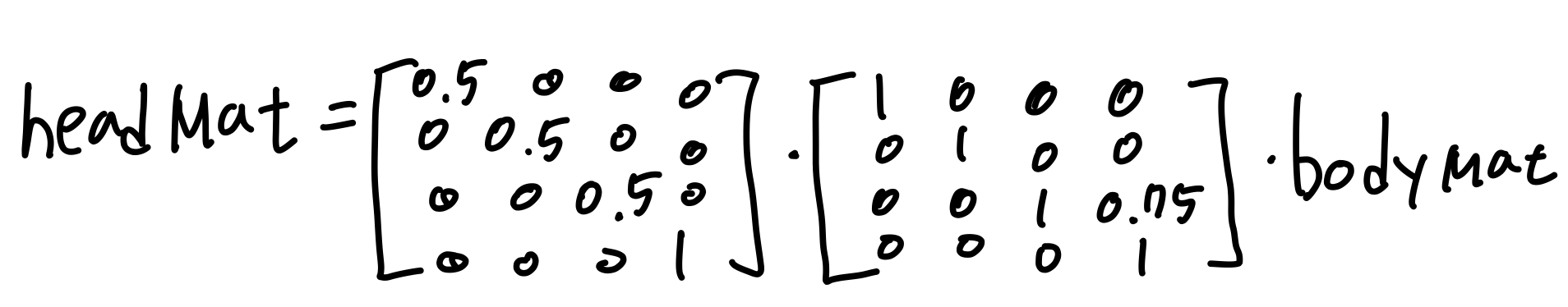


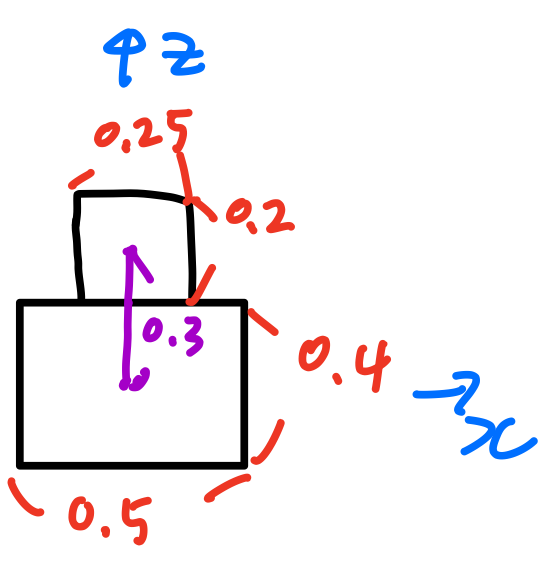
행렬식으로 나타내면 위처럼 되고, 그림으로 보면 왼쪽처럼 나타난다. 이제 여기에 viewMat과 projectMat을 곱해 화면 상에 출력한다.

1. Head



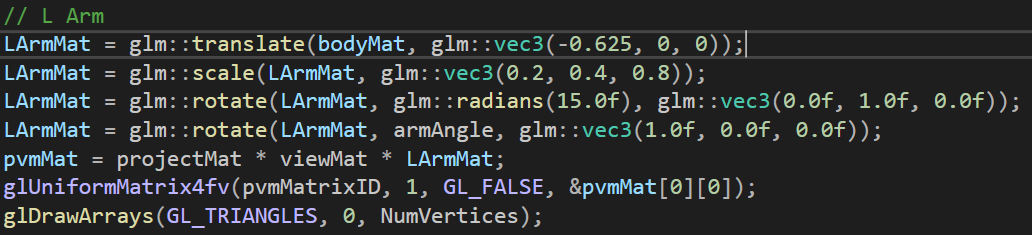
Head 부분이다. Body를 기준으로 z축으로 0.75 비율만큼 이동하고 전체 비율의 반절로 줄인다.



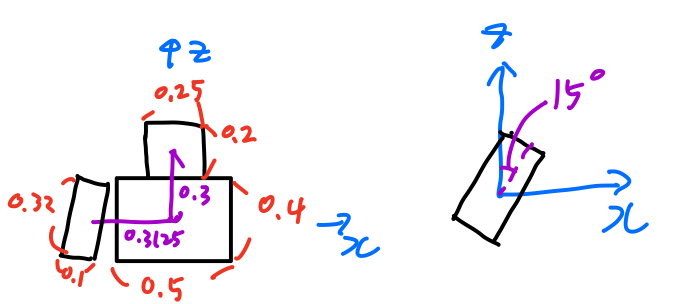
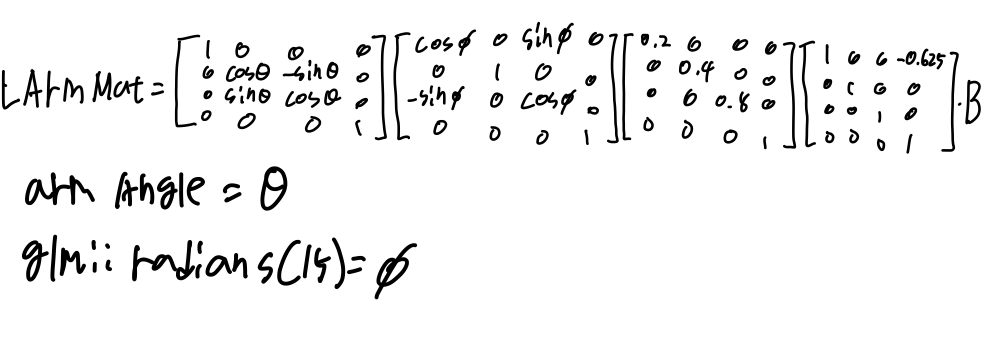


행렬식과 그림이다. z축 방향으로 0.75 비율만큼 이동한다. 기준이 되는 매트릭스가 bodyMat이므로, bodyMat의 z축 길이인 0.4의 0.75 배인 0.3만큼 이동한다. 그리고 0.5 배씩 scale 하므로, z축 길이는 0.25, x축 길이는 0.2, 그림에는 보이지 않지만 y축 길이도 0.2가 된다.

1. L Arm

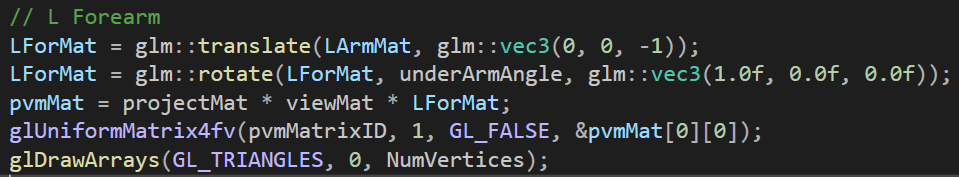


왼쪽 어깨 부분이다. 역시, bodyMat을 기준으로 x 축으로 – 0.625 비율만큼 이동하고 scale을 0.2:0.4:0.8로 바꾼다. 그리고 +y축에서 바라보는 방향으로 반시계 방향으로 15도 만큼 회전시킨다. 그리고 애니메이션을 위하여 x축을 기준으로 회전한다.

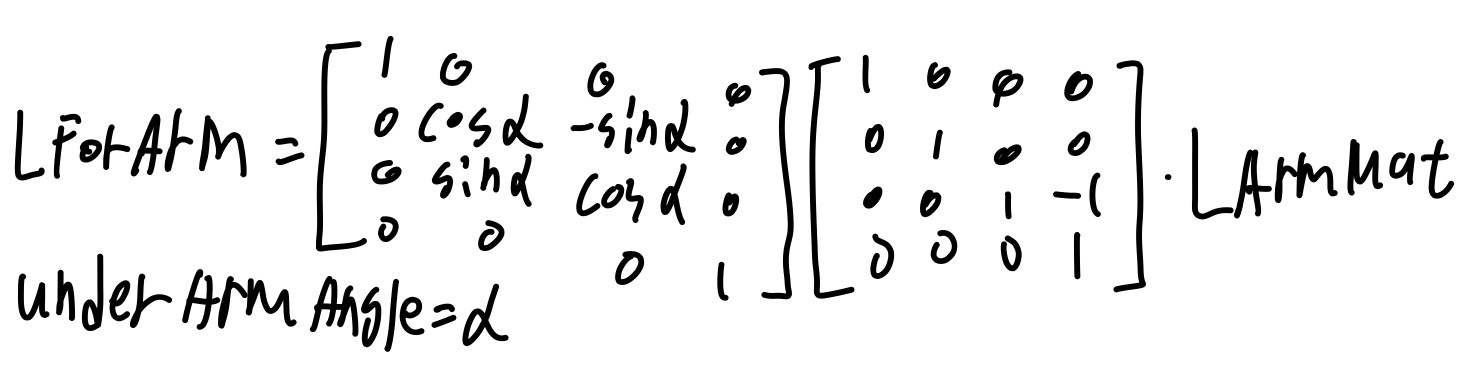


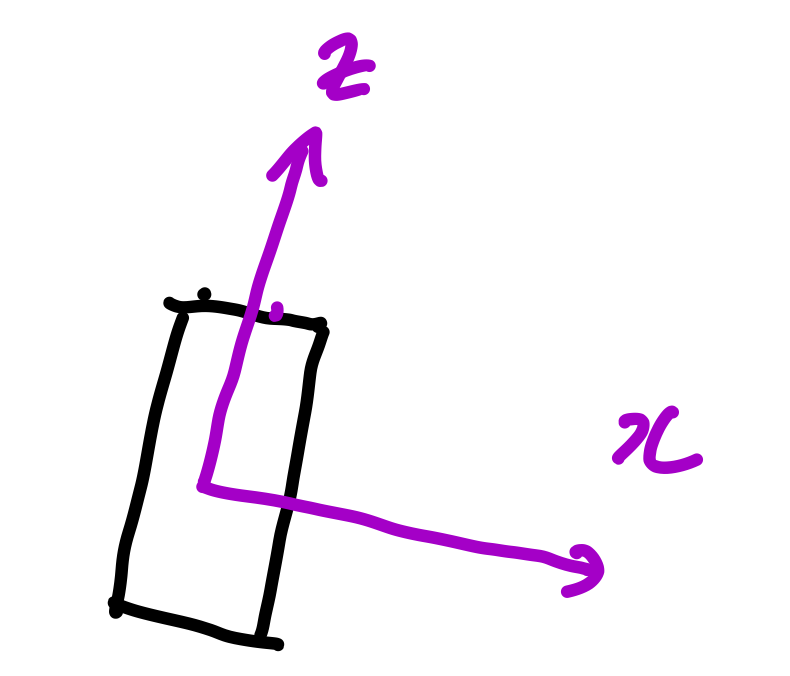
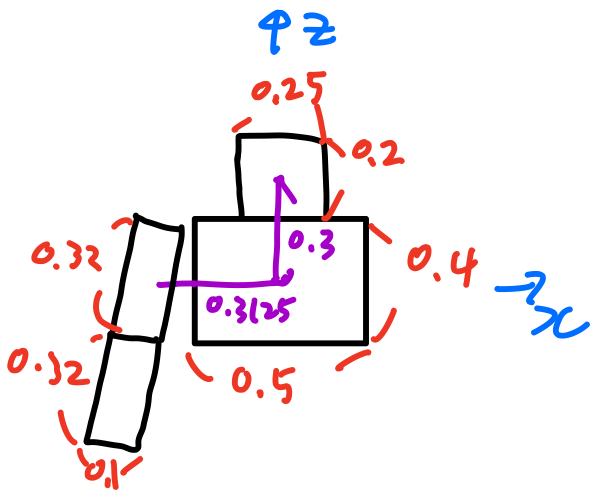
행렬식과 그림이다. B는 bodyMat을 의미한다. 먼저 translate가 x축으로 – 0.625만큼의 비율로 이동하니, -x축으로 0.5 \* 0.625 = 0.3125만큼 이동한다. 그리고 scale에서 x축으로 0.2 비율 = 0.5 \* 0.2 = 0.1, y축으로 0.4비율 = 0.4 \* 0.4 = 0.16, z축으로 0.8 비율 = 0.4 \* 0.8 = 0.32의 길이가 된다. 그리고 y축 기준으로 15도만큼 회전하는데, 현재 view가 물체를 기준으로 -y 축 방향에서 보고 있으니 y축으로 반시계로 15도 회전은 -y축에서 시계 방향으로 15도 회전하는 것과 같다. 그래서 실제 회전은 위의 그림처럼 오른쪽으로 기운 모양이 된다. 이후 x축을 기준으로 회전 하는 것은 시계, 반시계 방향을 둘 다 이동하니 view에서는 앞 뒤로 이동하는 것처럼 보인다.

1. L Forearm



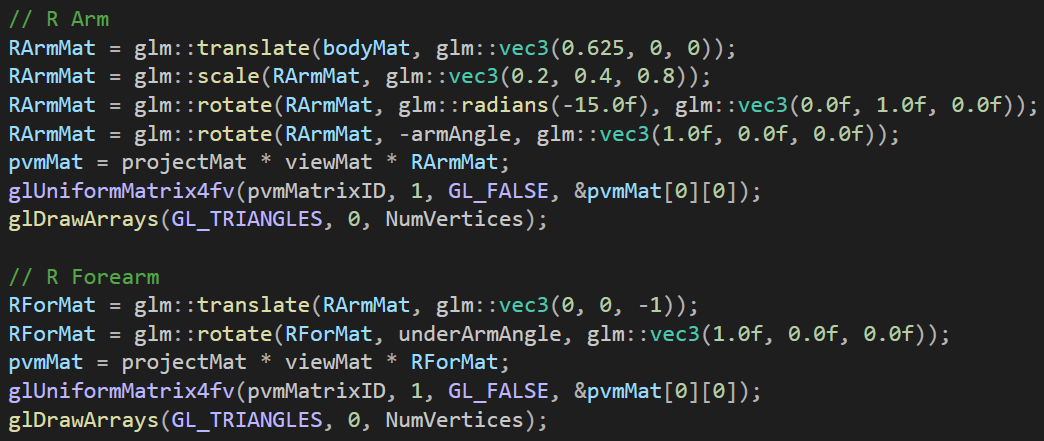
왼쪽 팔 부분이다. 어깨 부분을 z축으로만 -1의 비율로 이동하고, 애니메이션을 위하여 x축을 기준으로 회전한다.



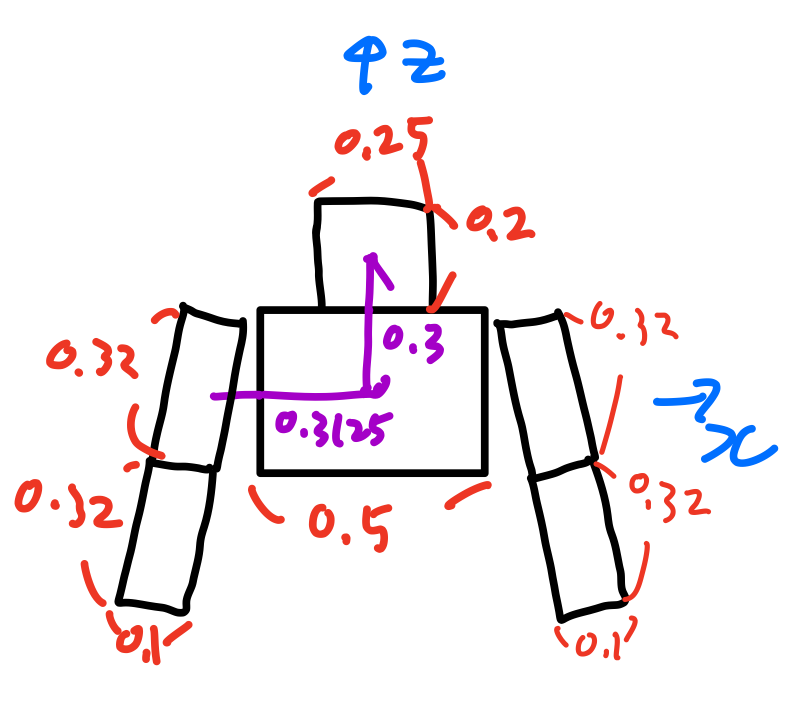
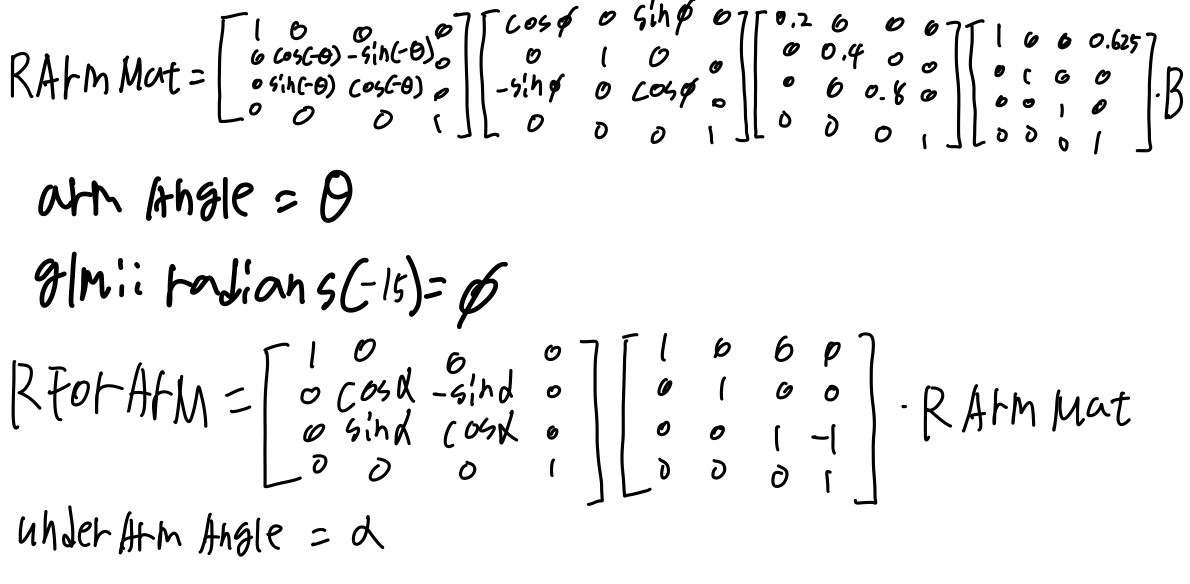


L Arm Mat의 z축으로 -1 비율만큼 이동한다. LArmMat은 이미 회전이 되어있는 상태이기 때문에 그것의 축만 따로 그려보면 오른쪽의 그림 같이 그려진다. 그렇기에 -1 만큼 이동하면 그대로 회전된 상태로 아래로 내려오는 것이다. 이후 애니메이션을 위하여 x축을 기준으로 회전시킨다.

1. R Arm, R Forearm

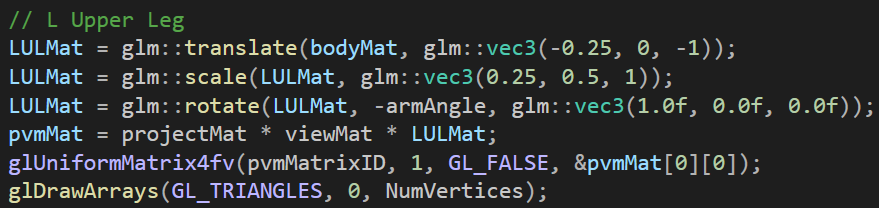


오른쪽 어깨와 팔 부분이다. 이 부분은 대부분 왼쪽 부분과 유사하다. 다른 점은 어깨의 위치가 bodyMat을 기준으로 +x축 방향 0.625라는 것과, 어깨의 회전 각도가 -15도 라는것, 또 어깨의 회전 방향이 왼쪽과 반대이기 때문에 armAngle에 -1을 곱했다는 것이다.

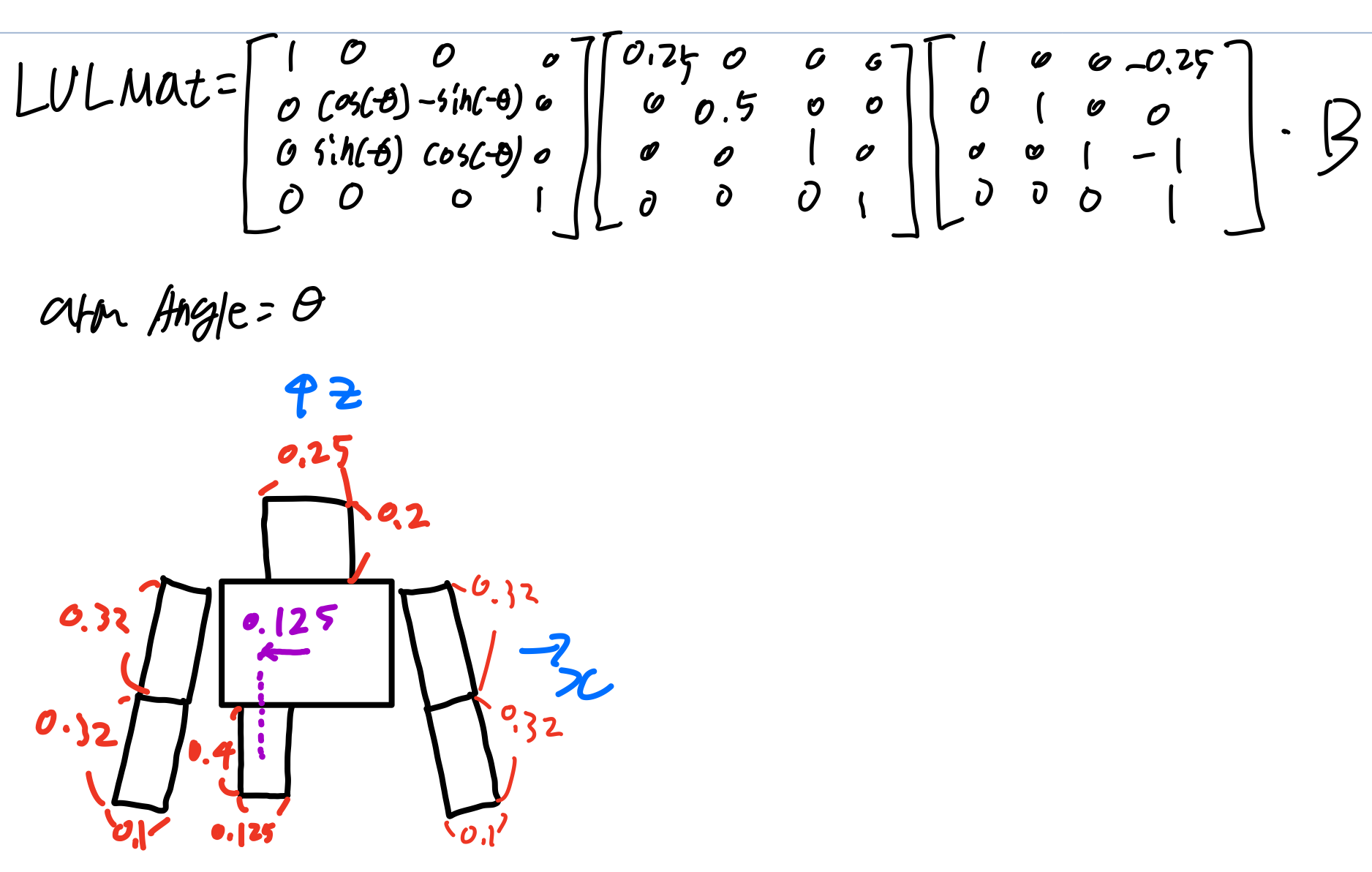


행렬식과 그림이다. 왼쪽 어깨, 팔과 반대이기 때문에 translate 결과와 회전 방향이 반대인 것을 확인할 수 있고, 나머지 부분에 대해서는 같다는 것을 확인할 수 있다.

1. L Upper Leg

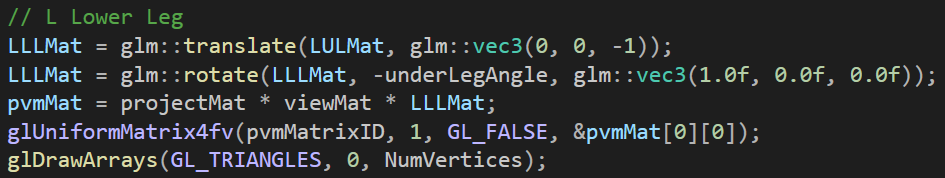


왼쪽 무릎 위 다리 부분이다. x축으로 – 0.25 비율, z 축으로 -1 비율 만큼 bodyMat을 기준으로 이동하고, scale을 0.25:0.5:1 의 비율로 바꾼다. 이후 애니메이션을 위하여 x축을 기준으로 -armAngle로 회전한다. L Upper Leg는 R Arm과 같은 각도로 움직이기 때문에 R Arm과 회전 방향을 일치시켰다. x축 길이는 0.5 \* 0.25 = 0.125, y축 길이는 0.4 \* 0.5 = 0.2, z축 길이는 그대로 0.4가 된다.

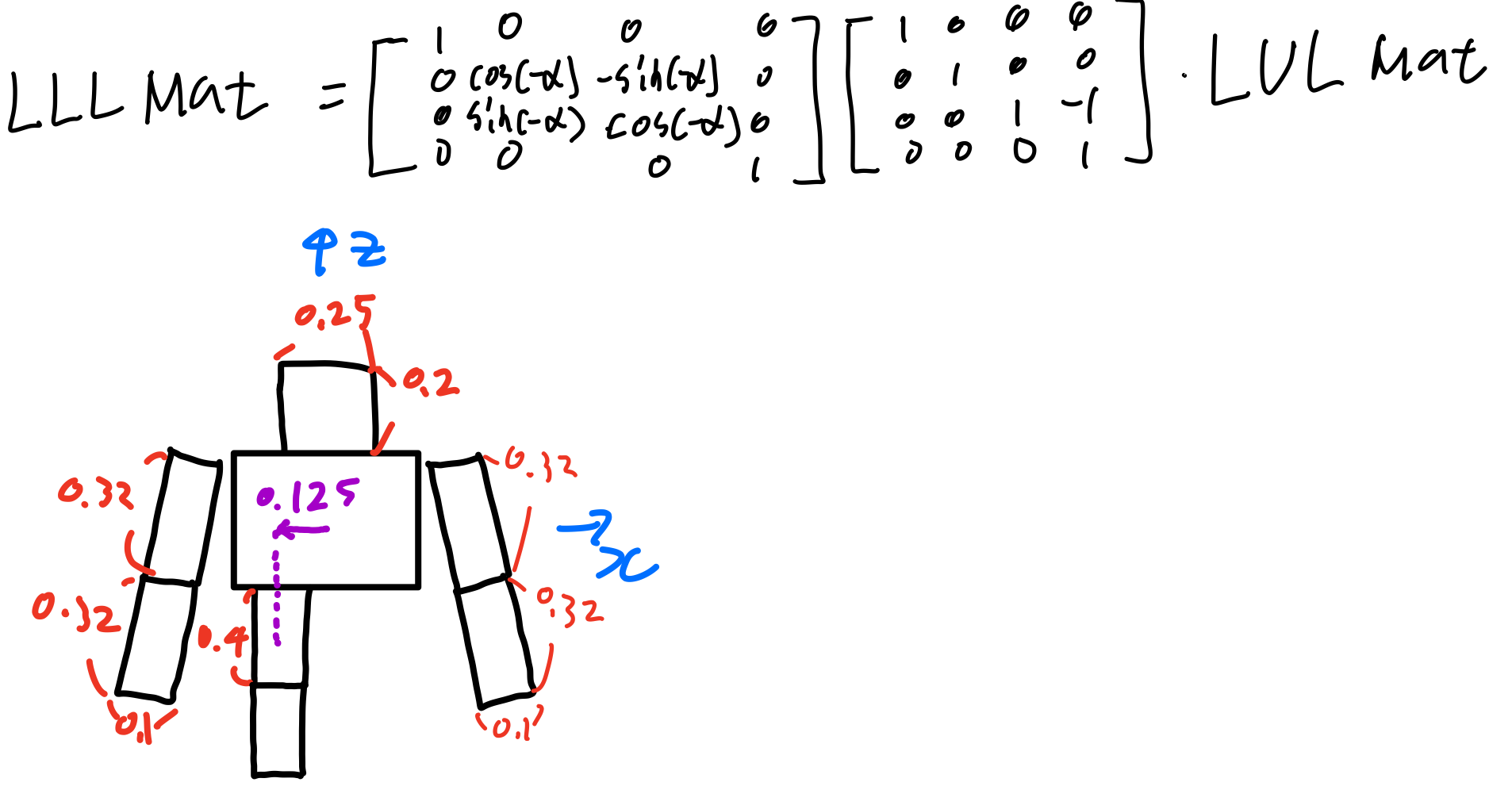


행렬식과 그림이다. z축으로는 아무런 scale변화가 없기 때문에 그대로 내려와서 같은 크기를 가지는 것을 확인할 수 있다.

1. L Lower Leg

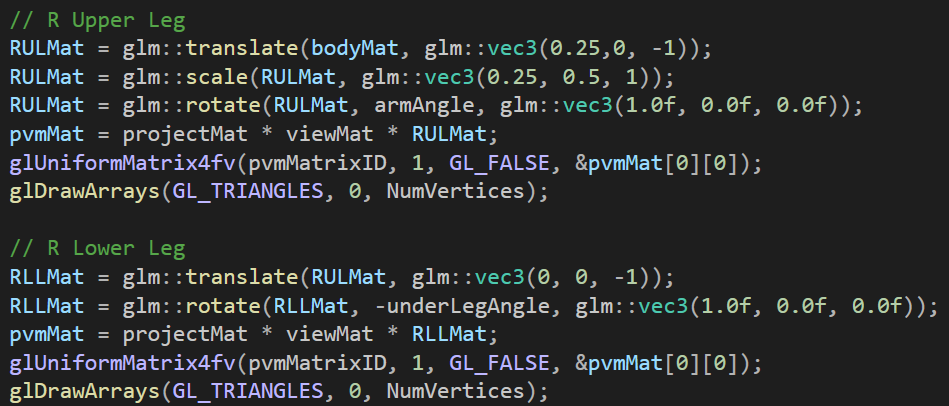


왼쪽 무릎 아래 다리이다. L Upper Leg를 그대로 z축으로 -1 만큼 내려오고, x축에 대해 회전하도록 하였다. 팔과는 다르게 x축에 대해 시계 방향으로 회전하기 때문에 -각도를 준다.

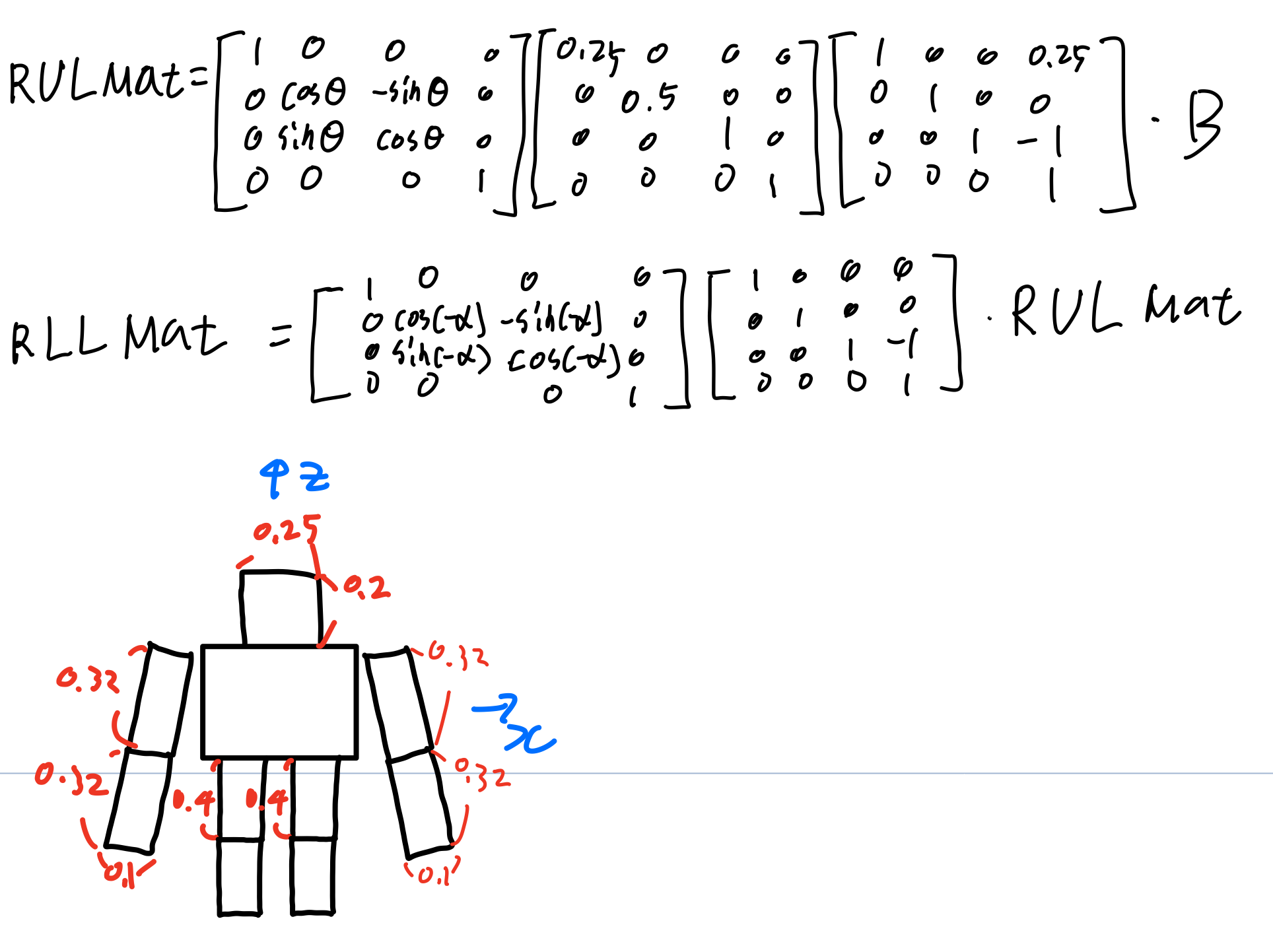


행렬식과 그림이다. LULMat에 대해서 z축으로 동일한 비율만큼 이동하기 때문에 그냥 아래로 내려온 그림이다.

1. R Upper Leg, R Lower Leg



오른쪽 다리 부분이다. 이것 또한 왼쪽 다리와 유사하지만, 기준이 되는 x좌표가 bodyMat을 기준으로 +0.25 부분이라는 점, 회전이 왼쪽 다리와 반대라는 점만 다르다.



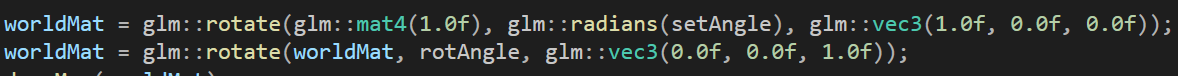
행렬식과 그림이다.

이렇게 모델의 구성은 끝났다. 이제 애니메이션에 대해 알아보자.

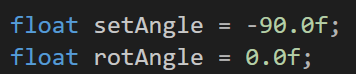
1. 애니메이션

이 프로젝트의 애니메이션에는 크게 모델 전체에 대한 애니메이션과 모델 각 부분에 대한 애니메이션이 있다.

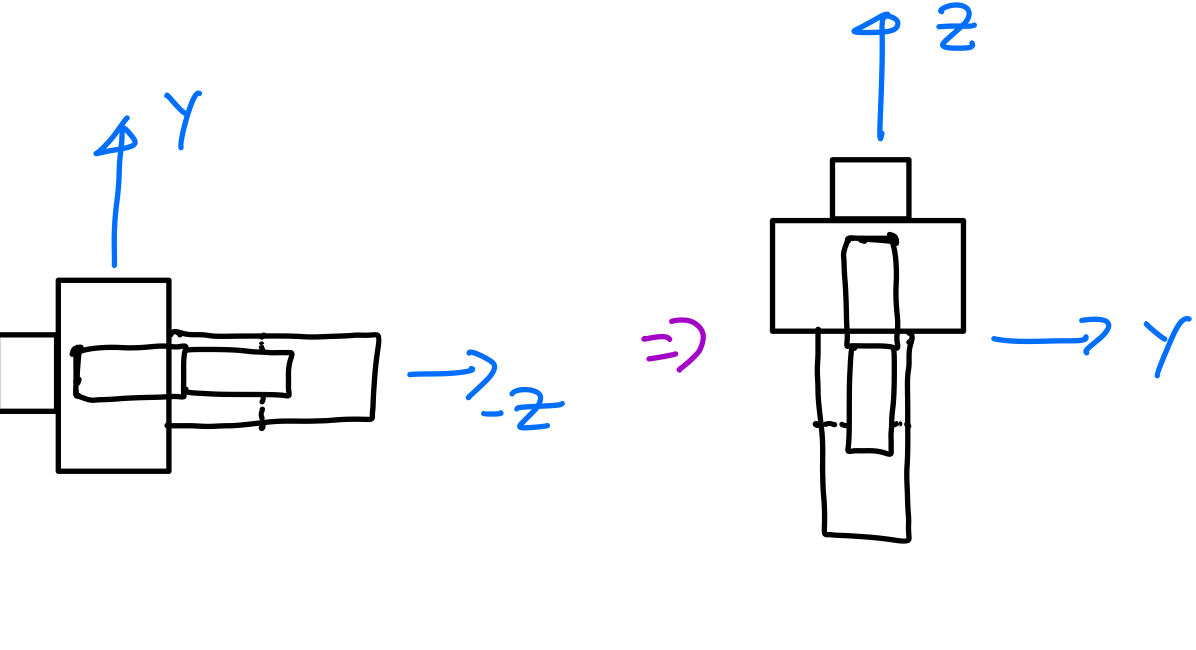
1. 모델 전체



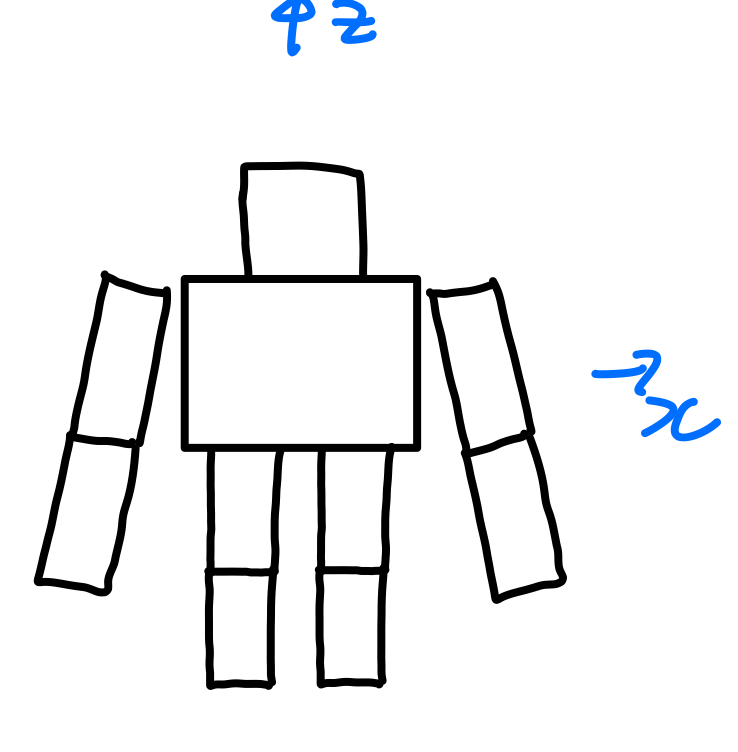




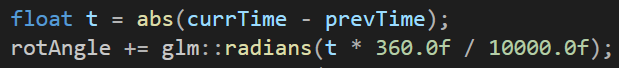
모델 전체에 대한 애니메이션이다. 먼저 기본 space의 x축을 기준으로 setAngle = -90도 만큼 회전한다. 즉, +x축에서 바라봤을 때 시계방향으로 90도 회전한 것이다.



이런 식의 회전이 된다. 위의 그림 view는 +x 축 방향에서 보고 있는 것이다. 이제 이것을 실제 카메라 view에서 바라보면



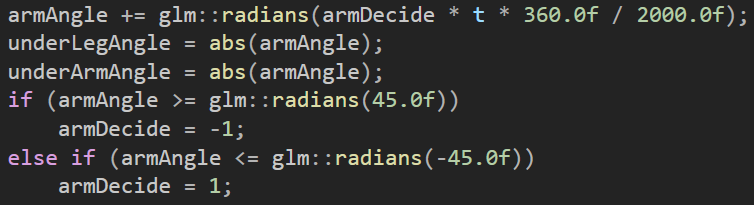
이런 view가 되는 것이다. 이제 이 모델을 z축에 대해서 반시계 방향으로 rotAngle만큼 회전시키는데 worldMat의 +z축 방향에서 바라보는 것이므로 위의 모델이 오른쪽으로 회전하는 것처럼 보일 것이다.



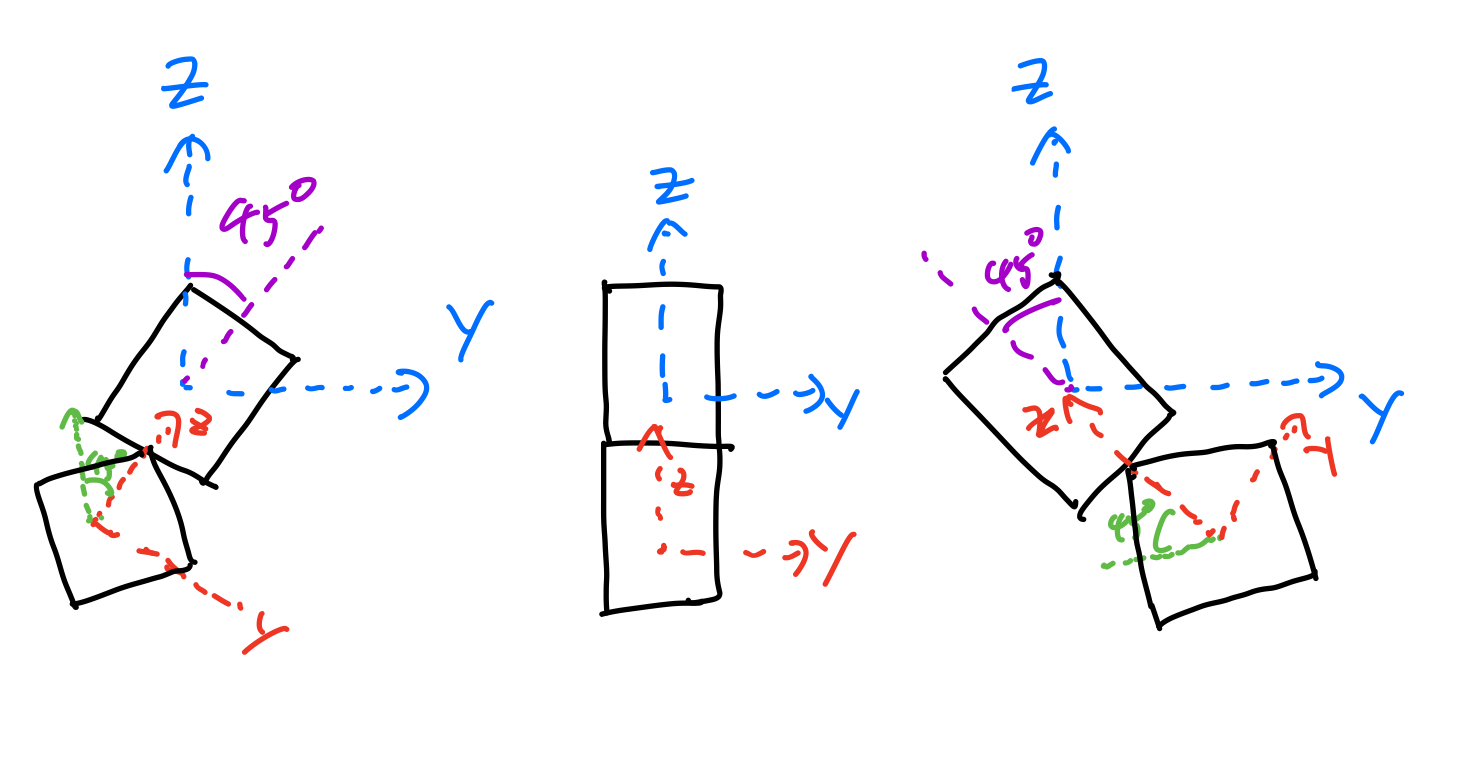
이 rotAngle은 Time-based로 회전하는데, 일정 시간이 지날 때마다 각도를 더하여 물체의 회전을 구현한다. 위의 코드에서는 일정 시간에 대하여 t \* 360 / 10000 만큼 회전 각도를 더하는데, t == 10000가 될 때 한 바퀴를 도는 것이다. 즉 10초에 1바퀴씩 돌도록 설정하였다.

1. 왼쪽 어깨와 왼쪽 팔





왼쪽 어깨와 왼쪽 팔의 회전에 대한 코드이다. 둘 다 객체 기준 +x축 방향을 중심으로 회전한다. 회전에 알아보기 앞서 먼저 각도에 대해 살펴보면, armAngle이 모든 회전의 기반이 된다. armAngle은 -45 ~ +45 도 사이의 각도를 가진다. -45를 시작으로 + t \* 360 / 2000이 45 가 되려면 t = 500이어야 하기에, 1초 마다 + 45 -> -45 -> +45 각도를 이동한다. 아래 팔의 경우 항상 앞 쪽으로만 회전해야 하기 때문에 armAngle에 절댓값을 취하여 armAngle에 맞춰 같이 회전하도록 구현하였다.

왼쪽부터 각각 -45, 0, +45 도 일 때의 상황이다. 그림 실력이 부족하여 완벽히 같지는 않지만 이와 비슷하게 작동한다. -45도 인 상황을 가정하고 설명하자.

전체 모델 좌표축이 파란색 일 때, +x축에서 바라본 기준으로 시계방향으로 45도만큼 회전한 것이 왼쪽 어깨의 축이 된다. 이것을 빨간색 좌표축으로 표현하였다. 왼쪽 아래 팔은 왼쪽 어깨 축을 기준으로 회전하기 때문에 빨간색 좌표축에서 abs(-45) = 45도만큼 반시계로 회전하면 맨 왼쪽 그림 같이 나온다.

+45도 인 상황도 마찬가지로 파란색이 모델 좌표축이고 빨간색이 회전한 왼쪽 어깨의 좌표축일 때, 이에 맞춰서 아래 팔이 45도만큼 회전하기 때문에 맨 오른쪽 그림과 같은 모양이 나온다.

이제 0도 일 경우에는 아무런 축 변화가 없기 때문에 가운데처럼 가만히 있는 모양이 나온다.

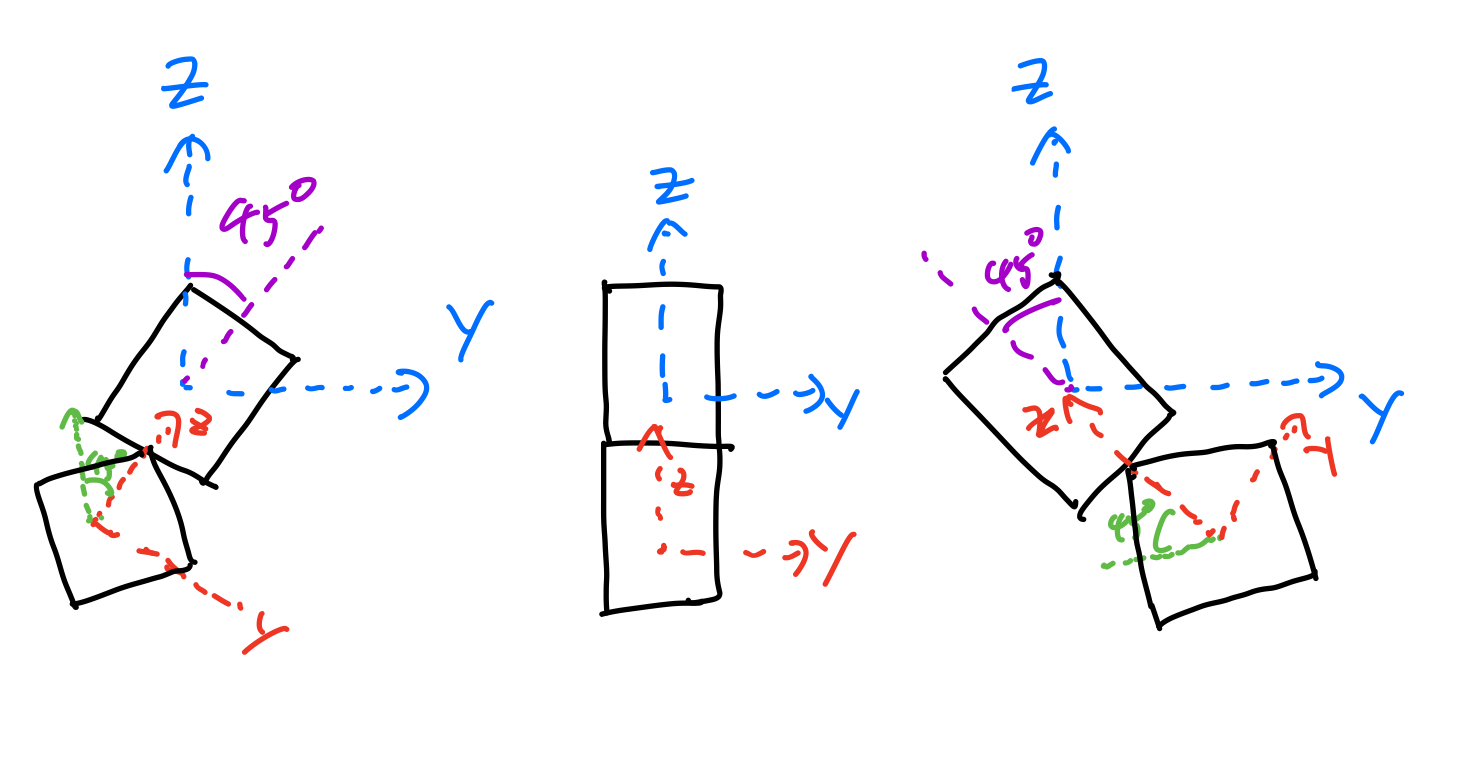
애니메이션은 가운데 그림부터 시작하여 오른쪽 그림 -> 가운데 그림 -> 왼쪽 그림 -> 가운데 그림 -> 오른쪽 그림 …. 순으로 보여진다.

1. 오른쪽 어깨와 오른쪽 팔





오른쪽의 경우 오른쪽 어깨 부분만 왼쪽 어깨 부분과 다르다. 즉 왼쪽 어깨가 +45도 이면 오른쪽 어깨는 -45도 이고, 그 반대의 경우도 마찬가지이다.



이 그림으로 다시 설명하면 왼쪽 어깨가

가운데 -> 오른쪽 -> 가운데 -> 왼쪽 -> 가운데 -> 오른쪽 …. 순 일 때,

오른쪽 어깨는

가운데 -> 왼쪽 -> 가운데 -> 오른쪽 -> 가운데 -> 왼쪽 …. 순으로 진행한다.

1. 왼쪽 허벅지와 왼쪽 무릎 아래

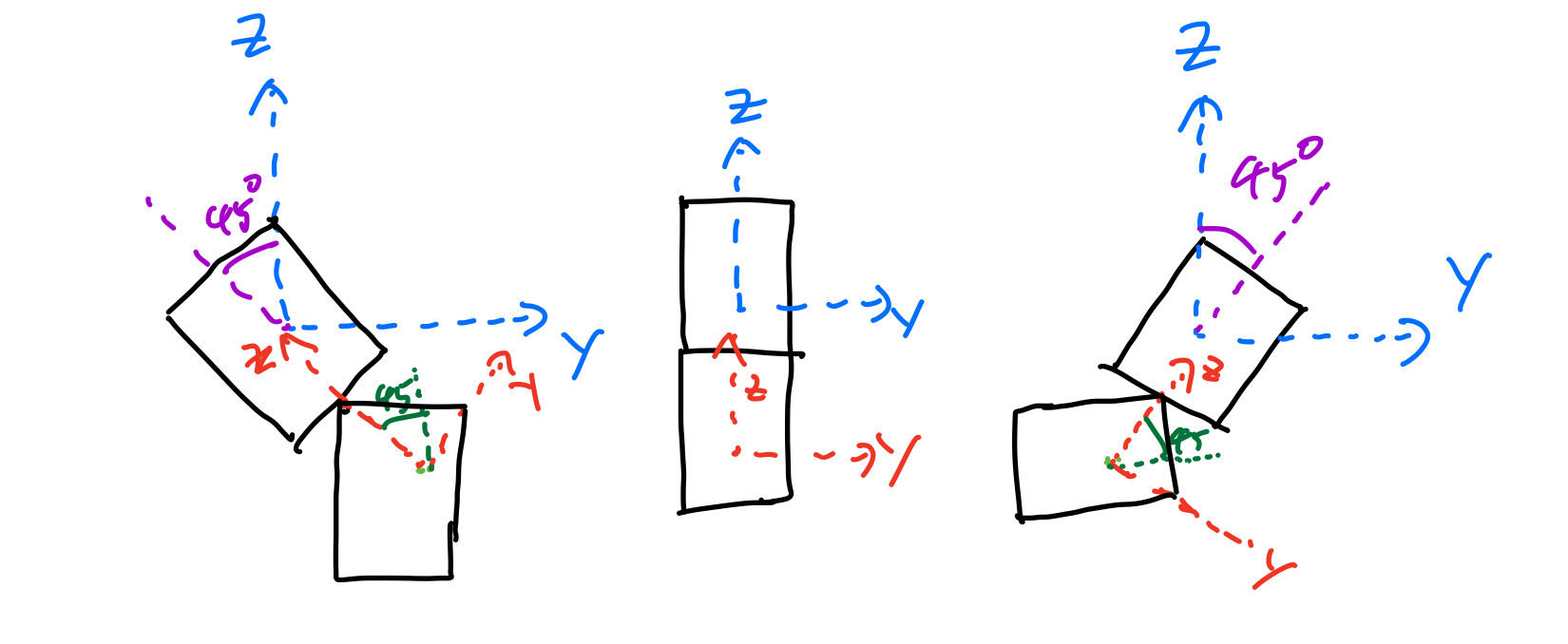






왼쪽 다리에 관한 회전이다. 다리의 회전은 팔의 회전을 기반으로 움직이기 때문에 armAngle을 그대로 사용하되, 부호를 바꿔서 같은 방향에 대해서는 반대 회전을 하도록 하였다. underLegAngle의 경우, underArmAngle과 같은 값이지만 이후 두 각도의 독립성을 위하여 따로 변수로 설정하였다.

팔과 같은 각도를 기반으로 움직이기 때문에 -45 ~ +45 도 사이의 각도를 움직이며 동일하게 1초를 기준으로 -45 와 +45 도 각도가 뒤바뀐다. 팔과 다른 점은 다리의 경우 아래쪽 다리가 위쪽 다리의 각도의 -절댓값으로 움직이기 때문에(-underAngle) 팔과는 다른 운동 양식을 보여준다.



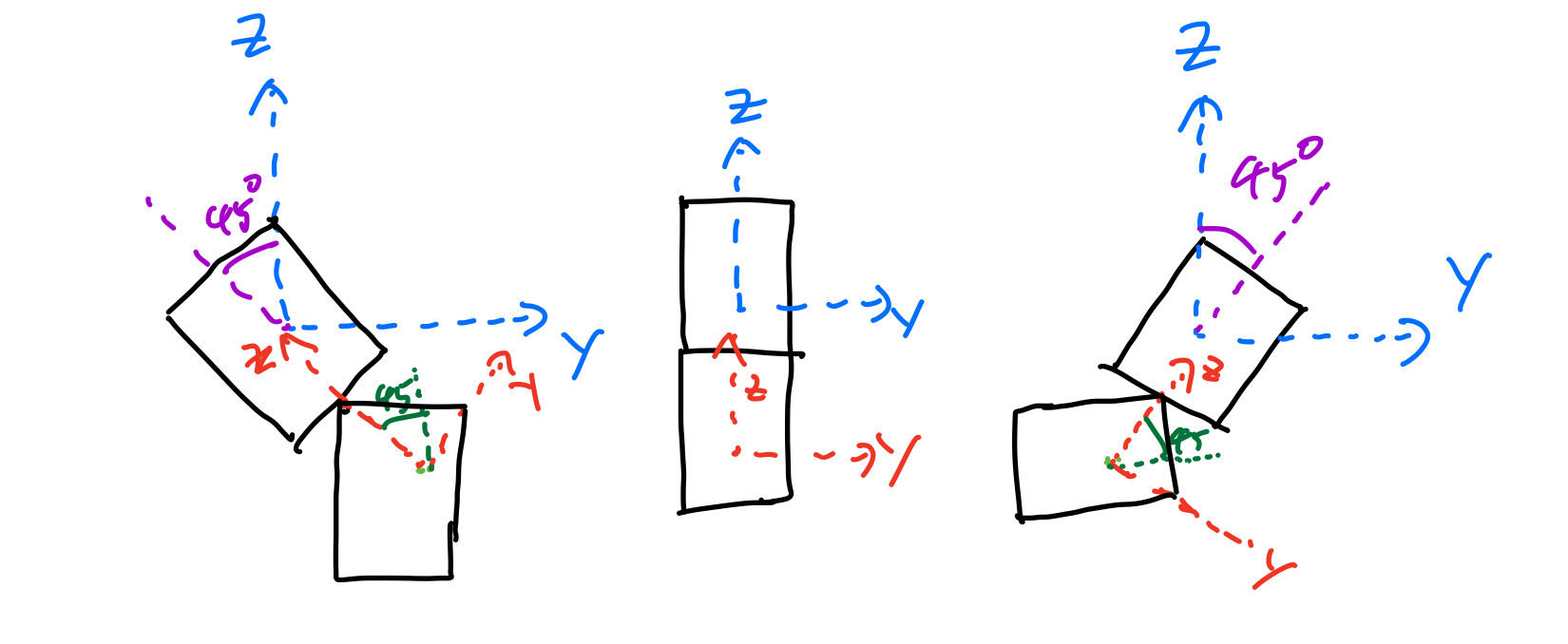
다리의 운동 양식이다. 왼쪽부터 armAngle이 -45, 0, +45 도인 경우이다. 다리의 회전 각도가 -armAngle이니 팔과는 반대로 회전한다. 다리 아래의 회전 각도도 -underLegAngle 이기 때문에 항상 위쪽 다리의 축을 기준으로 음의 방향(시계방향)으로 회전하는 것을 확인할 수 있다. 왼쪽 다리의 경우 armAngle에 따라서 가운데 -> 오른쪽 -> 가운데 -> 왼쪽 -> 가운데 -> 오른쪽 -> …. 순으로 동작한다.

1. 오른쪽 허벅지와 오른쪽 무릎 아래





오른쪽 다리는 왼쪽 다리의 동작에서 위쪽 다리의 회전 각도의 부호만 다른 것으로 이는 왼쪽 팔과 오른쪽 팔의 동작과 유사하다.

즉, 

이 그림에서 왼쪽 다리가

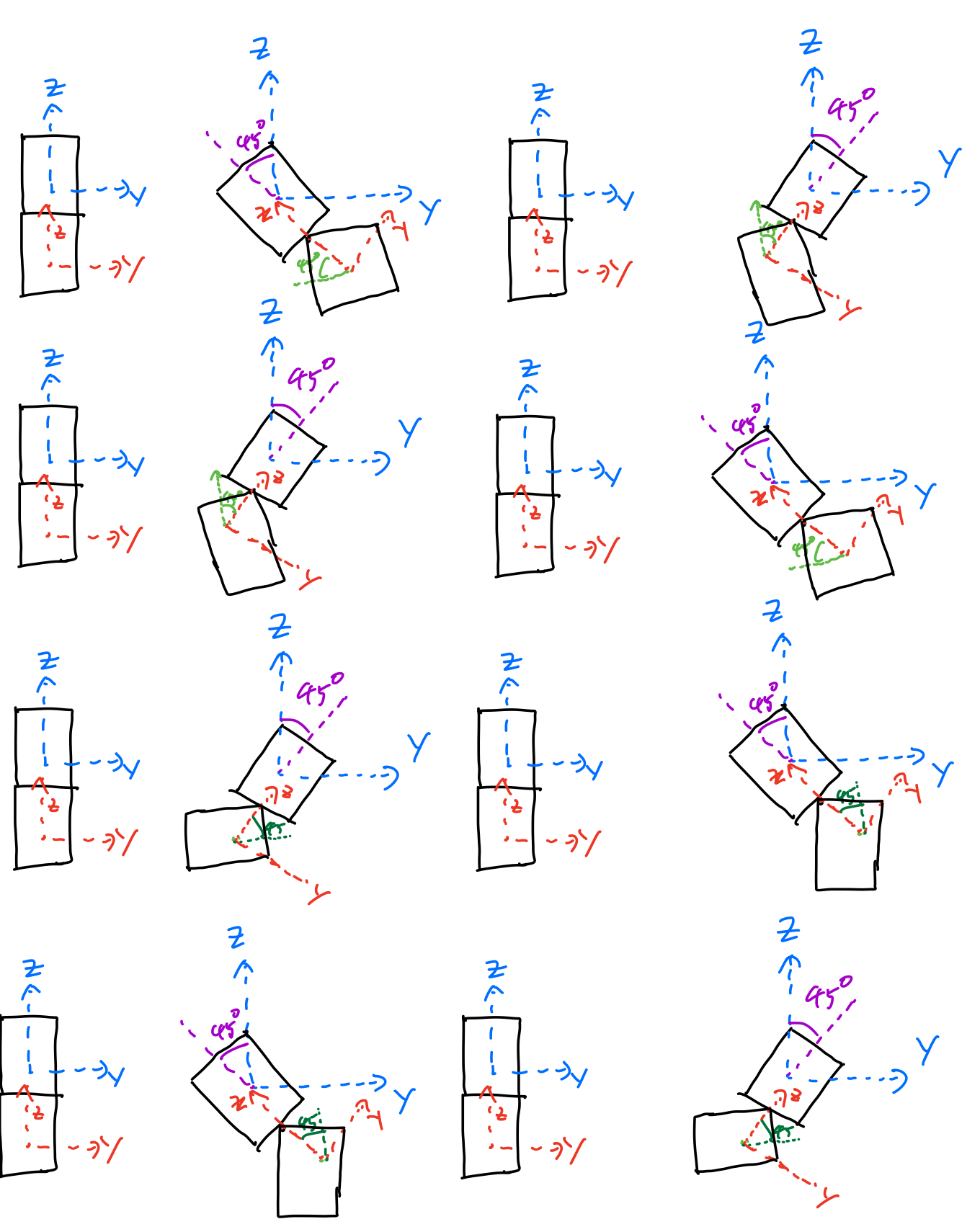
가운데 -> 오른쪽 -> 가운데 -> 왼쪽 -> 가운데 -> 오른쪽 -> …. 순이었다면

오른쪽 다리는

가운데 -> 왼쪽 -> 가운데 -> 오른쪽 -> 가운데 -> 왼쪽 -> …. 순으로 동작한다.

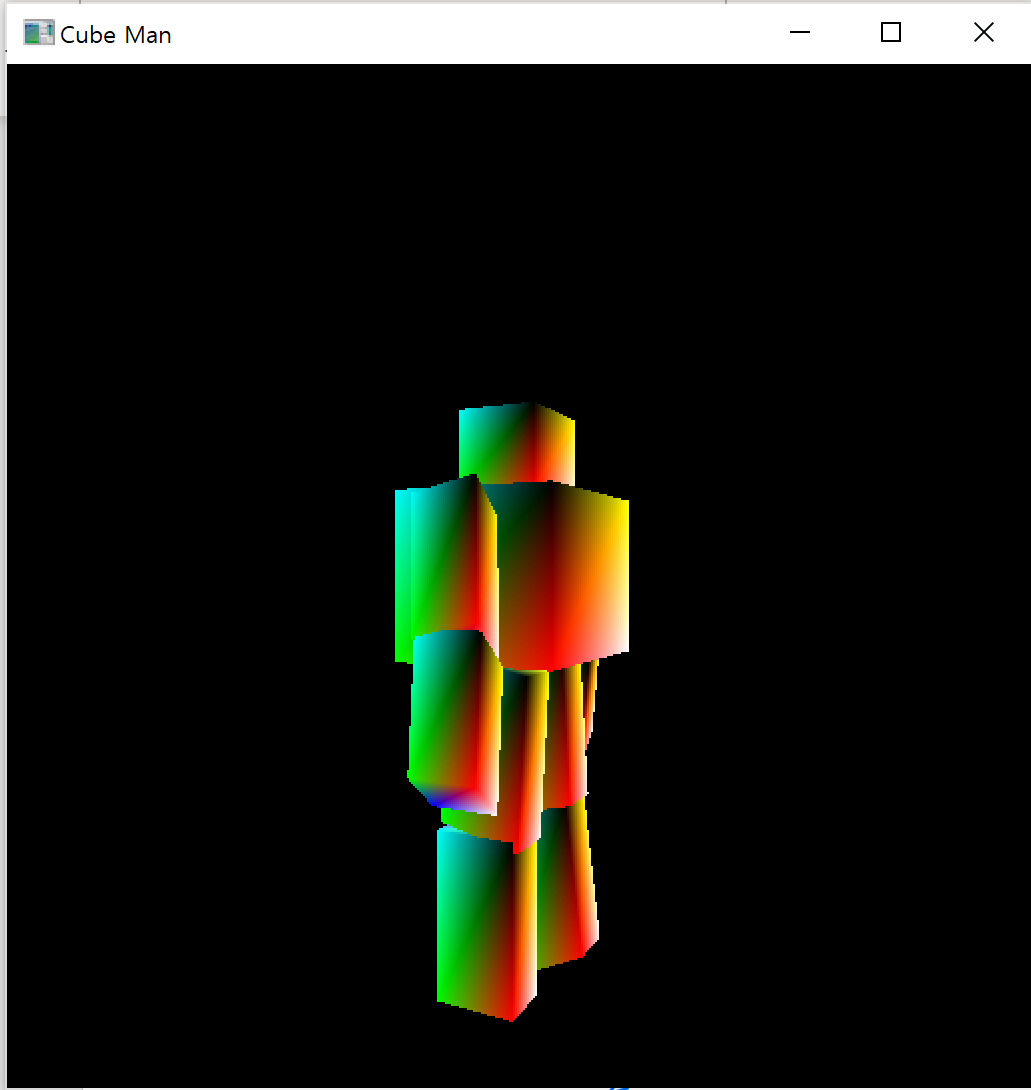
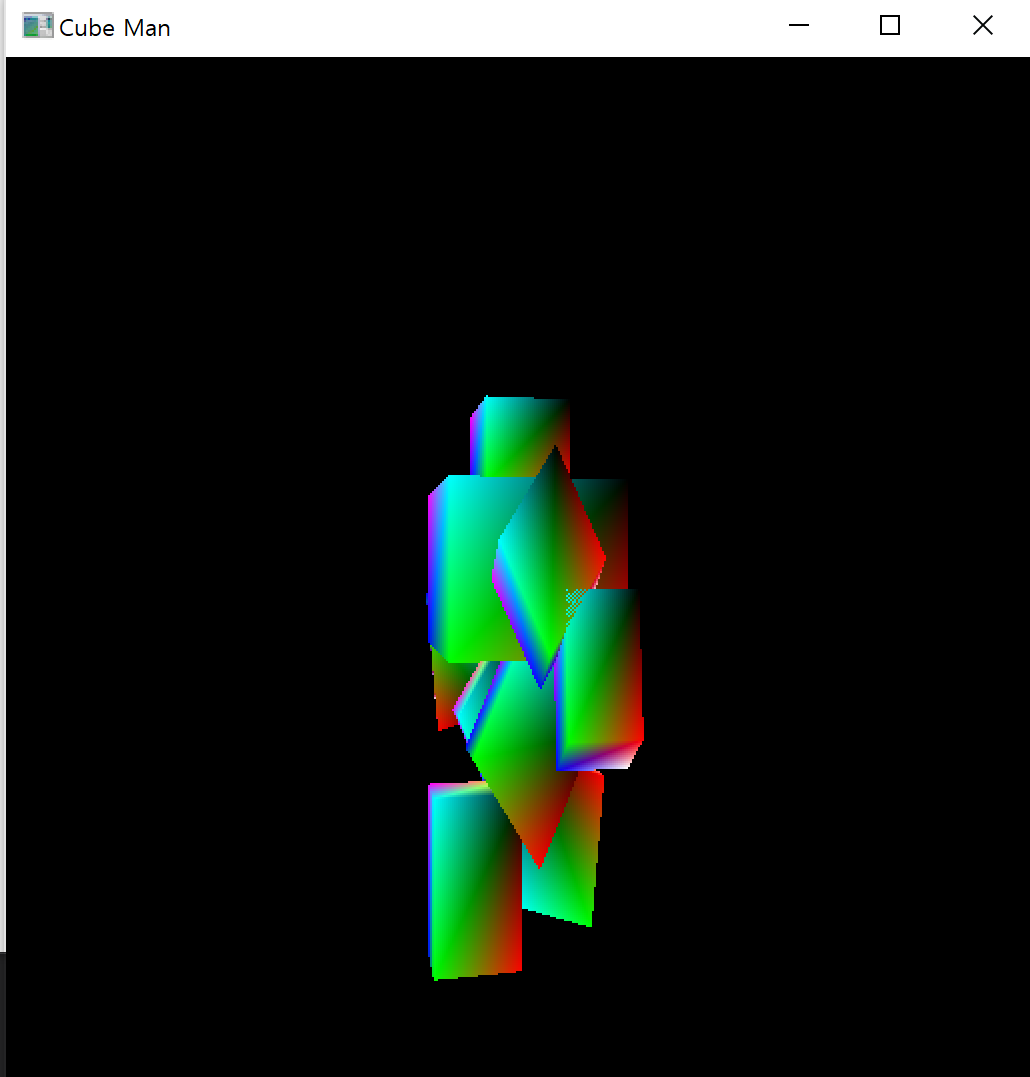
1. 모든 애니메이션

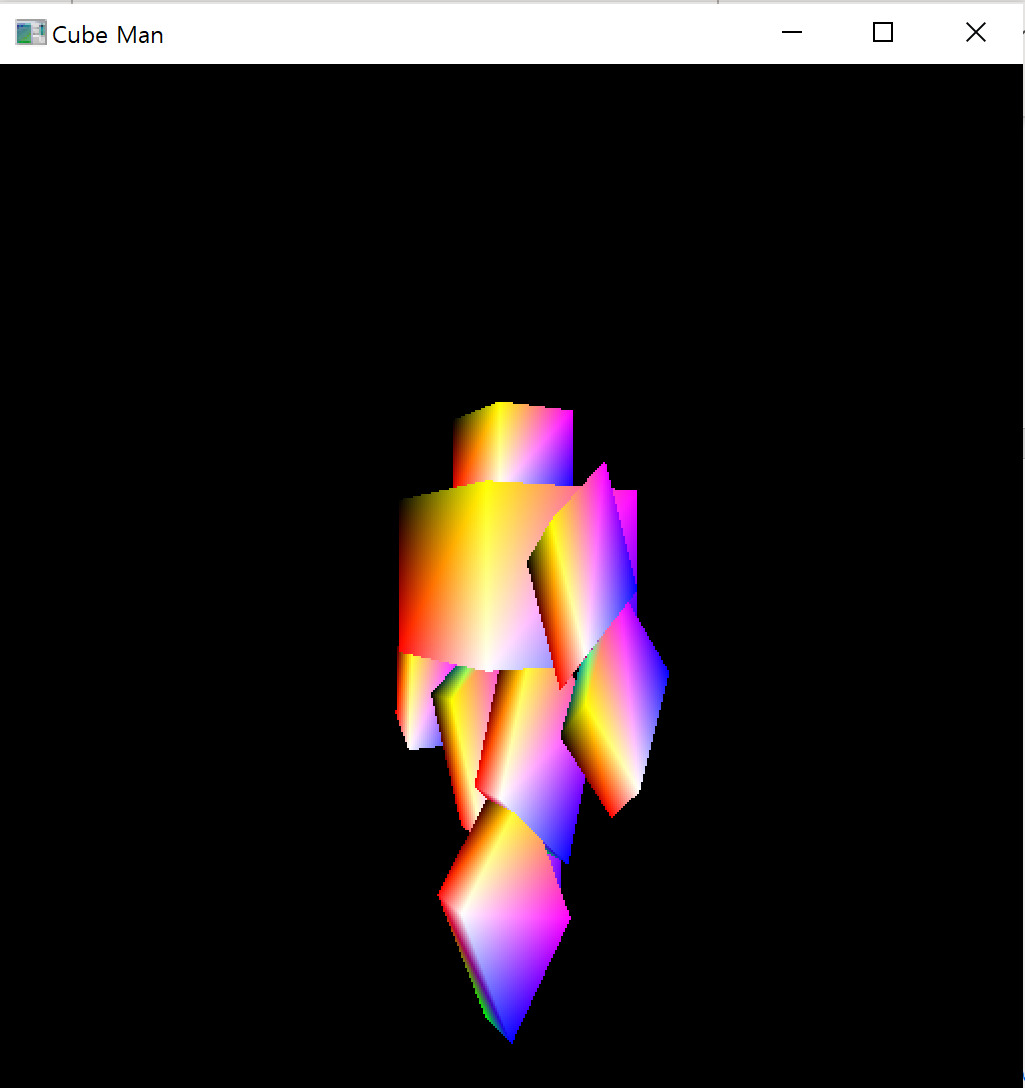
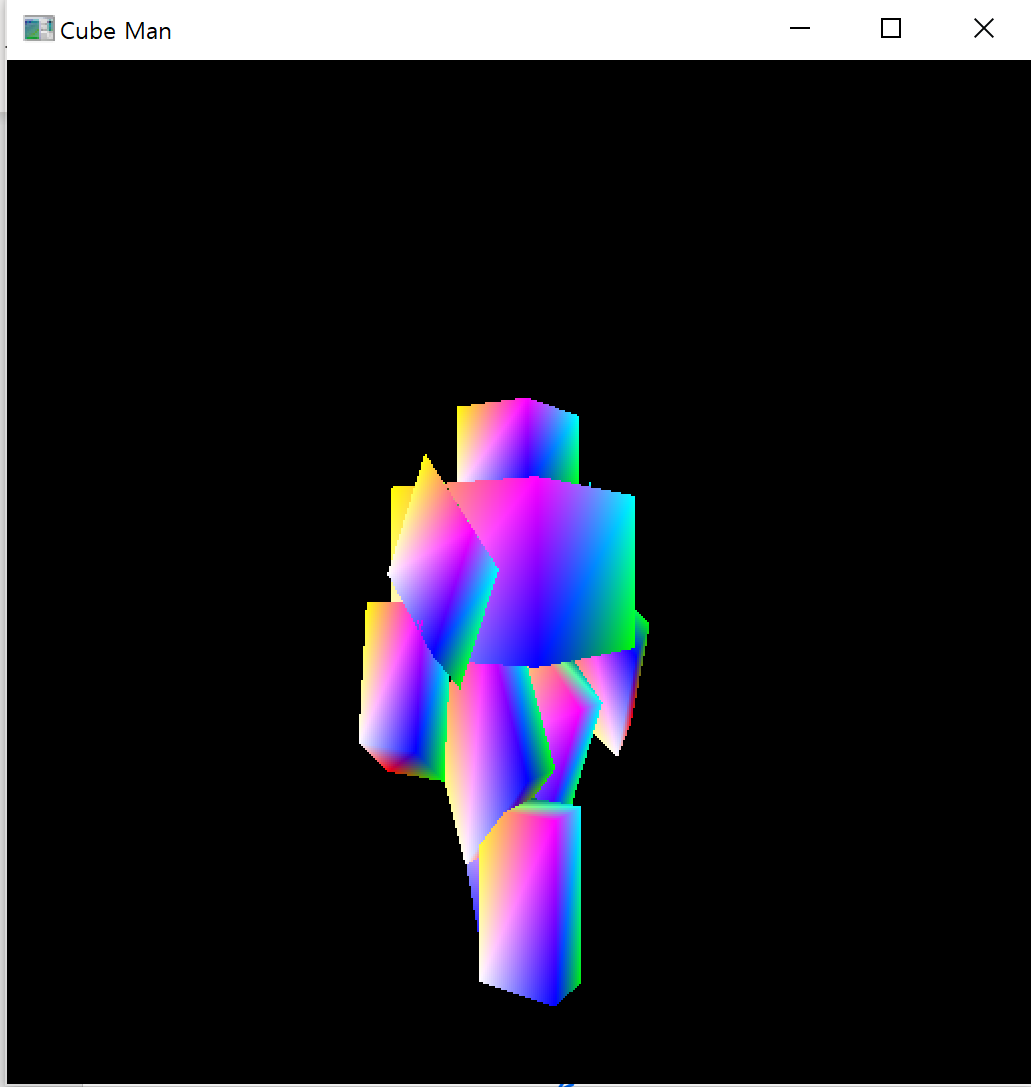
모든 애니메이션을 그림으로 나타내면



순서대로 왼쪽 팔, 오른쪽 팔, 왼쪽 다리, 오른쪽 다리이다.

1. 최종 모델



**소스코드 위치: /src/cube\_man.cpp**

**작업 OS: Windows 10**

**작업 컴파일러: Visual Studio 2019**