게임프로그래밍
기말과제 보고서

<목차>

학과	미디어기술콘텐츠학과
학번	202020128
이름	김가을

1. 요구조건 1 - 공 크기 축소

키보드 2번 키를 누르면 현재 발사 대기중인 공의 반지름이 1/2로 축소되고, 1 키를 누르면 원래대로 돌아와야 한다.

이를 위해 ProjectileGame.h에 bool isSmall 변수를 추가하여 키보드 입력에 따른 공의 상태를 지정해주었다. Ball.h 에 setRadius() 함수를 생성하여 공의 반지름과 위치를 설정할 수 있게 하였다.(편의상 함수의 매개변수는 생략하여 표기) ProjectileGame.cpp의 HandleEvents()에서 키보드 버튼이 눌릴 때, 발사 대기중인 공, 즉 공 벡터의 마지막 요소의 반지름을 바꾸었다. 그리고 다음 키보드 입력이 들어오기 전까지 공의 상태를 유지하기 위해 AddNewBall()에서 새 공을 추가할 때 알맞은 크기로 적용되게끔 조정하였다.

그러자 아래 그림1, 그림 2와 같이 공의 반지름 길이는 바뀌는데, 위치가 모서리에 붙어있지 않는 부자연스러운 현상이 나타났다. 따라서 위치를 조정하는 setPosition()함수를 추가하여 공의 위치를 올바르게 수정하였다.(그림 3, 4)

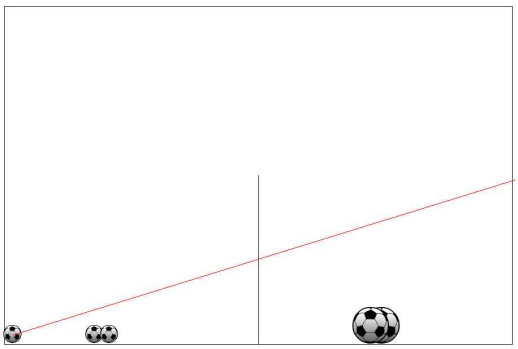


그림 1

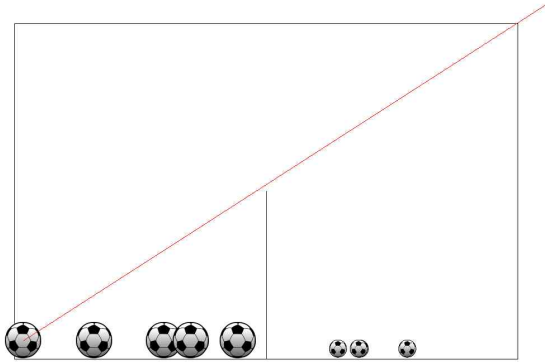


그림 2

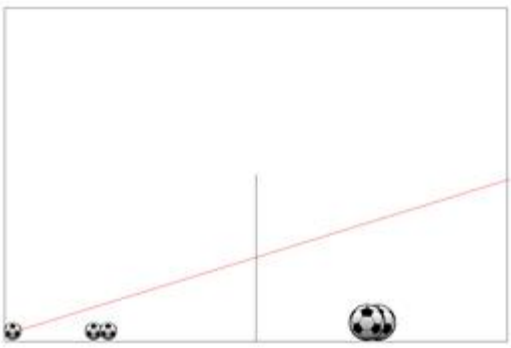


그림 3

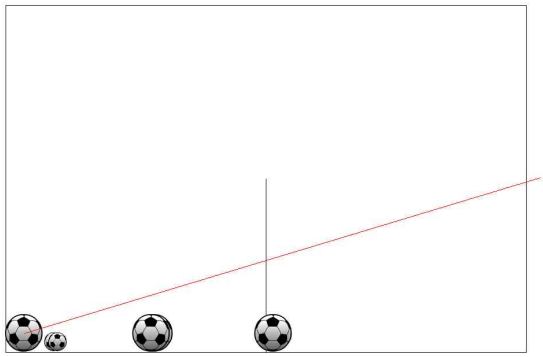


그림 4

2. 요구조건 2 - 마찰력 구현

마찰력을 구현하기 위해 Ball.h에 double friction(마찰계수)을 0.25로 정의하였고, Ball.cpp의 Update()에서 구현했다. 마찰력은 공의 밑부분이 방바닥 좌표보다 같거나 작은지로 확인하여 공이 바닥에 있을 때 적용하였다.

중력 가속도는 음수(-9,8)이므로, 마찰력 계산 시 양수로 만들기 위해 -1을 곱했다. 마찰력을 구현하기 위해 적용한 식은 다음과 같다.

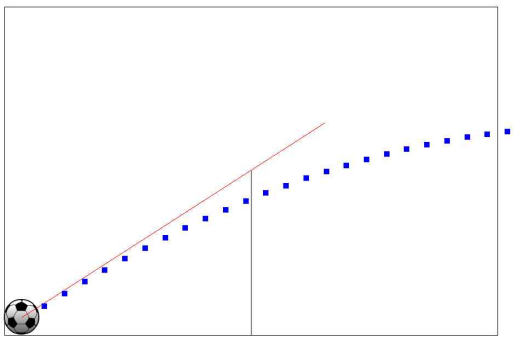
- 마찰력 : 마찰 계수 * 질량 * -1 * 중력 가속도
- 마찰가속도 : 마찰력/질량

만약 공이 바닥에서 오른쪽으로 움직이는 상태라면(v_[0] > 0), v_[0](x축 방향 속도) 에다가 (마찰가속도 * delta time)을 계속 빼서 공의 속도를 줄여 정지하게 하였다. 반대로, 공이 왼쪽으로 움직이는 상태라면 계속 더하였다.

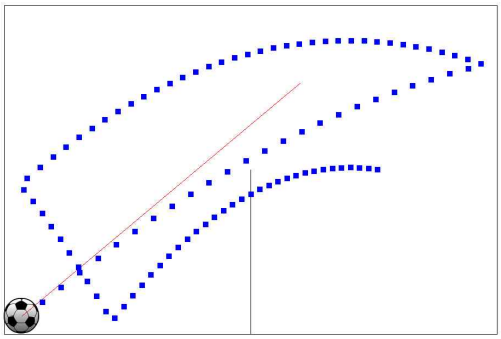
또한, 속도의 부호가 바뀌면 공이 멈추지 않고 계속 움직이기 때문에, 부호가 변하지 않도록 if문으로 조절하였다.

3. 요구조건 3 - 예측 경로 표시

먼저, Ball.h에 속도 반환 함수를 추가하였다. Projectile.cpp의 Render()에서 예측 경로를 계산할 때, 공의 초기 속도를 발사하는 힘으로 나누어 계산했다. 실제로 공을 발사할 때, 발사 힘을 질량으로 나눠 속도로 변환하기 때문이다. 예측 경로를 계산하는 반복문에서 g_timestep_s를 사용하여 예상 위치, 속도를 업데이트 했다. 중력 가속도 값은 Room 클래스에 정의된 것을 사용하였다.

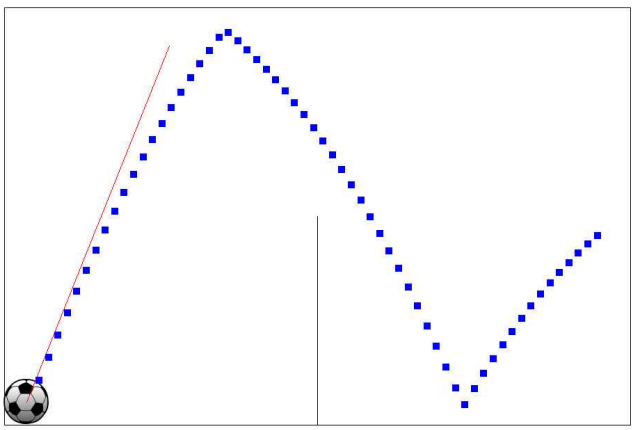


초기에 구현한 예측 경로는 우측 그림과 같다. 공이 벽에 부딪혀 튕겨나갈 때를 고려하지 않고 벽을 뚫고 나가는 모습이다. 따라서 땅, 천장, 왼쪽 벽, 오른쪽 벽과의 충돌 검사를 진행하고, 반사 계산을 추가하였다. 공이 벽과 부딪히면 위치를 조정하고 속도를 반사시켰다. 이때, 탄성 계수(coeff_of_restitution)을 적용하여 속도를 줄여 공의 모습이 자연스럽게 보이도록 했다.



그러자 우측 그림과 같이 공의 실제 경로와 예측 경로가 완벽히 일치하는 모습을 보였다. 그러나, 경로가 너무 길게 표시되는 문제가 있어 예측 경로는 공이 발사되고 1초 후까지만 표시되도록 수정했다. Render()에서 while 루프를 사용하여 흐른 시간이 1초보다 작을동안만의 경로를 표시하도록 했다. 각 반복에서 흐른 시간(time_elapsed)에 g_timestep_s를 더하여 경과 시간을 계속 갱신했다.

구현 완료한 결과는 아래 그림과 같다.



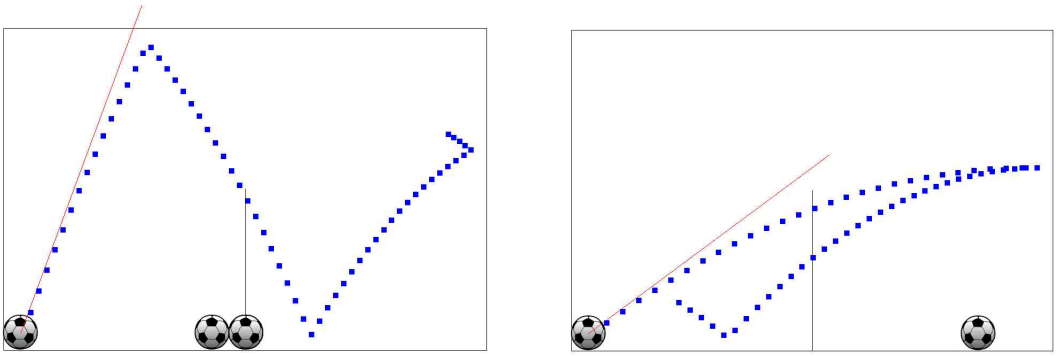
4. 요구조건 4 - 네트와 충돌 구현

네트와 공의 충돌 구현은 강의 자료 중 Line Segmentation으로 구현하였다. 네트를 직선 세그먼트로 설정하고, 충돌 검사를 한 뒤, 공이 네트에 충돌하면 속도를 반사하는 방식으로 설계하였다.

ProjectileGame.cpp의 Update()에서 충돌 검사 함수와 반응함수를 구현했다. 충돌 검사 함수에서는 공의 x좌표와 네트의 x좌표가 겹치는지와 y좌표가 네트의 높이보다 낮은지 체크하였다. 충돌 반응 함수에서는 공이 반사되면 속도가 줄어야 하므로, (-1 * 공의 원래 속도 * 탄성 계수) 하여 충돌 이후의 속도를 구했다. 처음에는 y 속도에도 이를 똑같이 적용해야 하는 줄 알았는데, 그렇게 하자 공이 떨어지지 않고 공중에 계속 떠있었다. 공은 y축에서 중력에 의해 움직이기 때문에, x 속도에만 적용하는 것이 옳은 것이다.

네트가 구현이 되었지만, 세 가지의 문제가 있었다.

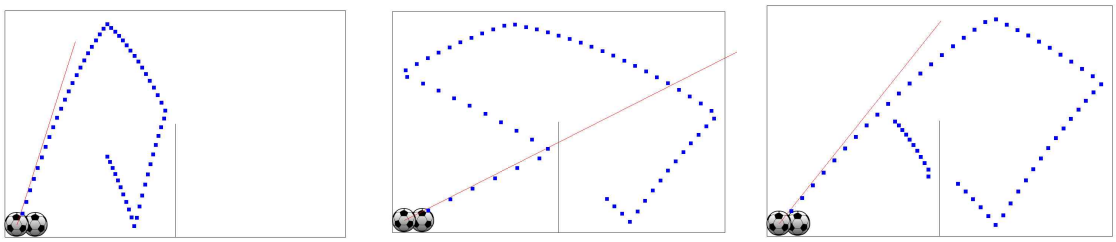
1) 공이 천장에서 튕긴 후에 네트 쪽의 바닥으로 떨어지면 네트를 뚫고 바닥에 착지한다.
2) 공이 네트 꼭지점에서 버벅이는 듯한 모습으로 이상하게 튕겨진다.
3) 예측 경로가 벽의 튕김까지 예측을 하지 못한다.



문제 해결 전 모습

1) 해결 방법 : 충돌 반응 함수에서 공이 무조건 네트 바깥으로 위치하도록 좌표를 고정시켰다. 그러자 자연스레 2번 문제도 해결이 되었다.

3) 해결 방법 : 예측 경로를 위한 충돌 검사와 반응 함수를 합친 예측 함수를 만들고, Render()에서 예상 경로의 네트 충돌을 구현하였다. 그러자 네트 기준으로 왼쪽 영역에서는 예측 경로가 잘 맞는데, 오른쪽 네트에 충돌했을 때의 예측 경로가 네트를 뚫고 왼쪽으로 가는 문제가 생겼다. (아래 우측 그림)



5. 미해결 부분

네트 우측 충돌에 오류가 생기는 것으로 보아 예측 경로 상의 공의 반사 속도와 위치가 제대로 계산되지 않은 것 같아 함수를 여러 번 수정해보았지만 아직 버그가 해결되지 않았다.