DX\_DOOM Report

B677011 방찬웅

1. 개발 배경

Doom이라는 게임은 제 생각에 FPS 게임의 최초 선구자라고 할 수 있고, 위대한 게임이라고 생각되기에 DirectX를 사용하여 이 게임과 비슷하게 만들고자 하였습니다.

1. 게임 시퀀스

Title Scene -> Main Stage -> Clear Scene or Fail Scene

1. 조작

WSAD : 기본 움직임

Left Click : 총 발사

1. 기능

FSM을 사용한 적 AI 구현

충돌 감지 (총알, 적, 벽)

A\* 알고리즘 사용을 통한 적 길 찾기 구현

객체 메모리 풀과 빈칸 리스트 기법을 사용하여 BulletPool이 Bullet을 관리 -> 메모리 단편화 현상 방지 및 속도 향상

Animation -> Billboarding을 응용하여 2D 텍스쳐를 각도에 따라 실시간으로 바꿔줄 수 있도록 함

Headbob 효과 (플레이어가 실제 걷는 듯한 느낌을 준다)

다양한 SFX (Hit, Shoot, music 등)

1. 시각 효과

2D UI / Text / Billboarding / Blending / Multi-Texturing / Normal Mapping / Skybox / Particles / Lighting

1. 어려웠던 점
2. 적과의 각도에 따라 텍스쳐를 변경하는 부분을 위해 벡터 공부를 추가적으로 진행함.
3. Navmesh

Navmesh를 구현하기 위해 네비메쉬 클래스에서 노드들을 바닥에 깔고 노드들에 대한 충돌검사를 Navmesh와 하고 갈수 있는지 없는지 값들을 0,1로 저장하여 A\* 클래스에서 경로 찾기를 실행하여 최단 경로를 생성할 계획을 했었다. 갈 수 있는 노드는 초록색, 갈수 없는 노드는 빨간색으로 실시간 렌더링을 하고 싶었다.

시도했던 Map의 노드 개수는 2만개 이상이었다.

시도 1, 노드 삼각뿔 모델 렌더링시 FPS 20까지 떨어짐

시도 2, 노드 삼각형 렌더링시 FPS 29까지 떨어짐, 삼각형의 Texture Update시 1FPS

시도 3, 노드 수직선 렌더링시 FPS 28, 수직선 Texture Update시 1FPS

시도 4, 노드 수직한 벡터만 사용 (렌더링은 불가), 삼각형 충돌 테스트 실행조차 안됨

시도 5, 네비메쉬를 모델이 아닌 Text File을 사용하여 갈 수 있는 곳과 없는 곳을 나눔, 최적의 길은 찾았다 하지만 길을 찾는 순간 프레임이 확 저하됨, DeltaTime을 이용하여 Update를 띄엄띄엄 하려했지만 찾는 순간에는 역시 프레임 저하가 일어남.

시도 5(문제 원인 및 해결)

원인 파악 중 A\*의 시간 복잡도는 Heuristic 함수에 따라 결정된다는 논문을 발견하여, Hueristic 함수를 다시 체크함.

Hueristic 함수에 f(h)를 endpoint와 endpoint의 거리로 잘못 설정하였음. 다시 f(h)를 endpoint - startpoint의 거리로 변경 하니 프레임 저하 일어나지 않음.

플레이어가 경로를 이탈해 경로를 찾지 못할 시에 프레임이 1로 확 떨어지는 현상 발생 -> 지정된 범위 밖으로 플레이어가 벗어나지 못하도록 설정 -> Bounding Box

여러 개의 적을 배치했을 때 길 찾는 시간이 너무 커 프레임 저하가 심하므로 노드 개수를 1/4로 줄이고, Stage를 작게 나누어 Stage 진행 시 그 안에 있는 적만 Update와 Render가 일어나도록 함. 하지만 아직도 먼 곳의 경로를 찾을 때엔 느려지는 점이 있음.

1. 벽과의 충돌

플레이어에게 Bounding Box를 설치해 벽과 충돌을 감지하려 하였으나 Camera는 항상 회전할 수 있기 때문에 앞, 뒤, 왼쪽, 오른쪽이 지역 좌표계와 월드 좌표계의 차이가 있어서 다른 방법을 생각해보았다. 카메라의 Forward Vector와 Right Vector를 이용하여 사방으로 Ray를 쏘고 그 거리를 계산하여 벽과의 충돌을 감지하는 방법을 사용하였다.

벽과의 충돌을 감지를 위해 Bounding Box를 배치하였다. 3개의 박스를 놓고 Collision Test를 진행하였을 때는 문제없이 충돌 처리가 되었다. 하지만 Bounding Box가 많아질수록 충돌 검사가 제대로 되지 않고 벽을 뚫었다. Delta Time 사이에 감지를 하지 못하고 뚫은 것 같다. 이를 해결하려면 충돌 감지 방식을 사용해야 할 것 같다..

1. 총알 관리

총알은 쏠 때 생성되었다가 사라지는 객체 이기 때문에 많이 쏘면 쏠수록 메모리에 빈공간이 생길 것을 우려하여 메모리 풀 기법을 찾아보았다. 게임 프로그래밍 패턴 책의 메모리풀 기법을 참고하여 구현하였고, 빈칸 리스트 기법을 사용하여 사용이 다 끝난 객체는 빈칸 리스트로 돌려주어 새로 생성할 시 바로 꺼내 올 수 있도록 하였다.

1. Release Mode

Debug 모드로만 코딩을 해본 사람으로써, 성능을 위해 Release 모드로 돌려보고 싶었지만, 에러가 굉장히 많이 떴다. 그 이유론 프로젝트 세팅을 디버그 모드와 같게 해주어야 하고, 없는 라이브러리가 있다면 추가를 해줘야 하는 것을 알게 되었다. 하지만 이 프로젝트는 중간에 Release 모드를 체크하며 한 것이 아니기 때문에 모든 에러를 해결하고 프로그램을 돌려봤지만 Release 모드로 실행을 할 수 없다고 나왔다. 너무 많은 것을 하고 멀리 왔기에 지금 Release 모드를 위한 디버깅은 불가능하기 때문에 앞으로의 프로젝트에선 중간에 Release 모드를 돌려서 체크 헤야겠다.

1. 수업 후기

학기 전에 정말 들을까 말까 고민을 많이 했던 수업이었습니다. 저는 졸업 작품을 언리얼 엔진으로 진행하다 보니 한 학기에 언리얼과 DirectX 포트폴리오 두가지를 동시에 진행할 수 있을까 많이 고민했지만 도전하였습니다. 학기 초반부터 중후반까지 언리얼 졸업 작품을 마무리하고, 늦게 제대로 시작한 DirectX 프로젝트라 시간이 조금 더 있었으면 하는 아쉬움이 있습니다. 학기 중 매일이 너무 힘들었지만 마치고 나니 보람차고, 포트폴리오 하나를 했다는 뿌듯함이 기분 좋은 것 같습니다. 수업에서 DirectX를 다루는 것이 중점이고 좋은 점이지만, 저는 수업에서 게임의 기본적인 구현 방법, 기초 등을 배운 것이 굉장히 좋았습니다. 상용 게임은 어떤 방식을 사용하고 있고, 제가 평소에 궁금했던 부분을 이론적으로 알 수 있게 되는 좋은 시간이었던 것 같습니다. 이 수업을 소수에 인원만 듣는 다니 마음이 아픕니다. 1년 동안 고생 많으셨습니다. 감사합니다.