3D게임 프로그래밍

과제3

게임공학과 3학년 2020182028 이규원

목차

1. **기존 구현 내용**

* **3D형태로 보이는 start 구현.**
* **미사일 발사 구현.**
* **플레이어 쉴드 발동 구현**

1. **추가 구현 내용**

* **배경 구현**
* **적 이동 AI 개선**

1. **조작법**
2. **기존 구현 내용**

* **3D형태로 보이는 start 구현.**

스크린샷, 일렉트릭 블루이(가) 표시된 사진

자동 생성된 설명

우선 Scene에 bool 변수 isMainGame를 넣어, Render, Animate에서 bool 변수에 따라 시작 화면, 인게임 화면을 구분 했다.

Framework에서 space바를 입력 받을 시 시작 화면에서 인게임으로 들어갈 수 있게 구현 했으며, 초반에 갑자기 죽을 수 있으니, elapsedTime으로 시간을 더해 게임 시작 후, 1초 동안 무적이 될 수 있도록 구현 했다.

* **미사일 발사 구현.**

스크린샷이(가) 표시된 사진

자동 생성된 설명

우선 게임 구현을 위해, Scene의 BuildObjects에서 플레이어, 적기, 미사일, 쉴드 에셋을 불러오고, 각각의 오브젝트들을 생성해 메쉬를 넣어 생성했다. 이 때, 메시와 맞지 않는 부분은 SetScale함수를 만들어 메시 크기를 조절했다.

가장 먼저 플레이어를 추가시키기 위해, InputPlayer로 플레이어를 생성하고, Framework에서 불러 왔다.

그리고 각각 플레이어와 적의 미사일 발사를 위해 미사일 생성함수 CreateMissile, CreateEnemyMissile을 구현했다. 이 때, 미사일 메시가 자연스럽게 날라가기 위해 바라보는 방향 LookAt, 생성 위치 SetPosition, 이동 방향 SetMovingDirection을 만들어 구현했다.

정확한 충돌처리를 위해 기존에 사용하던 AABB 충돌 -> OBB 충돌로 바꿔 사용했다.

AnimateObjects에서 적기가 플레이어를 바라보고 사격할 수 있도록 LooktAt함수를 재사용 했고, 일정 시간마다 적미사일을 생산할 수 있도록 EnemyMissileDelay 변수를 만들어 주기적으로 생산했다. 또한, 프레임마다 바운딩박스를 계속 업데이트 해 정확한 충돌처리를 할 수 있도록 구현했다.

위에 만든 바운딩박스를 활용해 플레이어 미사일 – 적기, 적미사일 – 플레이어의 OBB충돌을 구현 했다.

플레이어 미사일과 적기가 충돌할 시 적기의 체력이 100에서 10씩 감소되며, 0이 되면 사망한다. 체력이 줄 때마다, 적기의 색상이 빨강색에서 검은색으로 점점 변하게 된다.

적미사일과 플레이어가 충돌할 시 위와 마찬가지로 체력이 100에서 10씩감소되며 0이 되면 사망한다.플레이어의 함선 색이 점점 빨간색으로 변하게 되며, 0이 될 시, 게임오버되어 프로그램이 종료된다.

* **플레이어 쉴드 발동 구현**

스크린샷, 팬이(가) 표시된 사진

자동 생성된 설명

위와 마찬가지로 메시를 불러와 구현했다. 쉴드애니메이션 함수에 Rotate함수를 넣어 회전하게 만들어 역동감을 부여했다.

GameFramework에서 ‘z’키를 눌러 활성/비활성화 할 수 있도록 구현했다.

적기 미사일 – 플레이어 충돌 시 쉴드가 활성화 되어 있으면, 미사일은 사라지되, 플레이어에게 타격은 가지 않도록 구현했다.

1. **추가 구현 내용**

* **맵 구현**

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

기존 산의 terrain.obj로 존재하던 메쉬를 사용할 수 있도록, 엔진 유니티의 변환을 통해 하이트맵의 높이를 유지한 체로 terrain.txt로 변환했다.

BuildObjects에서 메쉬를 생성하고 SetScale로 충분한 만큼 크기를 늘려주고, SetPosition을 통해 플레이어, 적기보다 더 낮은 위치에 존재하도록 위치를 낮췄다. Rotate를 통해 이상하게 돌아가있던 회전각도 맞췄다.

* **적 이동 AI 개선**

스크린샷, 만화 영화, 애니메이션, 그래픽 디자인이(가) 표시된 사진

자동 생성된 설명

기존에 좌우로 왔다 갔다 했던 적들의 AI를 플레이어를 쫓아도록 개선했다.

적이 플레이어를 바라보고 쫓아올 수 있도록 GetPosition을 통해 위치를 구하고, Vector3::Subtract 함수를 이용해 플레이어를 향하는 방향을 구하고, 정규화 시켜 적이 플레이어를 자연스럽게 따라갈 수 있도록 구현했다.

1. **조작법**

이동 - wasd

무기 발사 – space

게임 시작 - space

쉴드 활성화 – Z키

캐릭터 시점 이동 – 마우스 왼쪽 드래그

카메라 회전 – 마우스 오른쪽 드래그

게임 종료 – esc