3D게임2 프로그래밍

과제1

게임공학과 3학년 2020182028 이규원

목차

1. **과제에 대한 목표 및 가정**
2. **구현 내용**
3. **조작법**
4. **과제에 대한 목표 및 가정**
5. **3D형태로 보이는 start 구현.**

START메뉴를 구현하기 위해 기존의 CExplosiveObject를 이용해 pCubeMesh, 육면체들로 START 글자를 만들고, 플레이어가 미사일을 발사해 Start 메시와 충돌 시 게임 시작이 되게끔 한다.

게임 시작 후, START 오브젝트와 플레이어 기체만 화면에 나올 수 있도록 만들고, 충돌 후에 터지는 이펙트 후, START 오브젝트를 비활성화 하고, 메인 게임을 구현할 수 있도록 한다.

플레이어가 적 미사일에 맞아 격추 시, 다시 START 화면으로 돌아오도록 한다.

1. **적기의 미사일 발사 구현.**

적의 미사일 발사를 구현하기 위해, 플레이어가 사용하고 있는, 미사일 발사를 적기에도 구현한다.

적기가 가만히 서서 쏘면 게임이 너무 쉽고 단조로워 지고, 너무 무작위성으로 이리 저리 날라다니면 플레이어 입장에서도 미사일 회피에 있어 너무 불합리하게 느껴질 수 있다. 그래서 좌우, 상하로 규칙성 있게 움직이면서, 정면으로 미사일을 발사하게 해, 플레이어로서 하여금 너무 단조롭지 않으면서 동시에 어느정도 컨트롤에 신경 쓸 수 있도록 한다.

위에 서술한 내용과 같이 플레이어와 적 미사일이 충돌 시 첫 화면으로 돌아갈 수 있게 한다.

1. **‘z’키를 누를 시 플레이어에게 쉴드 발동 구현**

z키를 누르면 플레이어 중심을 기준으로 육면체를 그려 쉴드가 생성됐음을 보여준다.

쉴드의 지속시간은 5초, 쿨타임은 쉴드가 꺼진 후 5초로 설정한다. 쉴드를 이미 발동중이거나, 쿨타임 중이면 쉴드 키를 눌러도 발동되지 않는다.

쉴드 발동 시 쉴드 바운딩 박스와 적 미사일이 충돌하게 되면, 적 미사일은 그 위치에서 소멸하게 한다.

1. **구현 내용 & 추가적으로 구현한 내용**

* 시작 모드

게임 실행 시 초기 모드를 설정하기 위해, StartMode에서 enum으로 설정했다. 초기 모드에 따라 메인 화면, 인 게임 화면으로 진입할 수 있다.

기본적으로 메인 화면에 진입해 조작키를 알려주고, 진입할 수 있다.

ESC키를 누르면 게임을 종료할 수 있고, 엔터를 누르면 메인 화면에서 인 게임 화면으로 전환된다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

* 인 게임 모드

인 게임 모드로 전환되면 게임을 시작하면서 필요한 객체를 Start 함수에서 Scene에 추가 한다.

Scene.cpp는 카메라, 캐릭터 제어, 적, 지형 객체 등을 초기화해 모든 객체 업데이트와 렌더링을 다 담당하는 프레임워크다.

KeyboardController, MouseMotionController, MouseController 함수를 통해 플레이어의 키보드, 마우스 입력을 scene에 전달하는 역할을 한다.

야외, 사람, 하늘, 잡기이(가) 표시된 사진

자동 생성된 설명

* 플레이어

Control.h 파일에서 플레이어인 헬리콥터 클래스를 관리한다.

자료구조 : 플레이어의 위치, 크기, 회전율, 객체의 충돌 범위, 헬리콥터의 방향 벡터를 포함하고 있다.

InputKey : 키보드 입력에 따라 MoveForward, MoveBackward, MoveLeft, MoveRight 변수를 설정하여 객체 이동 방향을 지정한다.

InputMouseMotion : 플레이어가 클릭하면 마우스를 감추고 화면을 회전할 수 있게 구현했다.

float cxDelta = (float)(mouse.CurrentPosition().x - MotionPosition.x) / 5.0f;

식으로 마우스의 현재 위치와 이전 위치간의 차이를 구해 회전 변화의 방향을 결정한다.

Update : 플레이 동안 갱신되는 값들을 처리하기 위한 값으로, 플레이어가 바라보는 방향으로 입력한 방향키에 따라 이동할 수 있도록 계산했다.

Position.x += Vec.Look.x \* FT \* 40;

Position.y += Vec.Look.y \* FT \* 40;

Position.z += Vec.Look.z \* FT \* 40;

바라보는 방향에 따라 거리를 구하기 위해 벡터의 룩벡터를 통해 구현 했다.

Render : InitMatrix 및 Transform 함수를 사용해 헬리콥터의 몸통의 크기, 이동, 회전을 렌더링 할 수 있게 구현했다.

헬기의 머리는 헬기 몸통의 변환을 그대로 가져와 몸통보다 위로 움직이고, 회전시켜 마치 헬기의 날개가 돌아가 나는 것처럼 구현했다.

추가로, 스카이 박스 메시와 텍스처를 통해 스카이 박스를 구현했지만, 세 면이 어둡게 보이는 버그가 있었고 아직 문제 원인 파악을 하지 못해 해결하진 못했다.

바운드 스페어를 OOBB로 구현해 Scene에 등록되어 있는 enemy의 바운드 스페어의 위치와 범위를 읽어와 충돌 검사를 하게 시켰고 충돌 시 플레이어가 자동적으로 뒤로 가게끔 구현했다.

* 적

Enemy.h 파일에서 적의 클래스를 관리한다.

기본적으로 플레이어와 구성과 마찬가지로 update를 통해 행동을 갱신하며, 위치에 따라 해당 위치에 렌더링 된다.

적 객체는 따로 입력을 받아 움직이지 않고 자동적으로 좌우로 움직이게 구현했다.

* 카메라

CameraController.h는 게임 내 플레이어의 카메라를 제어하는 클래스를 정의한다.

카메라의 회전을 부드럽게 하기 위해 lerp 함수를 사용 했다.

CamRotation.x = std::lerp(CamRotation.x, DestCamRotation.x, FT \* 15); 식을 통해 현재 카메라 각도와 목표각도를 선형보간을 통해 부드럽게 맞춰 자연스럽게 구현했다.

CameraUtil.cpp에서 카메라의 이동, 회전, 추적, 투영 설정 드을 정의하는 camera 클래스를 구현한다.