**3D 게임프로그래밍 1**

**과제 01 설명 문서**

김태순 2021182009

# • 프로젝트의 개요

해당 과제는 소프트웨어 렌더러를 기반으로 하되 기존 프레임워크 중심의 입력처리를 씬 중심의 입력처리 및 업데이트로 변경하고 기존 클래스들을 기반으로 하여 과제에 필요한 클래스들을 설계하고 제작하였다.

# • 프로젝트의 구조

GameFramework에서 CSceneManager를 생성하고 해당 객체를 통해 현재 씬의 Animate, Render(이하 업데이트)를 호출 씬의 업데이트에서는 해당 씬에 존재하는 모든 객체들에 대하여 업데이트를 호출한다. 각 씬은 처음 생성될 때 BuildObjects를 실행하여 씬에 필요한 객체들을 생성하며 씬이 바뀌거나 프로그램을 종료할 때 ReleaseCurrentScene을 호출하여 씬 내부의 객체들과 씬을 삭제한다. 또한 입력 처리는 프레임워크의 OnProcessingWindowMessage를 통해 마우스 입력과 키보드 입력으로 나누어 각 씬에서 처리하도록 함수를 호출한다.

# • SceneManager

이 객체는 모든 씬을 하나의 객체에서 관리하고 변경 가능하도록 만들기 위해 셜계하고 제작하였다. SceneManager는 CScene의 포인터로 이용하여 현재 씬 객체를 저장하고 있고 각 씬은 CScene에서 상속받아 오버라이딩한 업데이트(Animate,Render)함수들과 입력처리 함수들이 호출되어 각 씬의 내용을 실행한다.

씬을 변경할 때에는 우선 현재 씬의 종료 여부를 Animate에서 현재 씬의 IsFinished를 통해 bool값을 리턴받아 확인하고 이것이 true면 현재 씬에 저장되어 있는 다음 씬의 번호(int값)를 GetNextSceneID를 통해 확인하여 ChangeScene을 호출한다. 이때 각 번호에 해당하는 씬으로 SetCurrentScene을 호출하여 변경되고 이 과정에서 이전 씬의 객체들을 삭제하고 메모리를 해제하는 ReleaseObjects를 호출하고 현재 씬의 메모리도 해제한 이후에 변경될 씬을 새로 생성하고 BuildObjects를 통해 씬의 오브젝트들을 생성한다.

이 객체는 GameFrameWork의 소멸자 즉 게임이 종료될 때 삭제되며 이때 ReleaseCurrentScene을 통해 현재 씬 또한 해제된다.

# • SceneManager

* **CStartScene**: 시작 화면을 관리하기 위해 제작. 마우스 입력과 고정된 카메라를 통해 UI처럼 작동하게 설계.
* **CMenuScene**:StartScene과 동일하지만 더 많은 텍스트 객체를 관리하도록 설계.
* **CLevel1**: 롤러코스터 레벨을 관리하기 위해 설계.
* **CLevel2**: 탱크 레벨을 관리하기 위해 설계.
* **CTextCharacterObject**: 텍스트 오브젝트의 각 글자를 관리 및 표현하기 위해 설계.
* **CTextObject**: 텍스트 오브젝트를 관리 및 표현하기 위해 설계. 문자열을 인자로 하여 오브젝트를 생성할 수 있도록 설계하였다.
* **CRailObject/CRailMesh**: 롤러코스터의 경로를 표시하기 위해 설계.
* **RailSegment**: 실제 롤러코스터의 경로를 저장하는 구조체로 설계. 캣멀-롬 스플라인을 이용해 생성.
* **CTankPlayer**: 탱크플레이어를 관리 및 표현하기 위해 설계.
* **CTankEnemy**:

# • 구현된 알고리즘 및 핵심 처리 방식

- 씬 매니저 구조: CSceneManager에서 씬 전환 통제

# • 추가 구현 사항

-

- MenuScene에서는 다중 텍스트 객체를 일정 간격으로 배치