|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 2주차 | **기간** | 7.03~7.10 | **지도교수** | (서명) |
| 이번주 한일 요약 | 언리얼컨테이너 공부/ 캐릭터 기본적인 로코모션구현, 공격 구현  /언리얼엔진 Lanch | | | | |

<상세 수행내용>

엔진에서 제공하는 컨테이너들을 공부했다. Core/container 폴더 안에 소스코드들이 있기는 했지만 기존에 c++에 있는 컨테이너랑비슷한 거같아서 따로 보지는 찾아서확인하지는 않았습니다. 자료들을 찾아보니 전체적으로 언리얼 엔진에 더욱 알맞게 만들어진 컨테이너들인것 같습니다.

그리고 엔진을 활용하여 유튜브 영상들을 보면서 캐릭터의 기본적인 로코모션을 구현하고 총 공격과 칼 공격을 만들어 보았습니다. 유튜브 영상을 보면서 따라해서 어떤식으로 엔진에서 코드를 작성해서 게임을 만들어야하는지 알아가고 있습니다.

언리얼엔진에 있는 Launch와 관련된 코드들을 공부했습니다. 언리얼 도큐먼트에서는 Lanch를 엔진의 진입점이라 소개합니다. 해당 폴더에 들어가 여러 코드들이 있는것을 봤는데 중요한 부분은 일단 Launch.cpp에있는 GuardedMain 함수 입니다. 이 함수에서는 엔진의 많은 초기화를 진행하고 어느시점에 종료 요청이 들어올때까지 계속 루프로 돌아갑니다. GuardedMain 함수에 메인루프는 LaunchEngineLoop.CPP에 있는 FEngineLoop클래스에 구현되어 있고 초기화 단계도 FEngineLoop클래스에 PreInit이라고 구현되어있습니다. PreInit함수에서 저수준 엔진의 모듈들이 초기화 됩니다. 그래서 필수적인 시스템이 초기화되고, 필수적인 타입들이 정의 되는것입니다.

별개의 이야기 이지만 리플렉션 시스템은 UENUM(), UCLASS(), USTRUCT(), UFUNCTION(), UPROPERTY() 등 매크로 함수를 위에 붙여 놓은것을 사용해서 각각 클래스의 CDO를 구성합니다.

CDO는 그 클래스의 디폴트 상태를 나중에 사용 하기위에 저장한것 입니다. 덕분에 나중에 오브젝트를 매번 생성하지 않고 사용할수 있습니다.

Launch와 관련된 코드를 볼때

참고한 영상 : https://www.youtube.com/watch?v=IaU2Hue-ApI&t=159s

참고한 문서 : https://forums.unrealengine.com/t/how-does-the-game-loop-actually-work/90508

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** | 엔진 코드를 보는데 생각보다 너무 방대하다 | | |
| **해결방안** | 일단은 언리얼 포럼에서 자료를 찾고 api레퍼런스를 참고해야할거  같다 | | |
| **다음주차** | 3주차 | **다음기간** | 7.11~7.18 |
| **다음주 할일** | 로코모션하고 공격모션등을 구현했던것을 사용해서 충돌판정과 여러 기믹들 만들어보기/ MeshUtilities 와 KismetProceduralMesh 코드 보기 | | |
| **지도 교수**  **Comment** |  | | |