< 3D 게임1 - 과제 01 >

2018180035 게임공학과 장재문

1. **과제에 대한 목표**

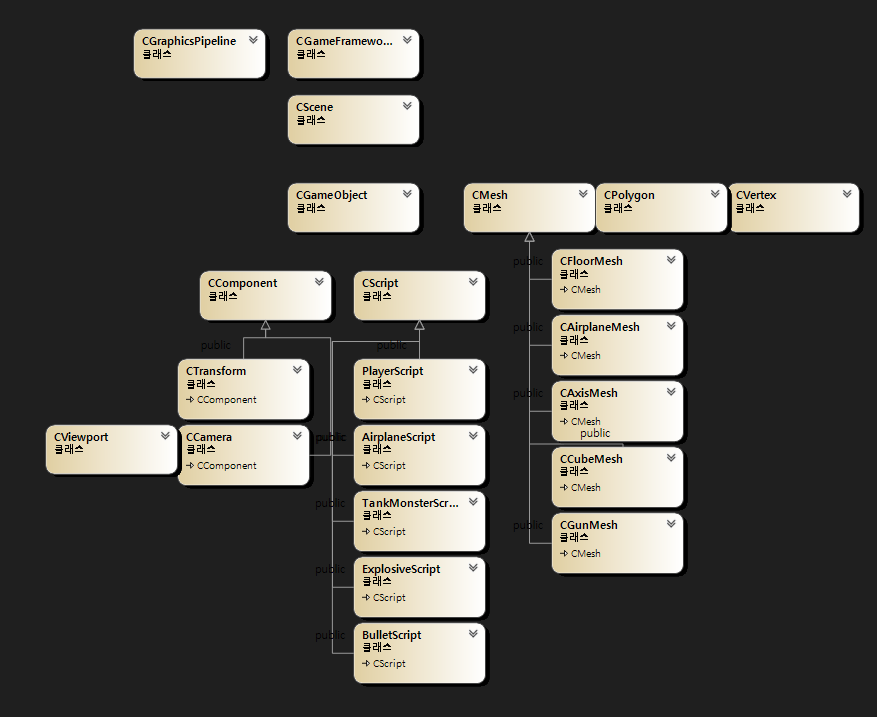
* 목표 : 교수님 프로젝트 이해하기, 탱크 게임 만들기
* 이번 과제를 하면서 저는 교수님의 LabProject의 동작과정, 구조를 완벽하게 이해하는 것에 중점을 두었습니다. 탱크 게임을 만드는 것은 결국 교수님의 프로젝트 구조를 완벽하게 이해해야 가능하다고 생각했습니다. 그래서 교수님의 프로젝트를 더욱 자세하게 이해하고자 노력했습니다. 프로젝트 시작 초반에는 탱크 게임을 만든다는 목적보다는 프로젝트를 이해하고 제가 직접 구조를 다시 짜보는 것에 시간을 투자했습니다. 렌더링 과정과 그에 맞는 행렬변환 적용방법, 그리고 Scene 과 GameObject의 관계, Graphicspipeline 의 역할, FrameWork 동작 등 교수님의 프로젝트 구조와 기능에 대해 공부했습니다. 이를 토대로 엔진을 저만의 구조로 다시 만들었고 이를 바탕으로 탱크 게임을 제작했습니다.

1. **실행결과 와 조작법**

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ( Jump 키는 구현하지 못했습니다. ) | |  |  | | --- | --- | | W | 현재 카메라가 바라보는 방향이 나아가는 방향이 되어 앞으로 이동 | | A | 현재 카메라가 바라보는 방향기준 좌측 방향으로 이동 | | S | 현재 카메라가 바라보는 방향기준  뒤 방향으로 이동 | | D | 현재 카메라가 바라보는 방향기준  우측 방향으로 이동 | | Q | 포를 시계반대방향으로 회전 - Yaw | | E | 포를 시계방향으로 회전 - Yaw | | R | 포를 위 방향으로 회전 - pitch | | F | 포를 아래 방향으로 회전 - pitch | | RBTN | 카메라를 회전한다 ( PRESSED )  ( 플레이어 기준 3인칭 시점 ) | | LBTN | 플레이어가 포탄을 발포한다. | |
| 게임 실행   * 마우스 오른쪽 버튼을 클릭하면서 카메라를 회전하며 Player 를 WASD 키로 이동한다. 마우스 왼쪽 버튼을 클릭하면서 포탄을 발포한다. 몬스터가 30대를 맞거나 Player 가 30대를 맞아 죽으면 게임이 종료된다. Player 의 포탄은 게임월드에 랜덤으로 배치된 박스에 닿아도 폭발한다. 몬스터의 포구는 Player 를 향해서 2초 마다 발포한다. | 빨간색 Tank Monster 는 좌우로 움직이며 Player 를 향해 발포한다. 맨 처음 Airplane 이 CatmullRomSpline 곡선을 이용한 움직임을 보인다.  Player 카메라는 3인칭으로 마우스 오른쪽 버튼을 통해서 회전한다. |

**변형한 구조 설명**

****

기존의 LabProject와 큰 구조는 같지만 기존에는 이동정보와 행렬이 각각 PlayerObject 안에 있다거나 CameraObject 안에 존재했습니다. 저는 이부분에 있는 이동정보만 뺴내서 Transform 클래스를 만들어서 관리했습니다. 또한 Camera 기능도 Component로 따로 빼서 관리했습니다. 그래서 GameObject에 Component를 붙이는 방식으로 Object를 생성 및 관리했습니다.

LabProject 코드를 분석하고 혼자서 만들어보면서 기존 구조를 조금 변형했습니다. 제가 만든 프로젝트의 큰 구조는 이렇습니다. CGameFramework는 하나의 Scene을 가지고 있습니다. 이 Scene은 여러 개의 GameObjects를 가지고 있습니다. GameObjects 는 각각 Component를 가지고 있습니다. Component 는 이동상태 정보를 저장하는 Transform 과 카메라의 기능을 수행하는 Camera 로 나뉩니다. 즉 GameFramework -> Scene -> GameObjects -> Component/Script 순으로 Update를 한 후 FinalUpdate로 행렬정보를 갱신 후 GraphicsPipeline 행렬을 이용해 Render 하게 됩니다.

|  |  |
| --- | --- |
|  | 기존 LabProject는 Animate를 한 후에 Render 하는 방식으로 진행이 됐었습니다. 저 또한 이 구조를 따라갔습니다.  **단,**   * Update( ) 일때는 이동정보 WorldPosition, WorldRotation 정보만 업데이트를 했습니다. * FinalUpdate( ) 일때 WorldPosition, WorldRotation 정보를 가지고 행렬정보를 업데이트했습니다. |

|  |
| --- |
| **Component – Transform [ 추가한 내용 ]** |
| |  |  | | --- | --- | |  | * **XMMatrixRotationX,Y,Z 를 이용한 회전행렬**   **계산 및 역행렬 이용**  RotationYawPitchRoll 로 한번에 MatRotate를 구하지  않고 XMMAtrixRoatationX,Y,Z 를 이용해서 행렬을  만들어보았습니다. 또한 이때 회전행렬의 역행렬을  미리 구해놓았습니다. 이 회전행렬의 역행렬은 부모의  회전행렬에 영향을 받고싶지 않은 상황이 있어서 구  해놓았습니다. |     WorldTransform 을 계산할 때 부모의 영향을 받는 상황에서 부모의 이동행렬만 영향을 받고 부모의 회전행렬의 영향은 받고 싶지 않았습니다. 그래서 회전행렬을 구할 때 역행렬을 미리 구해놓았습니다. 그리고 자식의 WorldTransform 을 계산할 때는 부모의 Rotation Inverse Matrix 를 이용했습니다. 부모의 매트릭스에 이 부모의 회전의 역행렬을 곱하면 회전이 적용안된 즉, 이동행렬만 적용된 부모의 월드행렬을 구할 수 있었습니다. |
| * **XMMatrixTranslation 를 이용해서 이동행렬을 계산해 보았습니다.**     41,42,43 에 직접 넣어도 똑같은 코드 입니다. 수업시간에 배운 내용을 적용해보고 싶어서 이렇게 구현해 보았습니다. 실제로 똑같이 잘 작동하는 것을 알 수 있었습니다. |
|  | |
| Rotation Matrix 와 Translation Matrix 를 따로 연산을 해서 값을 구한 뒤에 마지막에 곱해주는 방식으로 바꿨습니다. 저의 구조에서는 Update 일때 XMFLOAT3 인 Position 과 Rotation 이 갱신됩니다. 이를 FinalUpdate 에서 Matrix를 각각 구한후에 마지막에 ComputeWorldTransform()에서 이를 곱해주는 방식으로 Transform행렬과 World 행렬을 연산했습니다. 이때 교수님의 SRT 구조를 떠올리며 코딩했습니다. 처음에 잘 되지 않아서 시간을 많이 할애했는데 교수님이 말씀하신 SRT를 떠올리고 순서의 문제였다는 것을 깨달았습니다. 이 구조로 적용하는 것이 시간을 가장 많이 쓴 것 같습니다. 또한 큰 문제점은 Radian으로 회전행렬을 계산했어야 했는데 전부 Degree 로 계산을 했던 것이 큰 문제였습니다. | |

|  |  |  |
| --- | --- | --- |
| * KEY 클래스 추가  |  |  | | --- | --- | |  | 이 클래스는 이번 프레임에 어떤 키를 눌렀고, 그 키가 처음 눌린건지 아니면 이전에부터 눌려있던건지 떼어져있는건지 판단하는 class 입니다. | |
| |  |  | | --- | --- | |  |  | |
|  |
| GetAsyncKeyState 를 이용해서 KEY 정보를 저장하는 클래스입니다. 이전에 눌렸는지를 bool 로 체크하면서  TAP 인지 PRESSED인지 AWAY 인지 구합니다.  또한 마우스 정보도 활용해서 클라이언트 화면에서 x축 y축으로 얼마나 움직였는지 구했습니다. |
| 저의 게임에서는 콜백함수를 이용하지 않고 모든 키를 이 Key 클래스를 이용해서 계산했습니다. |

* GunMesh 를 제작해 GunObject 에 사용했습니다.
* CubeMesh는 중앙을 기준으로 사각형이 만들어지는데 GunMesh 는 무조건 z값이 0보다 크게 만들어서 Mesh 를 만들었습니다. 이걸 만든 이유는 CubeMesh 로 탱크에 포를 만들게 되면 터렛이 회전을 할 때 터렛의 중앙 부분을 뚫는 모습이 보였습니다. 이는 Transfrom 으로 해결하기에는 한계가 있다고 생각해서 Mesh 를 따로 만들었습니다.

|  |
| --- |
| * **CatmullRomSpline 을 이용한 이동을 구현했습니다.** |
| CatmullRomSpline 을 활용한 움직임을 구현해보고 싶었습니다. 그래서 이를 클래스로 만들었습니다. 탱크 게임에서는 처음에 게임이 시작될 때 Airplane Object 가 CatmullRomSpline 곡선을 그리면서 이동하는 것을 볼 수 있습니다. |
| 저장된 p0 p1 p2 p3 를 인자로 이용해 DirectX::XMVectorCatmullRom()함수를 이용해 위치값을 계산했습니다. 이를 넘겨주고 CatmullRomsPline 으로 움직이고 싶은 게임오브젝트의 Transform Component에 SetPosition 을 해주면서 이동했습니다. |

1. **자료구조**

|  |
| --- |
|  |
|  |
| 자료구조는 STL std::vector<> 자료구조와 배열 형식의 자료구조를 활용했습니다. Vector 와 배열은 메모리가 연속되어있어 접근하는데 있어 속도가 빠르다고 알고 있습니다. Std::array 를 쓰지 않고 std::vector 를 쓴 이유는 양 끝 동시접근이 가능하다고 해서 선택했습니다. 좀더 빠르게 메모리에 삽입,삭제 하는데에도 이점이 있어 vector 를 선택했습니다. |

1. **스크립트 추가**

|  |  |
| --- | --- |
|  | **PlayerScript**   * Key 클래스를 이용해서 키를 받고 이에 대한 움직임, 공격을 구현했습니다. * 카메라의 각도와, WorldDir 방향을 구해서 플레이어가 앞으로 나아가는 방향이 카메라가 바라보는 방향으로 설정했다. 이를 기준으로 좌우앞뒤를 움직이도록 설정해 움직일 때 자연스럽게 회전하도록 했습니다. |
|  | **ExplosiveScript**   * 기존에 LabProject 는 static 으로 CubeMesh를 공유해서 그림을 그리고 이동행렬만 바꿔서 그렸다. 이러한 기조를 바탕으로 ExplosiveScript를 갖는 GameObject의 메쉬를 공유 CubeMesh로 설정하고 ExPlosiveScript 에 있는 이동행렬배열들을 업데이트 해서 한번에 그릴 수 있도록 구현했다. 기존에 공유메쉬를 사용해서 그리는 구조를 그대로 가져갔다. |