졸업 작품 보고서

게임소프트웨어 학과

B793200

이지혁

**서론 – 시작하기에 앞서..**

지난학기 초에 졸업 작품을 정하면서 많은 고민이 있었습니다.

아직 배운 것이 많지 않다고 생각하는데 갑자기 졸업 작품을 정해야 한다고 생각하니 앞길이 막막하더군요.

사실 이 때까지만 해도 클라이언트로 갈 지 서버로 갈 지 조차 제대로 정하지 못해서 갈팡질팡 하고 있었습니다.

그러다가 마지막에 마음을 굳혀서 눈에 보이지 않는 실체가 없는 것들 보다는 실제로 눈으로 볼 수 있고 구현하면 결과로 그대로 나타나는 실체가 있는 것을 추구해보자 하여 클라이언트를 선택하였습니다.

플랫폼을 정하고 나니 어떤 엔진/SDK/API Library 로 구현해야할지 고민이 되었습니다.

Unity Engine 을 사용하자니 보다 프로그래밍 기술을 요구하는 쪽으로 가고 싶고, Unrial Engine 을 사용하자니 아직 한 번도 사용해보지 않은 엔진이라 낯설었습니다.

곰곰히 생각해보니 3학년 1학기 때 컴퓨터 그래픽스 과목에서 배웠던 Direct X 11 이 생각이 났습니다.

그래픽스 이론을 들으면서 Rendering Pipeline 에도 관심이 많았고 보다 Low Level 의 기술을 배우고 싶은 생각에 졸업작품 구현을 Direct X 11로 하기로 마음먹었습니다.

처음에는 자체 게임 엔진을 만들어보자는 생각이었습니다.

그러나 Windows API 에 대한 지식도 부족하고 Direct X 11 에 대한 지식도 많이 부족한 저에게는 장벽이 너무나도 컸습니다.

그래서 타협을 보아 그럼 그나마 게임 엔진 비슷하게 월드를 제작하고 그 안에 오브젝트를 배치할 수 있는 게임을 만들자 해서 생각해낸 것이 샌드박스형 게임이었습니다.

샌드박스형 게임하니까 떠오르는게 바로 Minecraft 였고, 또 최근에 Unrial store 에서 사서 플레이 했던 Satisfactory 가 생각났습니다.

그렇게 두 개의 게임에서 컨셉을 가져와서 만든 게임이 바로 저의 졸업 작품 ‘World Maker’ 입니다.

**구성도 1.**

**Diagram – 게임의 전체적인 흐름**

Stage 1 의 발전된 형태.  
Stage 1 에서는 단순히 자원이 스테이지 위에 뿌려져 있었다면, Stage 2 에서는 자원을 획득하려면 해당 스테이지에 존재하는 파괴 가능한 오브젝트들과 상호작용 하여 채취하여야 함.  
자원을 채취했으면 이후는 전과 마찬가지로 퀘스트 목표에서 지시하는 건물을 지으면 스테이지 클리어.

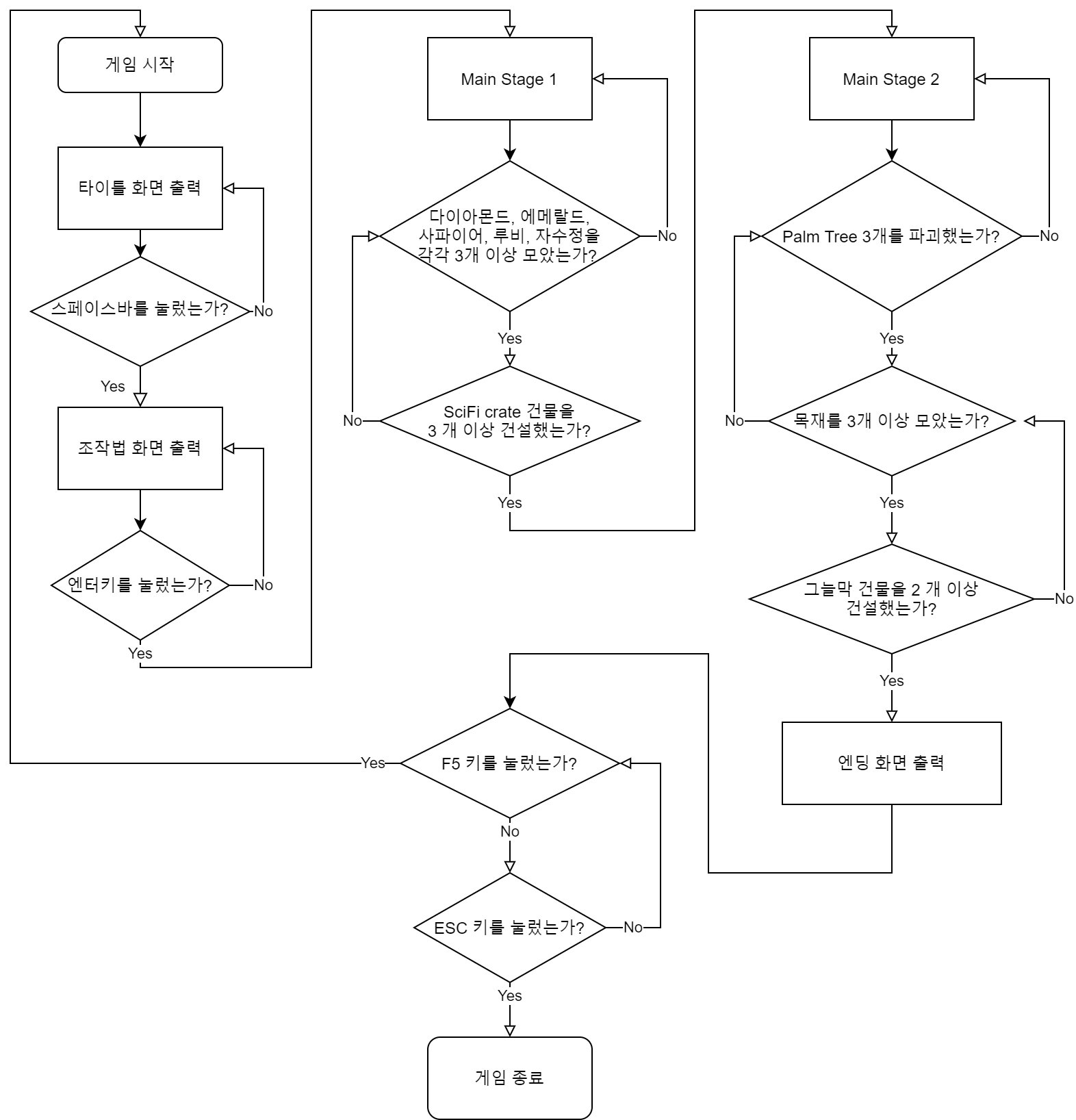
조작법 숙달을 목적으로 한 기본 튜토리얼.  
플레이어는 해당 스테이지에 흩어져 있는 자원을 획득하고 주어진 퀘스트 목표에 따라 건물을 지어 튜토리얼을 완료해야 함.

Stage 1, Stage 2 를 클리어 하면 나오는 화면.  
유저에게 튜토리얼을 모두 클리어 했다는 것을 ‘Tutorial Clear!’ 라는 문구를 통해 알려줌.

게임의 기본적인 조작법을 설명하는 화면으로 각각 키보드, 마우스로 나누어 조작법을 설명.  
Space 키를 누르면 다음 단계로 넘어감.

게임의 타이틀 ‘World Maker’ 를 보여주는 화면으로, Enter 키를 누르면 다음 단계로 넘어감.

**구성도 2.**

**Flowchart – 플로우차트**

**Implementation – 구현 내용**

* Multi scene control (Title, Controls, Main1, Main2, Ending)

Title Scene, Controls Scene, Ending Scene 은 PNG 포맷으로 제작된 이미지.

Bitmap 쉐이더를 구현했고 그것을 통해 Title 화면과 Controls 화면을 랜더함.

각 Scene 들은 기본적으로 1600 x 900 의 해상도를 갖고 있으며 창모드로 실행됨.

PNG 이미지로 제작된 Scene 들은 키 입력 (Space, Enter) 을 통해 다음 Scene 으로 진행하며 Main 1, Main 2, Ending Scene 은 각각의 조건을 만족해야만 다음 단계로 진행할 수 있음.

Main 1 에서 Main 2 로 넘어가는 조건은 Sci-Fi Crate 건물을 3 채 건설하는 것이며 Main 2 에서 Ending 으로 넘어가는 조건은 그늘막 건물을 2 채 건설하는 것임.

어느 Scene 에서든지 F5 버튼을 누르면 타이틀 화면으로 돌아갈 수 있고 게임의 진행 상태는 초기화됨.

어느 Scene 에서든지 ESC 버튼을 누르면 게임이 종료됨.

* Cube map 을 통한 Skybox 구현. (Background image)

Skybox 쉐이더를 구현하여 배경을 랜더함. (이하는 작동 방식)

플레이어 캐릭터를 중심으로 하는 구를 생성하고 캐릭터가 어디로 움직이든 항상 구의 중심에 자리잡게끔 캐릭터의 움직임에 따라 구를 이동시킴.

그리고 해당 구의 안쪽 면에 Cube map 텍스처를 받아와서 랜더함. (마치 방 안에 벽지를 붙이는 것처럼)

* OBJ parser, MD5 parser 구현.

정적 오브젝트 (Static objects) 들은 .obj 포맷을 사용하였으며, 동적 오브젝트 (Dynamic objects) 는 .md5 포맷을 사용하였음.

정적 오브젝트는 게임 배경을 조성하는데 쓰이는 오브젝트나 캐릭터와 상호작용할 수 있는 각종 오브젝트들에 사용되었음.

동적 오브젝트는 플레이어블 캐릭터의 구현에 사용됨.

각 포맷은 Direct X 11 에서 읽을 수 있게끔 parser 를 구현하였음.

OBJ 포맷은 v / vn / vt / g / s / f 의 파라매터로 구성되어 있으며 각각의 구성 요소는 vertex 좌표값, vertex 노멀값, texture UV 값, name, smoothing 그룹, face (하나의 면을 구성하는 값들을 총칭) 를 나타냄.

MD5 포맷은 md5mesh 와 md5anim 으로 나뉘어져 있으며, md5mesh 는 모델의 메쉬값(모델의 좌표값 등..) 을 가지고 있고, md5anim 은 모델의 스켈레톤 애니메이션에 쓰이는 여러 정보를 담고 있음. (Weight 값 등..)

각각의 Parser 는 각 포맷의 특정 key 값을 통해 인덱싱하여 파라매터를 찾아내고, Direct X 11 에서 읽을 수 있는 정보로 변환하고 정렬하여 buffer 에 저장해 쉐이더로 넘김.

* Multi lighting 구현.

‘World Maker’ 에서 사용하는 조명 타입은 두 가지이며, Directional Light 와 Spot Light 를 사용함.

두 조명을 함께 적용시키기 위하여 Light shader 를 HLSL 로 구현하였음.

Light shader 에 넘기는 버퍼의 파라매터에는 ambientColor, diffuseColor, lightDirection, specularPower, specularColor, pos, range, dir, cone, att, pading 이 있으며, ambientColor 부터 specularColor 까지는 Directional Light 구현을 위해 사용되고 pos 부터 paing 까지의 값은 Spot Light 를 구현하기 위해 사용됨.

* Visual Effects 구현.

‘World Maker’ 의 비쥬얼 이펙트는 Fire, Blue portal, Red portal 이 있으며, 이 이펙트들은 하나의 shader 로 구현하였음.

Effect shader 는 Multi Texturing 기법을 사용하였음.

3개의 각기 다른 텍스쳐를 정해진 시간 간격에 따라 빠르게 바꿔가며 랜더하여 불꽃이 일렁이고 포탈이 일렁이는 이펙트를 구현하였음.

빌보드 기법을 같이 적용시켜서 완성하려 하였으나 빌보드가 캐릭터 방향을 항상 바라보게끔 하는 작업에서 오류가 발생하여 포기하고 대신 여러장의 빌보드를 각각 다른 방향으로 겹쳐 어느 방향에서 보아도 어느정도는 이펙트가 보이게끔 구현함.

* 런타임 오브젝트 생성 / 제거 구현.

런타임 환경에서 오브젝트를 생성하고 제거하는 기능을 구현하였음.

이 기능은 본 게임에서 건물을 생성하고 해체하는 작업에 사용되는 기술임.

Render 함수 내에 True / False 값을 가지는 변수를 사용하여 오브젝트의 랜더링을 제어하였음.

예를들어 건물을 지을 수 있는 조건이 참이 되었을 때 마우스 왼쪽 버튼으로 Input 을 넣으면 특정 건물 랜더링을 제어하는 bool 변수가 참이 되어 오브젝트를 랜더하게됨. (건물 생성)

또한 해당 건물에 대고 마우스 오른쪽 버튼을 클릭하면 변수가 거짓이 되어 오브젝트를 랜더하지 않게 됨. (건물 해체)

* 캐릭터 애니메이션 구현.

캐릭터 애니메이션은 Skeleton Animation 으로 구현하였으며, md5anim 파일에 저장되있는 정보로 제어함.

Skeleton Animation 기법은 캐릭터가 움직일때마다 md5anim 에 저장되있는 애니메이션 프레임 대로 메쉬가 움직임.

각 애니메이션 프레임을 랜더할때마다 뼈대를 만들고 그 뼈대에 mesh 를 붙여서 캐릭터를 랜더함.

정해진 시간 간격에 따라 애니메이션 프레임 대로 메쉬를 움직이기만 해도 애니메이션은 구현할 수 있지만, 그렇게 되면 너무 어색하므로 각 애니메이션 프레임 사이의 보간 값을 계산하여 적용시켜 자연스러운 애니메이션을 연출함.

보간 계산에는 Slerp 라는 테크닉을 사용하였으며, DirectX 의 xnamath 라이브러리에 있는 XMQuaternionSlerp() 라는 함수를 사용하였음.

* WASD 키를 이용한 캐릭터 움직임 구현.

DirectX 의 Direct Input 라이브러리를 사용하여 구현하였음.

키보드의 입력값에 따라 각각 W 키를 눌렀을 때는 전진, S 는 후진, S 는 좌로 이동, D 는 우로 이동하게 만듦.

캐릭터 움직임은 Render 함수에서 캐릭터 모델의 좌표값을 버퍼를 통해 Shader 로 넘길 때 Direct Input 으로 받아온 키보드 키의 종류와 누르고 있던 시간에 따라 바꾸는 식으로 구현하였음.

또한 캐릭터가 바라보는 방향은 마우스로 조절하게끔 구현함. (마우스 입력도 Direct Input 으로 받아옴)

* 3인칭 카메라 워크 구현.

본 게임의 시점은 3인칭으로, 캐릭터 애니메이션을 보다 잘 보여주기 위해 해당 시점을 채택함.

WASD 키보드 값을 받아서 움직이는 캐릭터에 따라 카메라도 동기화시켜 움직이게 구현함.

여러 오류가 많이 발생하여 한정적으로 구현된 기술임. (위 아래 시선 움직임은 캐릭터를 따라가지 않고 공중에 고정되어 있음)

* Sound System 구현.

DirectX 의 Direct Sound 라이브러리를 사용하여 구현하였음.

Sound 는 Effect Sound 와 BGM (BackGround Music) 으로 구성됨.

Effect Sound 는 화면 전환 효과음, 자원 획득 효과음, 건물 건설 효과음, 건물 해체 효과음 이렇게 총 4 가지를 사용하였으며,  
BGM 은 타이틀&조작법 배경음악, 메인스테이지 배경음악, 엔딩 배경음악으로 3 가지의 음악을 사용하였음.

* UI System 구현 (Inventory, Interactive UI)

Bitmap Shader 를 통해 UI 를 구현하였음.

UI 는 Z-Buffer 상 가장 앞에 위치하게 설정하여 모든 오브젝트들 중에 가장 위쪽에 그려지도록 구현함.

또한 Alpha-blending 을 사용하여 PNG 이미지의 투명도를 살려서 랜더하였음. (반투명 UI 구현)

인벤토리의 경우 퀘스트에 필요한 아이템을 보여주는 하단 간략 아이템창은 구현하였으나, 실제로 아이템을 옮기고 버리고 할 수 있는 I 키를 눌러 열 수 있는 인벤토리는 제대로 구현하지 못하였음.

* 오브젝트 충돌 체크 구현 (Collision System)

AABB (Axis-Aligned Bounding Box) 를 이용한 충돌 체크를 구현하였음.

AABB 는 오브젝트를 감싸는 박스를 생성하는데 해당 박스의 노멀 벡터들이 World 상의 x, y, z 좌표와 일치함.

생성된 박스를 x, y, z 축으로 나누어 각각 비교하면서 겹치는 부분이 있나 체크하고 겹치는 부분이 하나라도 있으면 Collision True 를 반환하고 어느 부분도 겹치지 않으면 Collision Flase 를 반환하게 구현함.

본 게임에서는 플레이어의 캐릭터가 움직일때마다 Collsion 을 체크하여 통과할 수 없는 오브젝트에 부딪히면 더 이상 진행하지 못하고 제자리 걸음하도록 설정하였음.

또한 상호작용 가능한 오브젝트에 부딪히게 되면 아이템을 획득하게 되고 건물을 부수거나 부술 수 있는 오브젝트에 상호작용 할 때도 해당 오브젝트의 바운딩 박스에서 충돌을 감지하면 상호작용이 가능하도록 설계하였음. (Unity 의 Collider 와 유사함)

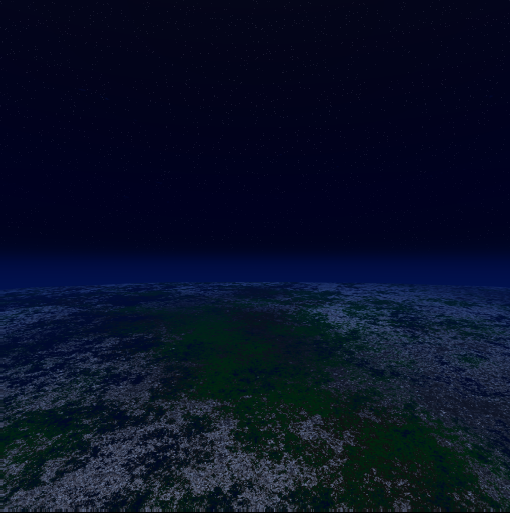
**Result – 구현 내용에 대한 항목별 결과 화면**

>> Parser 구현, 캐릭터 움직임, Sound System, 오브젝트 충돌 체크 등의 결과 이미지로 나타내기 힘든 것들은 제외함.

* Multi scene control (Title, Controls, Main1, Main2, Ending)

**** 

* Cube map 을 통한 Skybox 구현. (Background image)

* Multi lighting 구현.

>> 모닥불 주위로 동심원을 그리면서 밝아진 부분이 Spot Light, 나머지 부분들은 Directional Light



* Visual Effects 구현.

>> 왼쪽부터 핑크 포탈, 모닥불, 블루 포탈



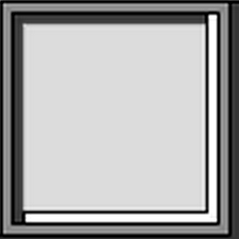
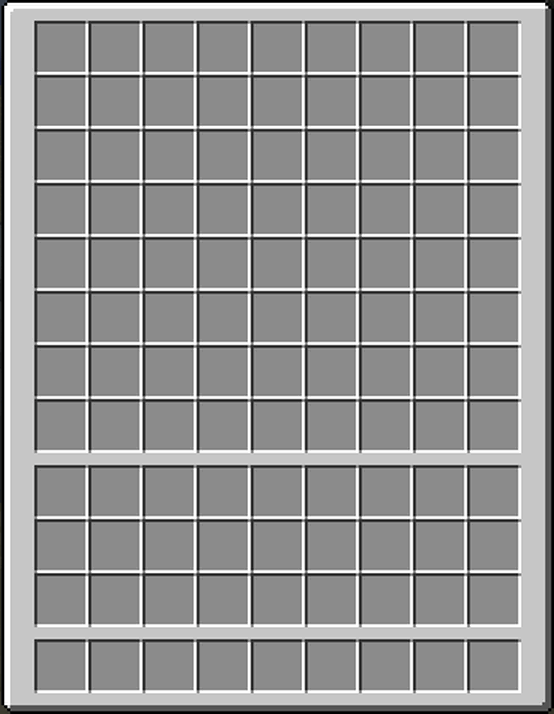
* 런타임 오브젝트 생성 / 제거 구현.



* 캐릭터 애니메이션 구현.

* UI System 구현 (Inventory, Interactive UI)



* 3인칭 카메라 워크 구현.

