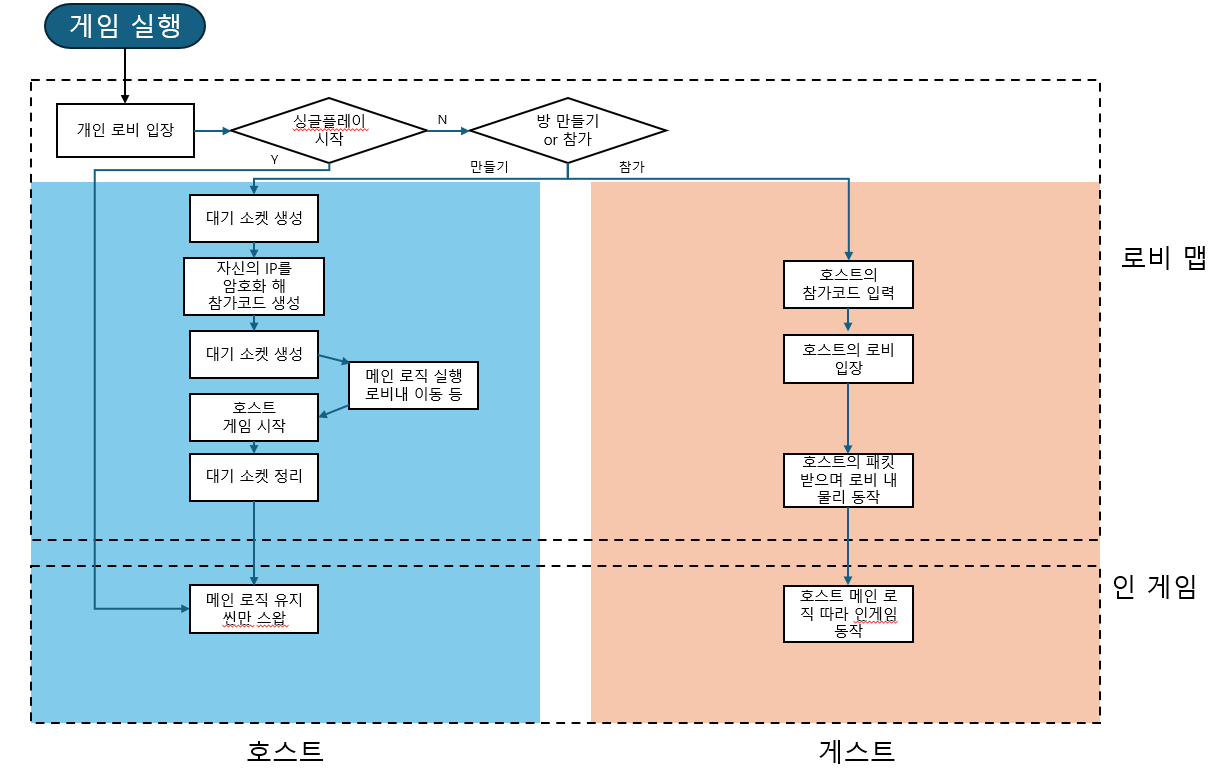
| **비단길(Silk Road)** | |
| --- | --- |
| **3주차** | **2024.01.06 (일) ~ 2023.01.13 (토)** |

**■ 주간 회의록**

**▶ 네트워크 모델 설계**

* 게임 실행
* [호스트]
* 로비에서 방을 만드는가 -> 대기소켓 생성
* 자신의 ip를 암호화해서 참가코드로 출력해주기
* 메인 로직 실행(대기실에서의 물리적 충돌, 이동, 시작 등등)
* 호스트 포함 최대 3명까지 받되 호스트가 시작시 대기소켓 정리
* 스테이지만 변경 메인로직 자체는 동일(대기실과 스테이지의 차이정도만)
* [게스트]
* 로비에서 문자열을 입력해 참가요청
* 문제가 없다면 호스트 로비에 입장
* 이후 호스트에게 패킷을 전송받아 게임 진행



e.g. 플레이어의 공격이 적에게 맞았을 경우 피격 모션이나 물리처리를 클라이언트에서 한 다음 실제 체력(수치)는 서버에서 계산을 한 다음에 반영

**▶ 이슈트래커 및 작업 현황 공유**

* Trello를 이용하여 작업 현황을 관리
* Issue, To do, Doing, Done, Release, Idea
* Issue: 해결하지 못하고 있는 문제나, 다른 사람이 구현한 기능에서 버그 발생 시 공유
* To do, Doing: 브랜치를 생성할/생성하여 작업할/작업중인 기능들
* Done: 기능 구현이 완료
* Release: 메인 브랜치에 머지까지 완료한 기능/작업들

**■ 다음 주 세부 계획**

**▶ [권순원]**

* 주간회의에서 서버 관련된 기반을 담당하는 것으로 결정되었으므로 초기 계획상 뒤쪽에 하기로 되어있던 파티클 시스템 관련된 부분을 잠시 멈추고 네트워크 기반 작업

**▶ [박준영]**

* 현재 하드코딩으로 구현된 텍스트 출력 기능을 최선을 다해 구조화 하여 푸시 및 팀원들이 편하게 호출하여 사용할 수 있도록 해보기
* 이벤트 시스템 구현으로 마우스 클릭 좌표를 받아서 캔버스 내의 오브젝트들이 클릭 되었는지 검사 기능 구현

**▶ [엄장헌]**

* 저번주에 애니메이션을 해보느라 못했던 collider와 옥트리를 다시 할 예정
* 3박4일 여행으로 위의 내용을 완성하는 것이 최선일 것 같다.

**■ 작업 일지**

**▶ [권순원]**

계획했던 새로 정한 코드스타일과 맞지 않는 부분을 전부 수정

추가적으로 gitignore 설정하면서 git에 업로드될 필요 없는 파일과 필요한데 업로드 되지 않은 파일을 정리

* FBX 관련 lib 파일들은 빌드에는 필요하지만 용량이 커서 git에 올라가지 않았음
* 모델 관련된 작업을 하면서 FBX lib 없이 자체적으로 모델 로드하는 기능을 구현해 관련 파일이 필요없도록 만들어야 할듯

파티클 시스템의 대략적인 흐름

1. 유니티 에셋 스토어에서 파티클 효과를 prefab의 형태로 다운로드
2. prefab에서 내가 정의한 형태의 파티클 파일로 변환해 저장
3. 게임이 시작하면 변환해 저장된 파일을 불러와 사용

파티클에 필요한 데이터를 정리해보면

* 지속 시간(주기)
* 재생 속도, 간격
* mesh, material
* 파티클의 움직임을 나타내는 쉐이더 코드 또는 그것을 정의할 수 있는 값

2번 변환의 경우는 본래는 prefab을 바로 불러와 게임에서 사용하는 것을 계획으로 했었는데, prefab 파일을 텍스트 에디터로 열어서 확인해보니 대략 3000줄의 문자열로 이루어진 파일이었기 때문에 변환기를 만드는 것으로 결정

단순히 바이너리 형태로만 바꾸는 것보다 이왕 변경할거면 게임에서 사용하지 않는 정보는 최대한 제거하고 필요한 부분만 저장하는 것이 좋을 것 같다고 생각함

2번 변환하는 과정에서 쉐이더 코드(또는 움직임을 나타낼 값)를 만들어 내는 과정을 얼마나 잘 만드냐가 파티클 시스템 활용의 중요한 점이 될 것 같음

이번 주 주간 회의를 하면서 네트워크 기반을 내가 만들기 시작하는 것으로 결정함에 따라 네트워크 관련 작업도 시작

* 회의 후 이틀 간은 이전 네트워크 작업물(넷겜프 텀프)을 보면서 어떤 식으로 구조를 만들 것인가를 구상하고 다음 주에 본격적인 코딩을 할 예정

네트워크 작업을 하면서 신경써야 할 부분

* 회의에서 결정한 것처럼 서버는 게임의 모든 수치(체력, 난이도 미터기 등)와 플레이어 및 오브젝트 움직임을 보정을 담당한다.
* 서버는 독립적으로 존재하는 것이 아니라 호스트 플레이어 코드 내에서 동작함으로 쓰레드를 이용해야 하고, 호스트 클라이언트와 데이터를 주고 받을 수 있도록 만들어야 한다.
* 싱글플레이 기준으로도 수치 계산을 위한 서버가 동작함으로 게스트로 게임에 참여하게 되면 해당 클라에서 사용하던 서버 스레드와 데이터를 주고 받는 것이 아닌 호스트의 서버와 데이터를 송수신할 수 있도록 변경해야 한다.

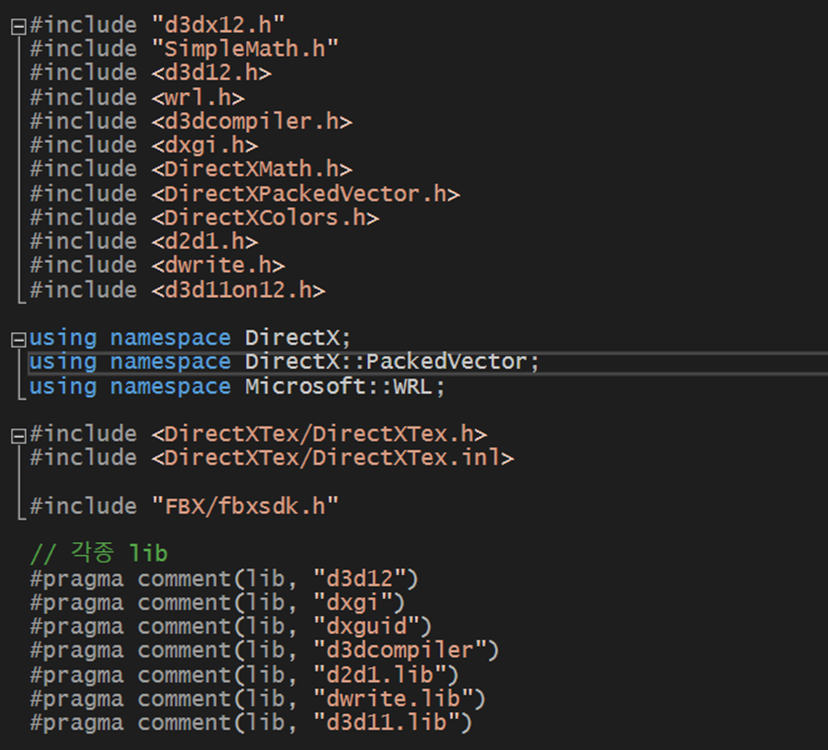
**▶ [박준영]**

본격적으로 ui 브랜치 생성 후 구현 시작

원래 계획대로 Direct2D, DirectWrite를 이용한 개발 착수

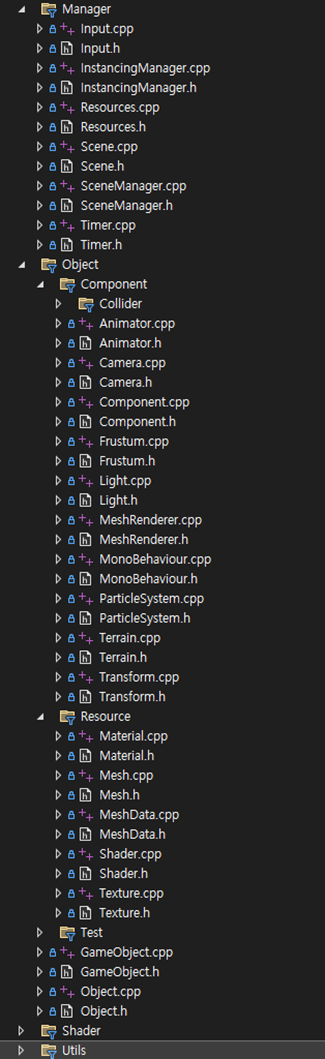
Direct2D등의 기능은 Directx11에 포함된 기능이기에 Directx12로 제작중인 우리 게임에 적용하기 위해선 D3D11On12를 이용해야함

D3D11On12 디바이스를 생성하며 Direct12디바이스와 같은 명령큐를 사용함. Engine의 init부분부터 접근



EnginePch.h에 위와 같이 d2d, dwrite, d3d11on12 라이브러리 및 헤더 추가

(d2d1과 dwrite의 경우 d2d1\_1, d2d1\_2 등 다양한 바리에이션 존재하는데 기능에 따라 구분한것으로 보임 추가 확인 필요)

현재 단순 출력은 가능 하나 거의 하드코딩 느낌

현재의 잘 구조화된 형태를 유지하며

자연스럽게 모듈로써 UI기능을 넣는데 집중 중

컴포넌트 패턴을 기반으로 하는 프로젝트에 맞춰

유사한 구조인 unity에서의 UI 구현 방법을

참고하는 중

유니티의 경우 canvas라는 개념이 있다

플레이어가 배치하는 모든 종류의 UI는 canvas의

자식으로써 배치되며 표시된다.

플레이어가 화면의 특정 영역을 클릭해서

어떠한 버튼이 눌렸는지 여부등을 검사하는

이벤트 시스템 또한 존재한다.

우리 게임에서도 canvas와 유사한 역할을 하는

싱글톤 객체를 생성하는 것을 우선으로 해야겠다.

Canvas는 Scene별로 포함되게 하면 될 것 같다

게임 엔진에서 Render를 호출 시  
 엔진 Render -> Scene Render -> Canvas Render -> 각 UI 오브젝트 Render 식으로 동작

화면 출력을 위의 기능을 기반으로 체계화 후 EventSystem을 구현하여 Canvas 내의 클릭 가능 오브젝트에게 마우스 클릭 정보를 전달해주면 될 것 같다.

고려사항

- 마우스 이동의 기본적인 기능은 시점 조작임. 그런데 업그레이드 창 등의 특정 팝업창에선 1인칭 카메라 시점조작을 멈추고 마우스 커서를 표시하여 2D UI를 사용할 수 있게 해야함. 팝업 생성 기능 구현 시 팝업을 생성할 때 위 기능을 켤 수 있는 옵션 필요

- 유니티에선 DOTween 과 같은 에셋을 통해 아래와 같이 UI 오브젝트에 동적인 애니메이션을 넣을 수 있음. 

잘 사용시 동일한 리소스를 사용한 UI 임에도 UX적 측면이나 미적인 측면에서 상당한 이점을 얻을 수 있기에 시간이 허락한다면 몇가지 대표 기능들을 만들어 보고 싶다.

* 현재 개발중인 프로젝트 디버그 시 일부 환경에서 해당 모니터의 최대 주사율로 프레임이 고정되는 현상 확인 (당장은 중요도 낮음)

**▶ [엄장헌]**

* 물리엔진을 위해 컴포넌트로 rigidbody 제작
* rigidbody에는 질량, 속도, 가속도등 물리량을 가진 변수와 중력을 사용할 지 등의 플래그들이 들어가며, update문을 통해 물리량 계산
* colider와 옥트리가 이번주 목표였는데 갑자기 애니메이션에 꽂혀서 시간을 너무 써버렸다..
* 애니메이션의 경우 mixamo 사이트를 통해서 우리의 모델에 애니메이션 코드를 입히고 fbx 파일로 만들어주는데, 이 경우 유니티에서는 잘 적용이 된다. 다만 window 3d뷰어와 우리의 코드에서 root motion이 적용이 안되는 듯하다. -> 골반이 거의 움직임이 없는 상태로 부자연스러움
* 유니티에 fbx파일에서 애니메이션 코드만 추출할 수 있는 기능이 있어서 이를 이용해 다른 모델에도 적용가능하게 애니메이션을 컴포넌트로 만들어보려했지만 실패했다. 위의 문제와 함께 조금은 나중에 다시 해봐야할듯
* 이번주는 조금 계획적으로하지 못 해서 일단 애니메이션은 잠깐 접어두고 다시 물리엔진으로 넘어가야겠다.