(1) High-Level 디자인

1. 프로토콜의 종류

- class CSceneMgr
- class CObject
- struct ObjInfo

(2) Low-Level 디자인

1. 프로토콜

```
enum eObjectType
        = { E_PLAYER, E_BULLET, E_BLOCK, E_ITEM, E_OBJ_END };
: 오브젝트의 종류
enum ePlayerType = { E_PLAYER_1, E_PLAYER_2, E_PLAYER3 };
: 플레이어의 번호
struct InitProtocol                         // accept()시 서버->클라
        ePlayerType eType;                 // 각 플레이어에게 할당될 번호
        vec2 vInitPos;                      // 시작 위치
struct ObjInfo
        vec2 vPos;                          // 위치
        int iValue                          // 각 오브젝트가 가지는 특성
        eObjectType eObjType                // 오브젝트 종류
struct ObjInfo_from_Client                          // 클라 -> 서버
        ObjInfo tInfo                       // 오브젝트 정보
        ePlayerType ePlayerType;            // 할당된 번호
***수정***struct ObjInfo_from_Sever                 // 서버 -> 클라
        ObjInfo tInfo                       // 오브젝트
struct vec2
        float fX;
        float fY;
struct vec3
        float fX;
        float fY;
        float fZ;
struct vec4
        float fX;
        float fY;
        float fZ;
        float fW;
```

2. 클라이언트

class CSceneMgr : 스테이트(state)를 관리하는 싱글톤 클래스. CScene 변수를 통해 현재 실행되는 스테이트의 갱신하고 그린다.

enum eSceneType
= {E_TITLESCENE, E_READYSCENE, E_GAMESCENE}

CSceneMgr* GetInstance();
처음 호출 시 멤버변수 m_pInstance == NULL 이라면 인스턴트 객체를 생성하는 함수.

DestroyInstance();
게임 종료시 객체를 삭제해 주는 함수.

Update();
게임 내의 모든 오브젝트들을 갱신 시키는 함수.

Draw();
게임 내의 모든 오브젝트들을 그리는 함수.

Change_Scene(eSceneType eType)
각 씬에서 Change_Scene()이 호출되면 들어온 매개 변수에 의해 씬이 변환된다.

CSceneMgr* m_pInstance = NULL;
싱글톤으로 사용하기위해 자기 자신을 포인터로 가지고 있는 변수.

eSceneType m_eCurSceneType;
현재 실행되고 있는 씬의 타입을 저장하는 변수.

CScene* m_pScene;
현재 실행되고 있는 씬을 저장하는 포인터 변수.

list<CGameObject*> m_pObjList[E_OBJ_END];
씬에 존재하는 객체들의 리스트

class CScene : 모든 씬의 부모가 되는 클래스

virtual void Initialize() = 0;

virtual void Update() = 0;

virtual void Draw() = 0;

class CWaitScene : public CScene

다른 플레이어를 기다리는 씬의 정보를 가지고 있는 클래스

virtual void Initialize();
connect() 호출하는 함수

virtual void Update();
다른 플레이어가 접속을 할 때까지 대기하고 있는다. 모두 접속했으면 키 입력을 통해 CSceneMgr::Change_Scene()을 호출한다.

class CGameScene : public CScene

실제 게임이 진행되는 씬의 정보를 가지고 있는 클래스

virtual void Initialize();

맵의 블록과 다른 플레이어의 초기 정보를 초기화하는 함수.

virtual void Update();

실제 게임이 진행되는 함수. 키 입력을 통해 움직임,

class CObjMgr : 게임 내의 오브젝트들을 관리하는 클래스. 싱글톤으로 만들어 사용한다.

Initialize()

list를 초기화 해주는 함수.

Update()

list안의 객체들의 Update()를 호출한다

Draw()

list안의 객체들의 Draw()를 호출하여 그려주는 함수.

Release()

객체가 삭제될 때 list안의 객체를 모두 삭제시킨다.

list<CGameObject*> m_pObjList[E_OBJ_END];

오브젝트의 종류에 따라 나누어 관리하는 리스트.

class CGameObject : 모든 오브젝트의 부모가 되는 클래스

virtual void Initialize() = 0;

virtual void Update() = 0;

virtual void Draw() = 0;

ObjInfo tInfo; // 객체의 정보를 담고있는 변수.

class CPlayer : public CGameObject

플레이어의 정보를 가지는 클래스.

virtual Initialize();

서버에서 받아온 정보(ePlayerType, e)

virtual Update();

키 입력을 통해 플레이어를 움직이는 함수.

키 입력이 들어왔을 때 서버에 플레이어 정보를 send()

virtual Draw();

vec4 Get_BoundingBox();

플레이어의 위치와 사이즈를 이용해

vec4(minX, minY, maxX, maxY)로 반환해준다.

vec2 m_vSize // 플레이어의 사이즈

2. Server