

포트폴리오

클라이언트 프로그래머 지원 최예람

CONTENTS

팀 프로젝트 참여

게임 설명

맡은 업무

참여도

Render Sector

Sector 구조

Sector Manager

Quad Tree

LRU Queue 구현

Map (Red Black Tree)

MemoryPool

IOCP

전체 구조

비동기화

State 패턴

Dump

A*

Excel Reader

Mouse Manager

팀 프로젝트 참여

게임 설명

플레이어마다 각 다른 직업을 선택.
플레이어의 직업에 배정된
정령을 이용 전투를 통한
보스전 게임

(3인 멀티 플레이)

PATH TO YIGGDASIL

PROJECT PYG



플랫폼	PC
장르	액션
카메라	<u>버드뷰</u>
엔진	UNITY

기본 컨셉

보스
섬멸

일반구간 없음
보스전만 진행

역할
수행

탱커, 딜러, 힐러 등
각자의 역할 수행

간접
전투

직접공격 불가
제약이 걸린 간접공격

팀 프로젝트 참여

맡은 업무

C++와 Unity의 서버 구현.

로그인/회원가입 DB 처리.

로그인 – 게임 입장까지의
UI 클라이언트 기능 구현.
서버 기능 구현.

SectorManager 구현.

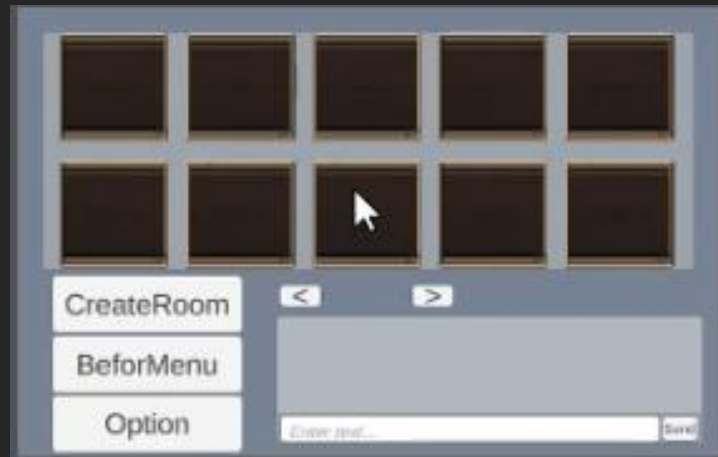
Excel Reader

A* 구현

서버+클라
로그인 – 게임 접속까지 처리
시연 영상 링크 넣기

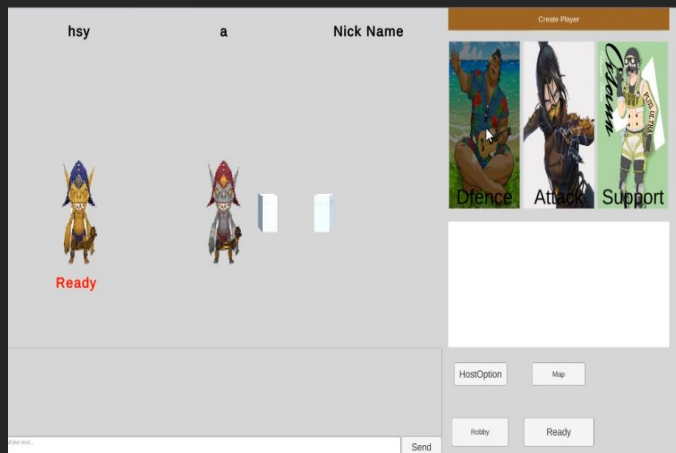


로그인/회원가입



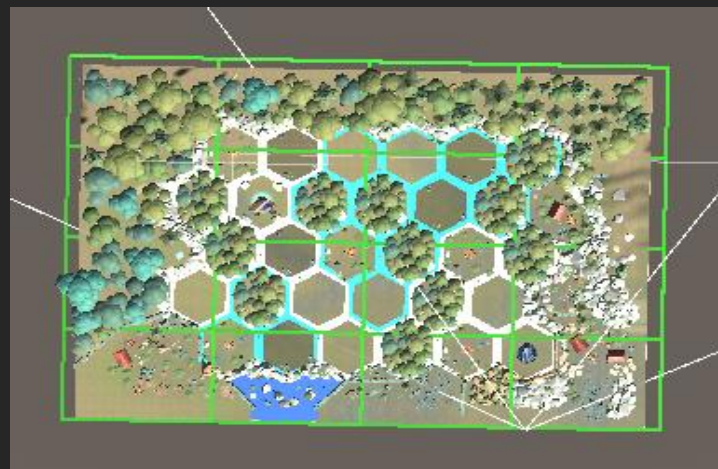
로비 기능

- 방 만들기
- 방입장
- 채팅
- 옵션 설정



방 기능

- 호스트/멤버 관리
- 레디
- 캐릭터 선택 (중복 선택 허용 X)
- 맵 선택



게임 기능

- 로딩에 필요한 정보
서버-클라 간의 교환 및
데이터 설정.

팀 프로젝트 참여

참여도

Roka0105

분배된 역할에 맞게 성실히
임하였으며 팀원들과의 적극적인
소통과 협업을 하였습니다.

필요한 사항, 궁금한 사항이
생길 시 소통을 통해 해결.

client update	게임 상태로 넘어가는 틀 완성 게임으로 썬 이동 해야 하는데 멀티C	23 8 2022 23:10	roka0105 <cocapepsi123@daum.net>
server update	뒷처리 추가 및 버그 수정	23 8 2022 22:47	roka0105 <cocapepsi123@daum.net>
server bug	수정 만약 다음 호스트가 없다면 nullptr 해놓기	23 8 2022 22:32	roka0105 <cocapepsi123@daum.net>
server update	버그 수정 호스트 나갔을때 다음 유저에게 호스트 정보 넘기기.	23 8 2022 22:30	roka0105 <cocapepsi123@daum.net>
client update	닉네임 버그 수정	23 8 2022 22:26	roka0105 <cocapepsi123@daum.net>
client update	닉네임 터지는 버그 수정	23 8 2022 22:17	roka0105 <cocapepsi123@daum.net>
server update	방 나가기 처리 -> 로비로 나가기 -> 강종 처리	23 8 2022 21:47	roka0105 <cocapepsi123@daum.net>
client update	로비로 나가기 처리 완료 (강종처리 아직 안함)	23 8 2022 21:38	roka0105 <cocapepsi123@daum.net>
client update	호스트가 게임 시작 버튼 눌렀을때 처리 구현중.. 일부 업로드	23 8 2022 19:50	roka0105 <cocapepsi123@daum.net>
Merge branch 'main' of https://github.com/hamsoyeon/Yggdrasil		23 8 2022 19:34	roka0105 <cocapepsi123@daum.net>
맵 안바뀌는 버그 수정		23 8 2022 19:34	roka0105 <cocapepsi123@daum.net>
UIScene Update		23 8 2022 19:25	JangHan Kim <kimjh741963@hanmail.net>
UIScene room->플레이어 닉네임 출력(서버쪽확인을 못함)		23 8 2022 15:05	MikangMark <kimjh741963@hanmail.net>
플레이어 추적경로타일 색 지우기		23 8 2022 13:52	rilphy <wuman130@gmail.com>
예람 할일 업데이트 2		22 8 2022 23:06	roka0105 <cocapepsi123@daum.net>
예람 할일 업데이트		22 8 2022 23:05	roka0105 <cocapepsi123@daum.net>
Merge branch 'main' of https://github.com/hamsoyeon/Yggdrasil		22 8 2022 22:59	roka0105 <cocapepsi123@daum.net>
server update	호스트 정보 string -> int 로 타입 바꿈	22 8 2022 22:58	roka0105 <cocapepsi123@daum.net>
client update	맵 선택 맵 값은 임시로 줌	22 8 2022 22:57	roka0105 <cocapepsi123@daum.net>
보스 스테미나 UI추가		22 8 2022 20:55	JangHan Kim <kimjh741963@hanmail.net>
client 버그 수정	다른 사람 채팅 출력 안되는 버그 수정 완료	22 8 2022 20:48	roka0105 <cocapepsi123@daum.net>
UIScene 텍스트수정		22 8 2022 20:05	JangHan Kim <kimjh741963@hanmail.net>
client 텍스트 프리뷰 추가		22 8 2022 19:47	roka0105 <cocapepsi123@daum.net>
Merge branch 'main' of https://github.com/hamsoyeon/Yggdrasil		22 8 2022 19:29	Jiny <jiny9789@gmail.com>
보스무브 수정(자잘한 버그발생)		22 8 2022 19:29	Jiny <jiny9789@gmail.com>
server update	로비 -> 룸 으로 넘어갈때 상태 안바뀌는 버그 수정. 및 방에서의 X	22 8 2022 19:28	roka0105 <cocapepsi123@daum.net>
client update	방에서의 채팅 기능.	22 8 2022 19:26	roka0105 <cocapepsi123@daum.net>
server update	이미 레디 상태일때 다른 캐릭터 선택해도 렌더링 안바뀌게 버그 P	22 8 2022 18:00	roka0105 <cocapepsi123@daum.net>
client update		22 8 2022 17:57	roka0105 <cocapepsi123@daum.net>
client update	레디했을때 UI적으로 보여주는거랑 누가 캐릭터 선택 후 레디하면 .	22 8 2022 17:21	roka0105 <cocapepsi123@daum.net>
server update	레디 부분 내용 수정	22 8 2022 15:37	roka0105 <cocapepsi123@daum.net>
[UIScene]맵선택 (임시)		21 8 2022 23:37	JangHan Kim <kimjh741963@hanmail.net>
client update	일반 레디 명령 처리 렌더링 부분은 장한이 코드랑 합칠 예정	21 8 2022 17:26	roka0105 <cocapepsi123@daum.net>
client update	요청 받은 부분 수정 완료	21 8 2022 14:56	roka0105 <cocapepsi123@daum.net>
소연 클라 업데이트		21 8 2022 12:51	hamsoyeon <chair3203@naver.com>
[UIScene] 레디부분 수정 타 플레이어 캐릭선택시 그캐릭 선택 비활성화		21 8 2022 5:21	JangHan Kim <kimjh741963@hanmail.net>
예람 할일 update		21 8 2022 2:37	roka0105 <cocapepsi123@daum.net>
Merge branch 'main' of https://github.com/hamsoyeon/Yggdrasil		21 8 2022 2:19	roka0105 <cocapepsi123@daum.net>
client update	기존코드는 먼저 들어간 유저가 캐릭터 선택 한 뒤에 다른 플레이아	21 8 2022 2:19	roka0105 <cocapepsi123@daum.net>
매인씬 카메라 시네머신 작업중		21 8 2022 0:48	JangHan Kim <kimjh741963@hanmail.net>
client update	test object prefab 추가 플레이어 직업 선택시 어떤 모델 렌더링할지	21 8 2022 0:25	roka0105 <cocapepsi123@daum.net>
server update	플레이어 입장에 관한 정보 모두에게 뿌리는거 추가 플레이어가 캐	21 8 2022 0:24	roka0105 <cocapepsi123@daum.net>

Render Sector

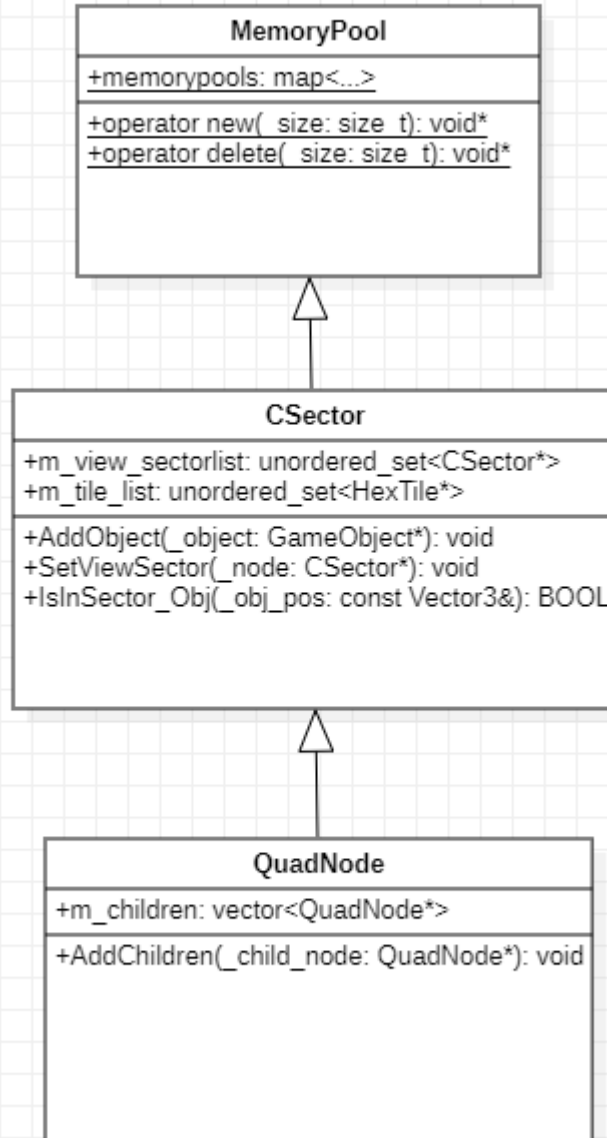
Sector 구조

QuadNode 는
Children 정보를 관리

Sector 는
현재 Sector의 시야 내에
들어오는 타일들의 정보와
Sector들의 정보를
관리 합니다.

+ 오브젝트들의 정보는
Sector를 통하여
해당하는 타일에
등록됩니다.

확장성을 위해 추후에
Csector와 QuadNode의 상속구조를
바꿀 예정입니다.



```
#include "MemoryPool_2.h"
class GameObject;
class HexTile;
class CSector : public MemoryPool_2
{
protected:
    CSector();
    CSector(Vector3 _sender_pos, Vector3 _distance);

    virtual ~CSector();
public:
    void AddObject(GameObject* _object); // 오브젝트 등록
    const Vector3 GetDistance();
    void SetDistance(Vector3 _pos);
    const Vector3 GetSender();
    const Vector3 GetStartPos();
    BOOL IsInSector_Direction(const Vector3 _obj_pos, E_NodeType _type);
    BOOL IsInSector(const Vector3 _obj_pos); // 오브젝트가 노드안에 있는지 체크
    BOOL IsInSector_Obj(const Vector3& _obj_pos);
    void SetViewSector(CSector* _node);
    unordered_set<CSector*> GetViewSector();
    unordered_set<HexTile*> GetTileList();
private:
    unordered_set<CSector*> m_view_sectorlist;
    unordered_set<HexTile*> m_tile_list;
    Vector3 m_start_pos;
    Vector3 m_sender_pos; // 노드의 중심 위치
    Vector3 m_distance; // 말변/2
};

class QuadNode : public CSector
{
public:
    static int GetCreateCount();
public:
    QuadNode();
    QuadNode(Vector3 _sender_pos, Vector3 _distance);
    virtual ~QuadNode() final;
    void AddChildren(QuadNode* _child_node); // 자식노드 등록
    QuadNode* GetChildNode(int index); // 자식노드 가져오기

    void SetID();
    int GetID();
    QuadNode* GetParent(); // 부모노드 가져오기

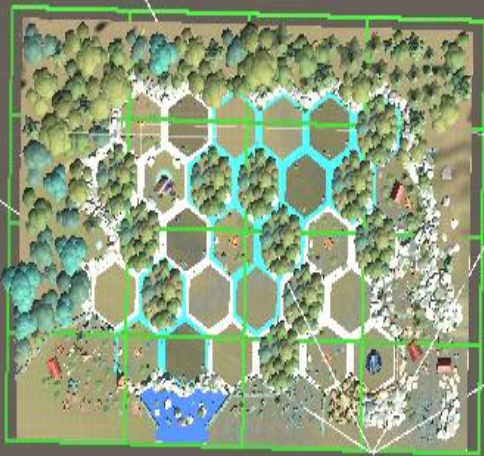
    int Child_Size();
private:
    static int create_count;
    int m_id;
    vector<QuadNode*> m_children; // 자식노드
    QuadNode* m_parent; // 부모노드
    BOOL m_is_culled; // 컬링 여부
    const float m_limit_depth = 4; // 트리의 최대 깊이
    E_NodeType m_type;
};
```


Render Sector

SectorManager

Sector Manager 에서
Game에 돌입했을 때
해당 게임의 맵에 대한
Sector와 그 자식들의 값을
설정합니다.

서버에서 충돌 처리를 하게 될
경우를 생각해 Quad Tree로
구현 하였습니다.



초록선=sector / 현재 depth:2

CSectorMgr
+m_roots: map<...>
-CreateQuadTree(...): void
-CreateChildren(_root: QuadNode*, ...): void
-SerchObjectNode(_root: QuadNode*, ...): QuadNode**
+AddQuadTree(...): void
+SerchObjectNode(_gameinfo: t_GameInfo*, _pos: Vector3): QuadNode*
+SendViewSectorFunc(...): void
+SendViewTileFunc(...): void
+PlayerMoveProcess(...): void

```
#pragma once
#include "pch.h"
#include "CSector.h"
#include "CGameMgr.h"

class CSector;
class GameObject;
class CSession;
class CSectorMgr
{
public:
    static CSectorMgr* GetInst();
    static void Create();
    static void Destory();
private:
    static CSectorMgr* m_instance;
    CSectorMgr();
    ~CSectorMgr();

    void CreateQuadTree(t_GameInfo* _gameinfo, t_MapInfo* _mapinfo);
    void CreateChildren(QuadNode* _parent, Vector3 _centerpos, Vector3 _distance, int _curdepth, t_GameInfo* _gameinfo);

    QuadNode** SerchNode(QuadNode* _parent, int _id, int _curdepth, t_MapInfo* _mapinfo);
    void SetViewNode(QuadNode* _parent, int _curdepth, t_MapInfo* _mapinfo, UINT _gameid);
    QuadNode** SerchNode(QuadNode* _parent, Vector3 _pos, int _curdepth, t_MapInfo* _mapinfo);
    QuadNode** SerchObjectNode(QuadNode* _parent, Vector3 _pos, int _curdepth, t_MapInfo* _mapinfo);

    void SendViewSectorFunc(CSession* _session, t_GameInfo* _gameinfo, Vector3 _objpos);
    void SendViewTileFunc(CSession* _session, t_GameInfo* _gameinfo, Vector3 _objpos);
public:
    void Init();
    void End();
    void AddQuadTree(t_GameInfo* _gameinfo, t_MapInfo* _mapinfo);
    void SendInit(UINT _gameid, CSession* _session, t_MapInfo* _mapinfo);
    QuadNode* SerchObjectNode(t_GameInfo* _gameinfo, Vector3 _pos);
    //test
    QuadNode* return_root(UINT gameid)
    {
        return m_roots[gameid];
    }
    /* ... */

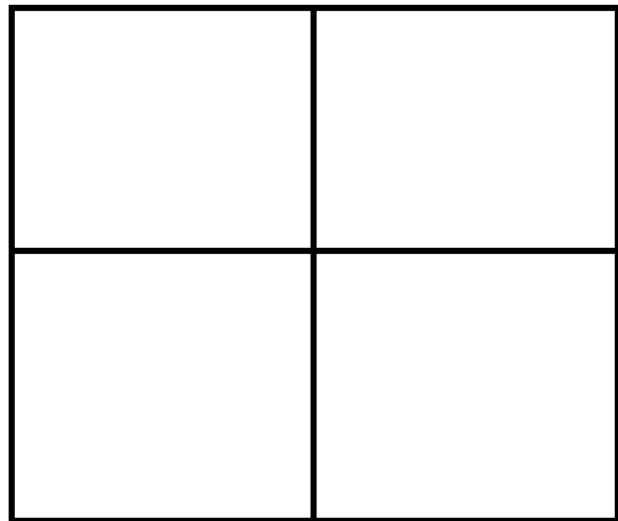
    void PlayerMoveProcess(CSession* _session, t_GameInfo* _gameinfo);
public:
    void Packing(unsigned long _protocol, Vector3 _startpos, Vector3 _endpos, float _h_distance, float _v_distance, int _sectorcount, CSession* _session);
    void Packing(unsigned long _protocol, list<Vector3>& _starts, Vector3& _distances, CSession* _session);
    void UnPacking(byte* _recvbuf, Vector3& _pos);
private:
    Clock* m_lock;
    RBT<UINT, QuadNode*> m_roots; // key = gameinfo id, value = rootnode
};
```

사용 언어 : C++

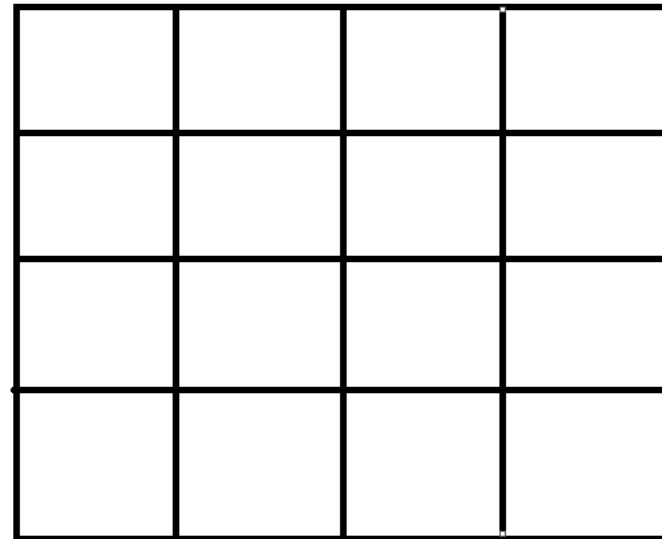
Render Sector

Quad Tree

Depth값을 주어 해당 값
수정 시 트리가 4의 depth승
만큼 구역이 쪼개집니다.

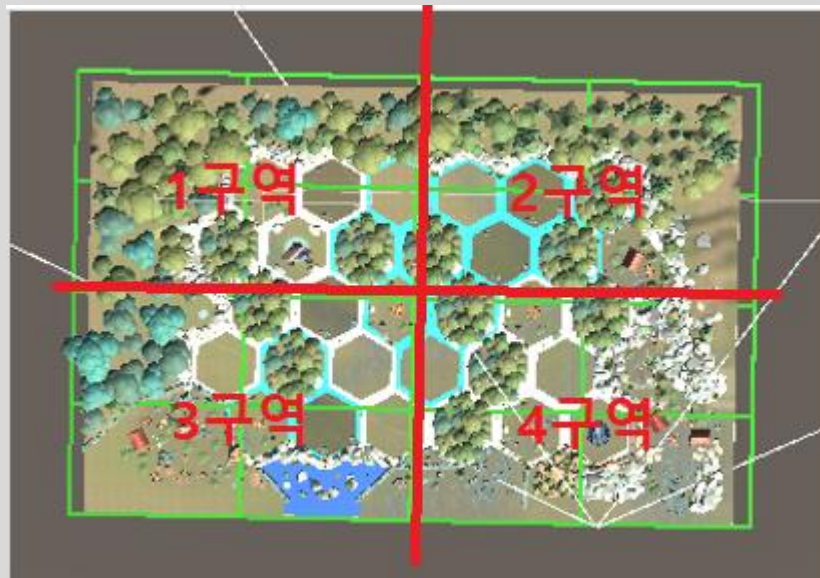


Depth = 1



Depth = 2

유효한 데이터는 제일 깊이 있는
노드들 이며, 상위 노드들은 구
역의 의미만을 갖습니다.

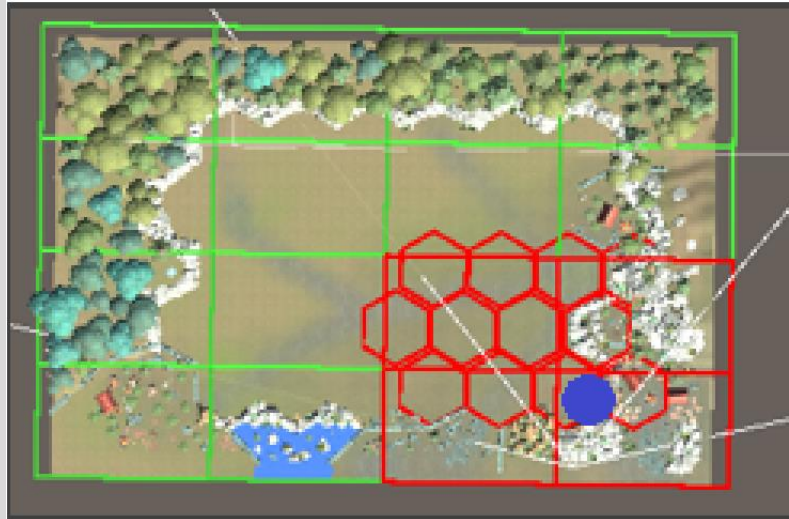


Render Sector

Quad Tree

Player의 위치를 받아와
해당 위치에 속하는
Sector를 탐색.

해당 Sector와 Tile 정보를
잘 읽어서 클라에 전송했는지
확인을 위해 해당 범위를
빨강게 변하도록 처리했습니다.



Player 위치 = 파란 원

```
QuadNode* CSectorMgr::SerchObjectNode(t_GameInfo* _gameinfo, Vector3 _pos)
{
    QuadNode* node = reinterpret_cast<QuadNode*>(*SerchObjectNode(m_roots[_gameinfo->m_id], _pos, 0, _gameinfo->m_mapinfo));
    return node;
}
#pragma endregion
```

```
QuadNode** CSectorMgr::SerchObjectNode(QuadNode* _parent, Vector3 _pos, int _curdepth, t_MapInfo* _mapinfo)
{
    if (_mapinfo->m_depth == _curdepth)
    {
        if (_mapinfo->m_start_position.x <= _pos.x && _mapinfo->m_end_position.x >= _pos.x
            && _mapinfo->m_start_position.z >= _pos.z && _mapinfo->m_end_position.z <= _pos.z)
        {
            //if (_parent->IsInSector_Direction(_pos, _type))
            //{
                return &_parent;
            //}
        }
        return nullptr;
    }
    QuadNode* child = nullptr;
    QuadNode** item = nullptr;
    int size = _parent->Child_Size();
    for (int i = 0; i < size; i++)
    {
        child = _parent->GetChildNode(i);
        if (child == nullptr)
        {
            return nullptr;
        }
        if (child->IsInSector_Obj(_pos))
        {
            item = SerchObjectNode(child, _pos, _curdepth + 1, _mapinfo);
            if (item != nullptr)
                return item;
        }
    }
    return nullptr;
}
```

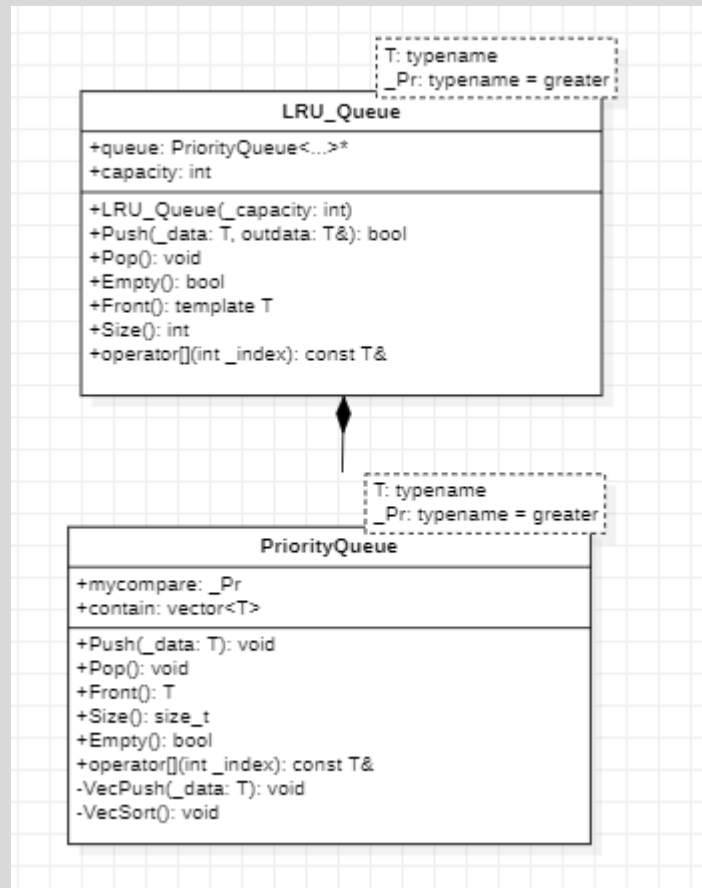
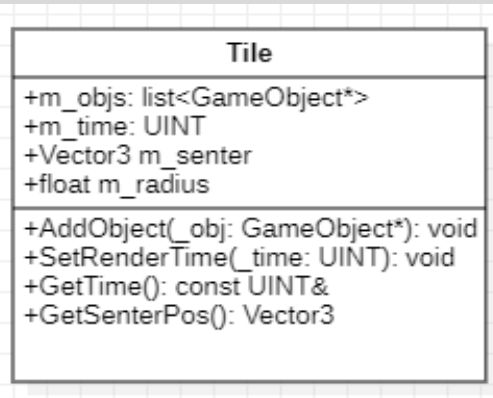
위치값 기반으로 Sector 검색

Render Sector

LRU Queue

우선순위 큐를 구현하여
렌더링 범위에 들어올 때 마다
해당 Tile에 대한 time 변수를
그때의 시간으로 갱신해 줍니다.

불 필요하게 렌더링을 ON/OFF
비용을 줄이기 위해
Queue의 용량 만큼은
렌더링을 유지하되
새로운 오브젝트가
추가로 들어왔을 때는
가장 시간 갱신이 안된
오래된 오브젝트부터 렌더링에서
제외시킵니다.



```
void VecPush(T _data)
{
    contain.push_back(_data);

    for (int i = 1; i < contain.size(); i++)
    {
        int child = i;
        do
        {
            int root = (child - 1) / 2;
            if (mycompare(contain[root], contain[child]))
            {
                T temp = contain[root];
                contain[root] = contain[child];
                contain[child] = temp;
            }
            child = root;
        } while (child != 0);
    }
}

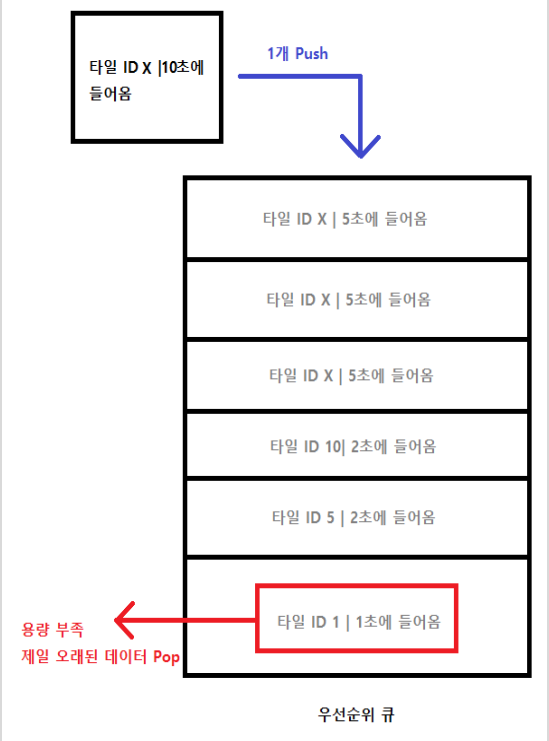
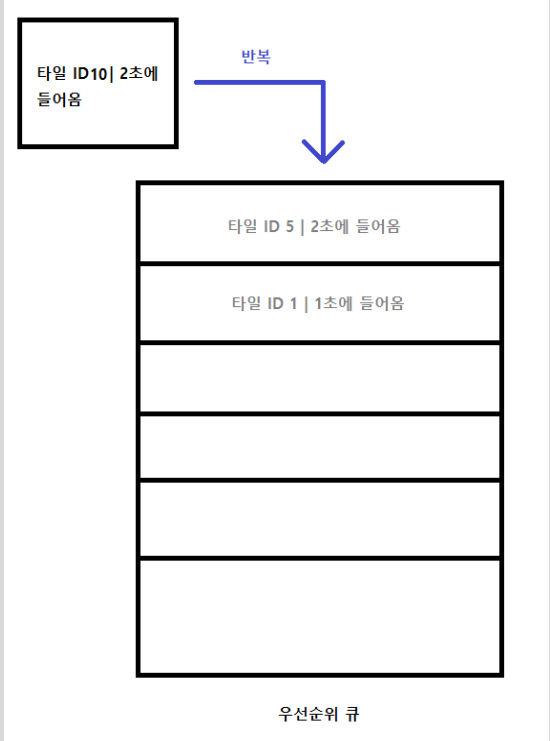
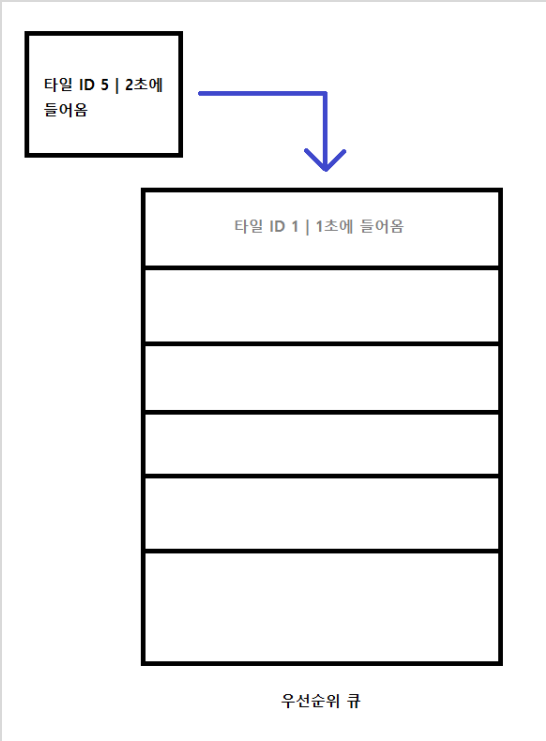
void VecSort()
{
    if (contain.size() == 1)
    {
        return;
    }
    for (int i = contain.size() - 1; i >= 0; i--)
    {
        T temp = contain[0];
        contain[0] = contain[i];
        contain[i] = temp;
        int root = 0;
        int child = 1;
        do
        {
            child = root * 2 + 1;
            if (this->mycompare(contain[child], contain[child + 1]) && child < i - 1)
            {
                child++;
            }
            if (this->mycompare(contain[root], contain[child]) && child < i)
            {
                temp = contain[root];
                contain[root] = contain[child];
                contain[child] = temp;
            }
            root = child;
        } while (child < i);
    }
}
```

Priority Queue.h

Render Sector

LRU Queue

갱신이 오랫동안 안되어 있는데
Queue가 꽉 찬 경우
갱신 시점이 오래된 Tile 부터
Queue 에서 제외 시켜 줍니다.



Render Sector

LRU Queue

각 Session 들이
렌더중인 Sector 정보가
다를 것 이기 때문에
Session 마다 LRU Queue를
소유하게 했습니다.

실행 화면(Server)은
제외된 타일의 ID와
해당 타일이 렌더링
시작된시점의 시간을 출력합니다.

실행 화면(Client)은
검정 원->빨간 원의 위치로
이동했을 때 시야 범위이며

검치는 부분의 오브젝트=시간갱신!
아닌 부분=제외될 수 있음!

```
CSession.h
LRU_Queue<HexTile*, mygreater>* m_real_queue;
LRU_Queue<HexTile*,mygreater>* m_temp_queue;

void SetSector(QuadNode* _nodesector)
{
    m_sector = _nodesector;
    double curtime = 0;
    time_t last_update_time = time(NULL);
    unordered_set<CSector*> viewlist = m_sector->GetViewSector();

    bool same = false;
    unordered_set<HexTile*> tilelist;
    list<HexTile*> suc_tile;
    list<HexTile*> new_tile;
    while (m_real_queue->Empty() == false)
    {
        same = false;
        HexTile* temp = m_real_queue->Front();
        m_real_queue->Pop();
        for (auto sector : viewlist) { ... }
        if (!same)
        {
            //리얼 큐에 있고 타일 리스트에 없는 경우=원래 시간으로 넣기.
            //다 들았는데 tilelist에 없었다. real queue에만 있었다.
            HexTile* data = nullptr;
            if (m_temp_queue->Push(temp, data))
            {
                cout << data->GetTime() << " 이거 지워짐\n";
                delete data;
            }
        }
    }

    //리얼 큐에 없고 타일 리스트에만 있는 경우
    for (auto sector : viewlist) { ... }

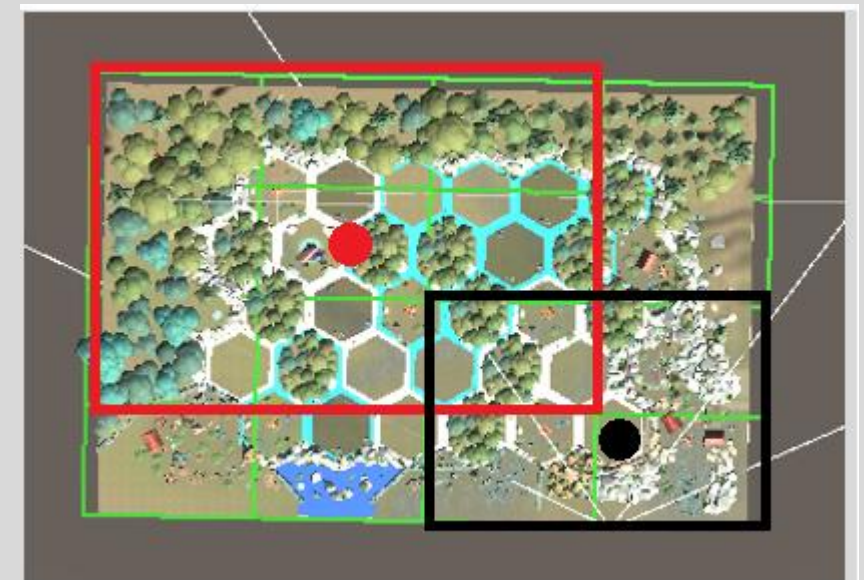
    LRU_Queue<HexTile*,mygreater>* tempqueue;
    tempqueue = m_real_queue;
    m_real_queue = m_temp_queue;
    m_temp_queue = tempqueue;
}
```

Queue의 렌더링 정보를 갱신하는 코드 입니다.

TileID | 갱신 시간

36	1663167027	이거	지워	짐
37	1663167027	이거	지워	짐
22	1663167027	이거	지워	짐
23	1663167027	이거	지워	짐
31	1663167027	이거	지워	짐
38	1663167027	이거	지워	짐
39	1663167027	이거	지워	짐
12	1663167062	이거	지워	짐
13	1663167062	이거	지워	짐
14	1663167062	이거	지워	짐
1	1663167062	이거	지워	짐
2	1663167062	이거	지워	짐
3	1663167062	이거	지워	짐

실행 화면 (Server)



실행 화면(Client)

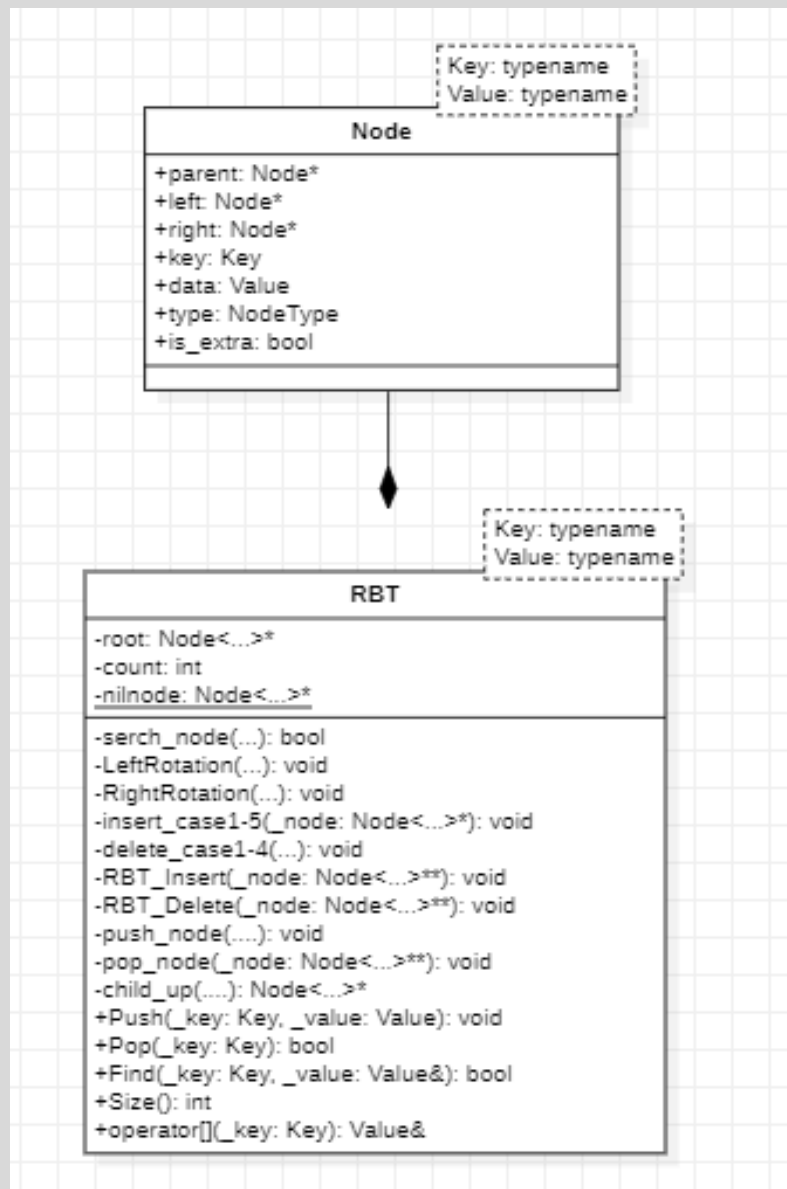
사용 언어 : C++

Map

Red-Black-Tree

Red-Black-Tree 알고리즘을
학습하여 완전 이진 트리를
항상 유지하도록 하여
검색 시간을 단축하도록 했습니다

하지만 템플릿 특수화는
적용하지 않았기 때문에
key값은 int 형만 가능합니다.



```
enum class NodeType
{
    None,
    Black,
    Red
};
```

노드 멤버 변수 Type Enum

```
template<typename Key, typename Value>
class Node
{
```

```
};
template <typename Key, typename Value>
class Node<Key, Value*>
```

노드에 대한 특수화
(포인터 데이터일 경우)

```
public:
    RBT() { ... }
    void Push(Key _key, Value _value)
    {
        Node<Key, Value*> node = new Node<Key, Value*>(_key, _value, nilnode, NodeType::Red);
        push_node(&root, node);
        RBT_Insert(&node);
        count++;
    }
    bool Pop(Key _key)
    {
        Node<Key, Value*>*& serchnode;
        if (serch_node(&root, _key, serchnode))
        {
            pop_node(serchnode);
            count--;
            return true;
        }
        return false;
    }
    bool Find(Key _key, Value& _value) { ... }
    int Size() { ... }
    Value& [ ... ]
};
```

RBT.h public 함수

사용 언어 : C++

Map

Red-Black-Tree

구현한 STL을 map으로 사용.

```
UB11C:  
static RBT<DWORD, t_ThreadInfo*> g_threadinfo;  
protected:  
Clocp.h* ↗ ✕
```

Clocp.h

= 스레드ID,스레드 정보

```
RBT<UINT, t_MapInfo*> m_maps;  
CMapMgr.h ↗ ✕
```

CMapMgr.h

= 맵ID,맵 정보

```
RBT<UINT, QuadNode*> m_roots; // key = gameinfo id, value = rootnode  
CSectorMgr.h ↗ ✕
```

CSectorMgr.h

= GameID,root Node주소

```
static RBT</*객체크기*/UINT,/*pool 정보*/t_pool_info*> memorypools;  
MemoryPool_2.h ↗ ✕
```

MemoryPool_2.h

= 메모리 크기,생성된 memorypool 정보

MemoryPool

동작 방식

메모리 조각화 없이 재사용하면서 사용하기 위해서 구현했습니다.

1.Operator new /delete
사용으로 메모리 풀을 구현.

2.사용 요청이 들어온 데이터를
일정 단위로 구분된 메모리풀 중
해당되는 단위의
메모리풀을 배정.

(생성이 안 되어있다면 최초
사용할 때 생성해서 map에 등록)
배정된 size = map key

3.그 메모리풀에서
메모리 블록을 할당 받는다.

```
static RBT</*객체크기*/UINT,/*pool 정보*/t_pool_info*> memorypools;
```

```
#define MAXMEMORY_BYTE 32768

#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>
#include <vector>
#include <map>

using namespace std;

struct t_pool_info
{
    t_pool_info(const int _capacity, const int _size) { ... }
    ~t_pool_info() { ... }
    char* current;
    //이미 생성된 블록이 꽉 차서 새로운 블록을 만들었을 때 그 블록의 시작 주소.
    vector<char*> startptr;
};

class MemoryPool_2
{
public:
    static void* operator new(size_t _size);
    static void* operator new[](size_t _size);

    static void operator delete(void* _object, size_t _size);

    static int AssignSize(size_t _size);
    static void End();

protected:
    static RBT</*객체크기*/UINT,/*pool 정보*/t_pool_info*> memorypools;
};
```

Memorypool_2.h

사용 언어 : C++

MemoryPool

AssignSize()

만약 1byte의 메모리 요청이 들어온다면 8byte memorypool에서 메모리 블록을 반환 합니다.

8byte부터

2의 제곱만큼 size를 할당하는 이유
64bit 기준 포인터 크기 = 8byte

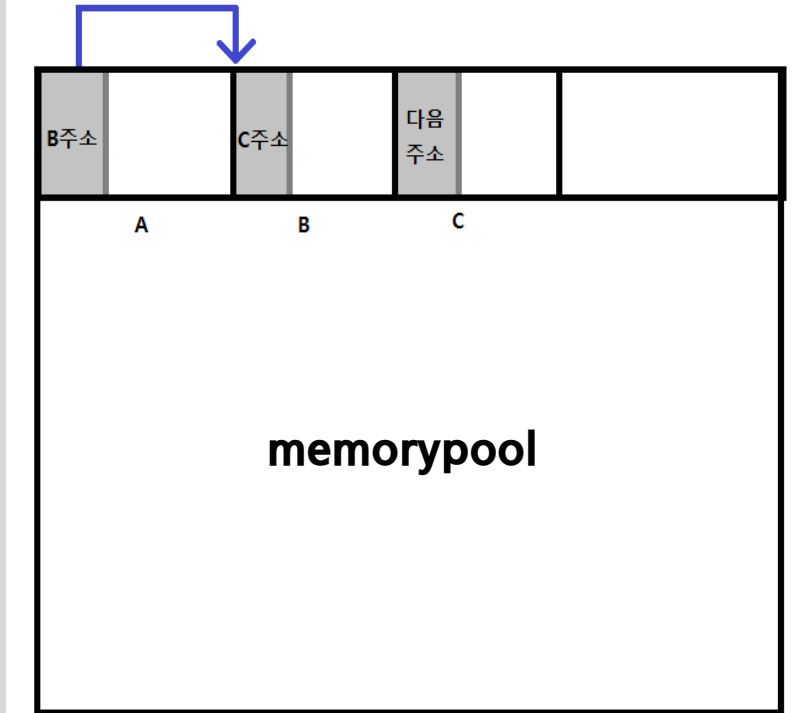
현재 메모리풀은 각 블록에서 다음 할당 시 호출할 메모리 주소가 쓰여져 있습니다.

따라서 메모리 블록의 최소 크기는 8byte 여야 했습니다.

또 1byte 단위로 다르다고 모두 Memorypool을 만들어주는것은 해당 단위의 객체가 많이 사용 되지 않을 경우 낭비가 되기 때문에 일정 단위로 pool을 만들어 사용하도록 하였습니다.

```
int MemoryPool_2::AssignSize(size_t _size)
{
    if (_size * 2 > MAXMEMORY_BYTE)
        return 0;
    int befor = 0;
    for (int i = 8; i <= MAXMEMORY_BYTE / 2; i *= 2)
    {
        if (_size > befor && _size <= i)
        {
            return i;
        }
        befor = i;
    }

    //음수인 경우
    return -1;
}
```



new 요청 시 배정해 줄
메모리 주소

Current

현재는 A 주소가 들어감.
A 를 할당 해 주고
A의 8byte를 읽어와
다음 주소를 다시
current에 넣어줍니다.

MemoryPool

New/Delete

NEW

모든 블록은
MEMORY_BYTE=32768
만큼 할당 받고 해당 블록에서
쪼개서 사용합니다.

최초 할당 or 사용중인 블록이 용량X
=
새로운 블록 추가 할당

DELETE

반환된 memory를
memorypool에
반환합니다.

New

```
void* MemoryPool_2::operator new(size_t _size)
{
    char* next_ptr = nullptr;
    int size = AssignSize(_size);
    if (size == -1)
    {
        //뭔가 잘못 됨.
        return nullptr;
    }
    else if (size == 0)
    {
        // MAXMEMORY_BYTE/2 보다 값이 큰 경우 메모리풀을 생성하는게 낭비이기 때문에
        // 그 byte size 자체로 new 를 해준다.
        return malloc(_size);
    }
    t_pool_info* dummy=nullptr;
    if (memorypools.Find(size,dummy)==false)
    {
        //키 못찾음
        memorypools.Push(size, new t_pool_info(MAXMEMORY_BYTE, size));
    }
    else
    {
        //키 찾았는데 current 가 null 인 경우(이미 모두 할당한 경우 새로운 블록 추가로 받기)
        if (memorypools[size]->current == nullptr)
        {
            char** ptr;
            char* ptr2;
            memorypools[size]->current = (char*)malloc(MAXMEMORY_BYTE);
            memset(memorypools[size]->current, 0, MAXMEMORY_BYTE);

            next_ptr = (char*)memorypools[size]->current;
            memorypools[size]->startptr.push_back(memorypools[size]->current);

            for (int i = 1; i < MAXMEMORY_BYTE / size; i++)
            {
                ptr2 = memorypools[size]->current + (size * i);
                ptr = &ptr2;
                memcpy(next_ptr, ptr, sizeof(char*));
                next_ptr += _size;
            }
            ptr2 = nullptr;
            ptr = &ptr2;
            // 맨 마지막에 nullptr 넣기
            memcpy(next_ptr, ptr, sizeof(char*));
        }

        char* curptr = memorypools[size]->current;
        char* result = curptr;

        memcpy(&next_ptr, curptr, sizeof(char*));
        memorypools[size]->current = next_ptr;

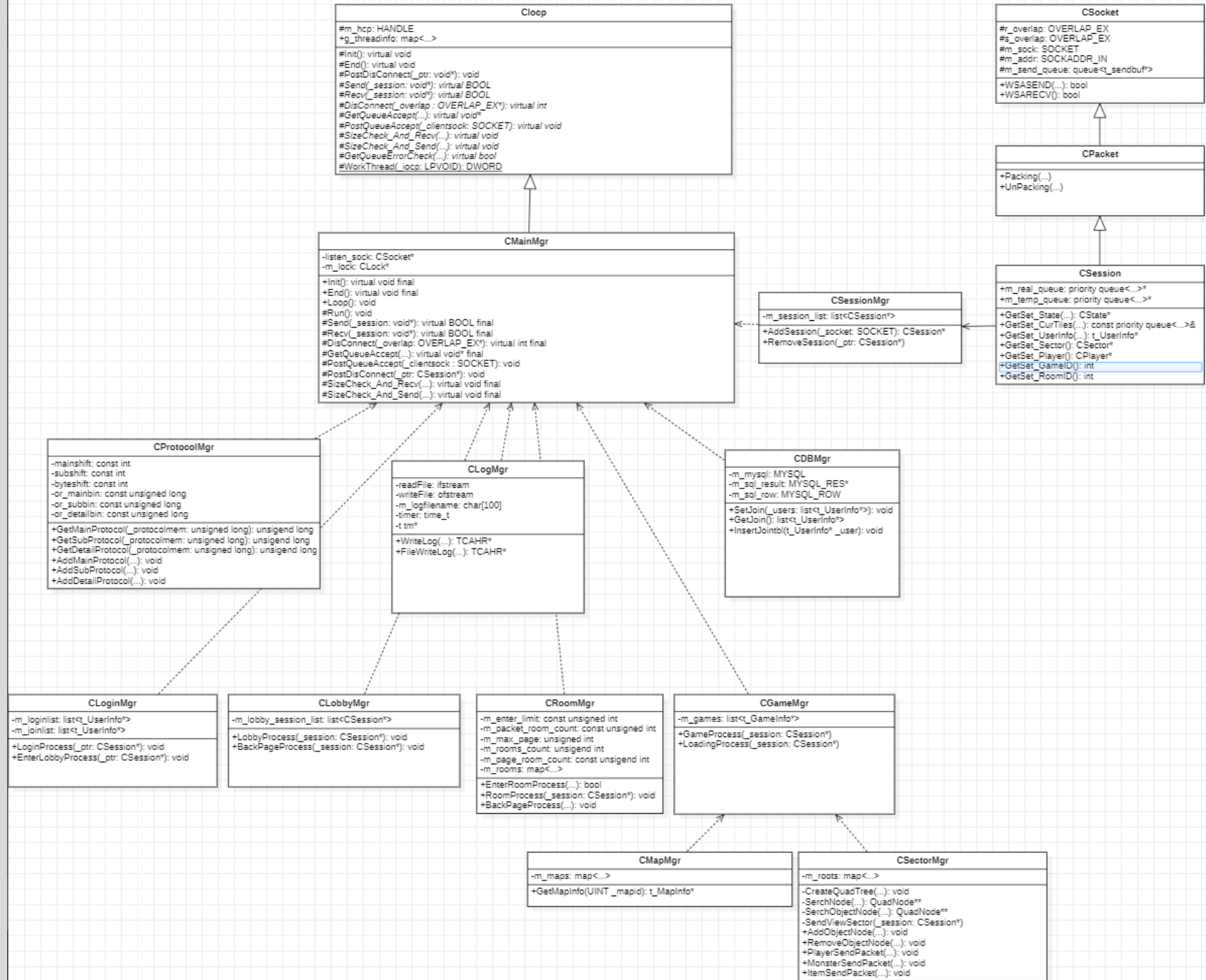
        return result;
    }
}
```

Delete

```
void MemoryPool_2::operator delete(void* _object, size_t _size)
{
    int size = AssignSize(_size);
    if (size == -1)
    {
        //뭔가 잘못 됨.
        return;
    }
    else if (size == 0)
    {
        // MAXMEMORY_BYTE/2 보다 값이 큰 경우 메모리풀을 생성하는게 낭비이기 때문에
        // 그 byte size 자체로 new 를 했기 때문에 그 메모리 자체를 delete 해준다.
        free(_object);
        return;
    }
    char* curptr = memorypools[size]->current;
    memcpy((char*)_object, &curptr, sizeof(char*));
    memorypools[size]->current = (char*)_object;
}
```

IOCP

전체 구조



IOCP 비동기화

IOCP를 사용하여
소켓 함수들을
비동기화

```
DWORD CIocp::WorkThread(LPVOID _iocp)
{
    int retval;

    CIocp* ciocp = reinterpret_cast<CIocp*>(_iocp);
    t_ThreadInfo* myinfo = new t_ThreadInfo();
    g_threadinfo.Push(GetCurrentThreadId(), myinfo);

    while (1)
    {
        DWORD cbTransferred;
        SOCKET clientsock;
        OVERLAP_EX* overlap_ptr;
        void* session;
        retval = GetQueuedCompletionStatus(ciocp->m_hcp, &cbTransferred, &clientsock, (LPOVERLAPPED*)&overlap_ptr, INFINITE);

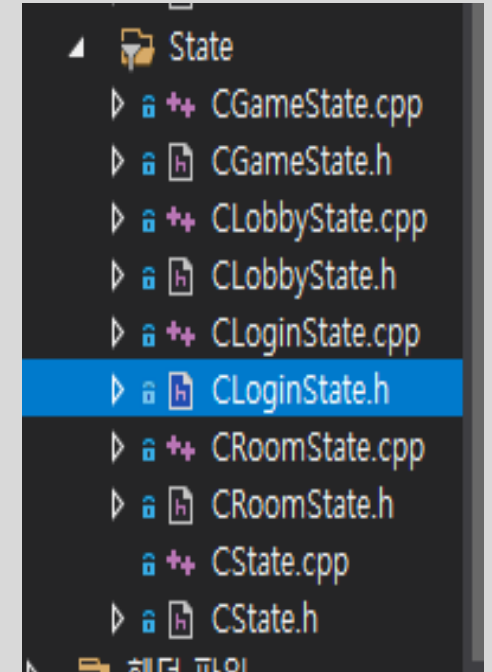
        bool check = ciocp->GetQueueErrorCheck(retval, cbTransferred, overlap_ptr);
        // check에서 에러체크후 오류발생시 type을 disconnected로 바꾸어준다.
        switch (overlap_ptr->type)
        {
            case IO_TYPE::ACCEPT:
                session = ciocp->GetQueueAccept(clientsock, overlap_ptr);
                ciocp->Recv(session);
                break;
            case IO_TYPE::RCV:
                session = overlap_ptr->session;
                ciocp->SizeCheck_And_Recv(session, cbTransferred, myinfo); // 여기서 함수하나에 전부 처리하도록
                break;
            case IO_TYPE::SEND:
                session = overlap_ptr->session;
                ciocp->SizeCheck_And_Send(session, cbTransferred, myinfo);
                break;
            case IO_TYPE::DISCONNECT:
                ciocp->DisConnect(overlap_ptr);
                break;
        }
    }

    return 0;
}
```

IOCP State 패턴

State 패턴을 이용하여
현재 요청 들어온
client(Session)의
현재 상태에 따라
수행할 명령을 다르게 합니다.

```
IOCP_Server (전역 범위)
1 #pragma once
2
3 class CSession;
4
5 class CState
6 {
7 public:
8     CState(CSession* _session) { m_session = _session; }
9     virtual void Recv(t_ThreadInfo* _threadinfo) = 0;
10    virtual void Send(t_ThreadInfo* _threadinfo) = 0;
11 protected:
12     CSession* m_session;
13 };
14
15
```



```
CSession* GetSession() { ... }
CState* GetState() { return m_curstate; }
CState* GetLoginState() { ... }
CState* GetLobbyState() { ... }
CState* GetRoomState() { ... }
CState* GetGameState() { ... }
void SetState(CState* _state) { ... }
void SetPlayer(int index) { ... }
```


Dump

에러 발생에 대한 Dump 파일
생성 및 Text 파일로 필요한
내용 기록하도록 했습니다.

저는 그때의 메모리 상황과 어떤 패킷에 대해
어떤 처리를 하다 오류가 발생 했는지의 정보를
알기 위해 State , Protocol 정보를 남겼습니다.

Exception_Handler.h

```
#pragma once
#include "pch.h"

#pragma warning(push)
#pragma warning(disable : 4091)
#include <DbgHelp.h>
#pragma warning(pop)
#include "Clock.h"
#include "ClockGuard.h"

#pragma comment(lib, "Dbghelp.lib")

class Exception_Handler
{
private:
    static Exception_Handler* volatile m_instance;
    static Clock* m_lock;

    Exception_Handler() {}
    Exception_Handler(const Exception_Handler& other);
    ~Exception_Handler() {}

public:
    static void volatile Create()
    {
        if (m_instance == nullptr)
        {
            ClockGuard<Clock> lock(m_lock);
            if (m_instance == nullptr)
            {
                m_instance = new Exception_Handler();
            }
        }
    }

    static Exception_Handler* volatile Instance()
    {
        return m_instance;
    }

    static void volatile Destroy()
    {
        delete m_instance;
        delete m_lock;
    }

public:
    DWORD initialize(_in LPCTSTR dump_file_name,
        _in const MINIDUMP_TYPE dump_type = MINIDUMP_TYPE::MiniDumpNormal);
    DWORD run();
    static LONG WINAPI Exception_Callback(_in struct _EXCEPTION_POINTERS* exceptioninfo);

private:
    wstring _dump_file_name;
    MINIDUMP_TYPE _dump_type;
    LPTOP_LEVEL_EXCEPTION_FILTER _prev_filter;
};
```

[2022_9_4] Log.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

[날짜:2022년9월4일] [시간: 1시 23분]

memory info

dwMemoryLoad: 0

dwTotalPhys: 8364258

dwAvailPhys: 1957472

dwTotalPageFile: 14036016

dwAvilPageFile: 1591052

dwTotalVirtual: 4294967232

dwAvailVirtual: 4292810040

thread info

IO_TYPE: RECV

E_STATE: LOGIN

MAINPROTOCOL: LOGIN

SUBPROTOCOL: LoginInfo

DETAILPROTOCOL: DETAILPROTOCOL 없음

server_exception.dmp

2022-09

Text log 와 dump 파일

LONG __stdcall Exception_Handler::Exception_Callback(_EXCEPTION_POINTERS* exceptioninfo)

```
{
    do { ... } while (false);
```

```
    MEMORYSTATUS memoryinfo;
```

```
    GlobalMemoryStatus(&memoryinfo);
```

```
    t_ThreadInfo* curthread = CIocp::g_threadinfo[::GetCurrentThreadId()];
```

```
    wstring iotype = IOTYPEtoString(curthread->iotype);
```

```
    wstring state = ESTATETOString(curthread->cur_state);
```

```
    wstring main, sub, detail;
```

```
    PROTOCOLtoString(curthread->protocol, curthread->cur_state, &main, &sub, &detail);
```

```
    CLogMgr::GetInstance()->
```

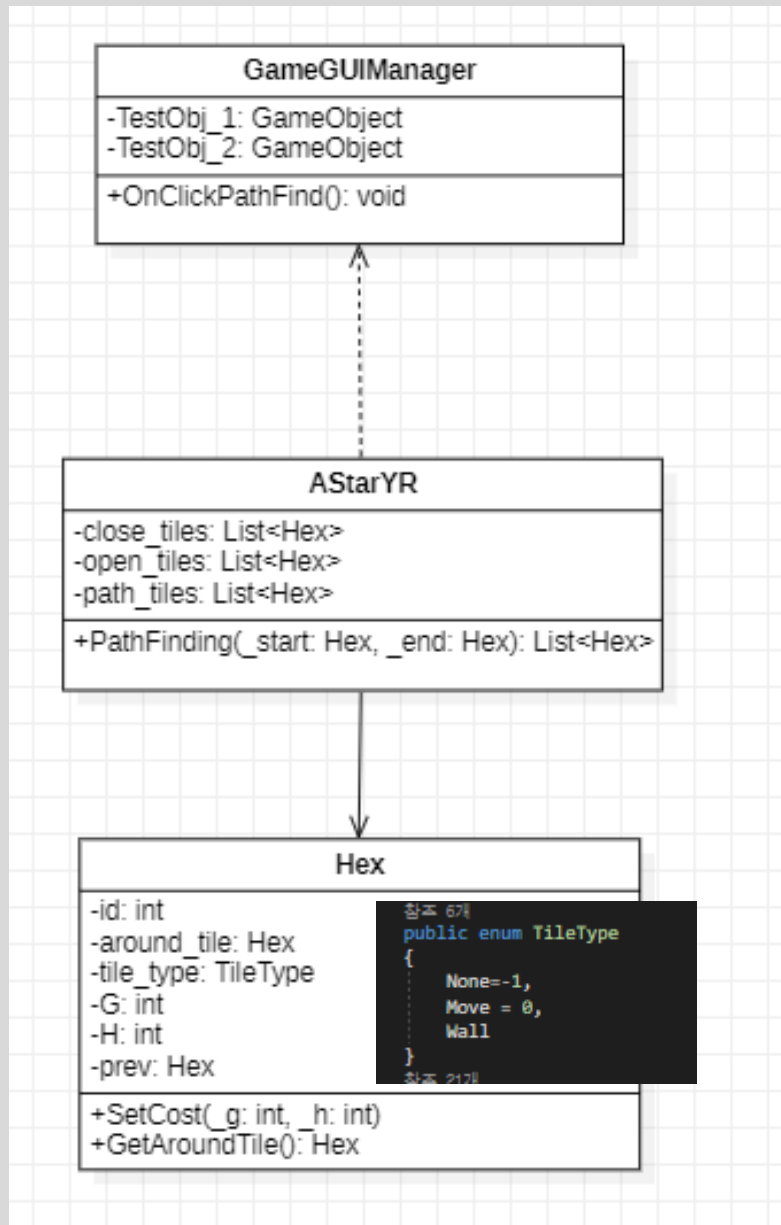
```
        FileWriteLog(_T("\n\nmemory info\ndwMemoryLoad: %lu\ndwTotalPhys: %lu\ndwAvailPhys: %lu\ndwTotalPageFile: %lu\ndwAvilPageFile: %lu\ndwTotalVirtual: %lu\ndwAvailVirtual: %lu\n",
            memoryinfo.dwMemoryLoad/(1024*2), memoryinfo.dwTotalPhys / (1024 * 1024), memoryinfo.dwAvailPhys / (1024 * 1024), memoryinfo.dwTotalPageFile / (1024 * 1024), memoryinfo.dwAvilPageFile / (1024 * 1024), memoryinfo.dwTotalVirtual / (1024 * 1024), memoryinfo.dwAvailVirtual / (1024 * 1024)));
```

```
    return (Exception_Handler::Instance()->_prev_filter) ? Exception_Handler::Instance()->_prev_filter(exceptioninfo) : EXCEPTION_EXECUTE_HANDLER;
}
```

사용 언어 : C++

A*

프로젝트 초기에
전체적인 기획안을 듣고
Boss, Monster가 Player를
추적해야 하는 기능이 필요하다고
여겨져 A*를 구현하였습니다.



```
public List<Hex> PathFinding(Hex _start, Hex _end)
{
    TileClear();
    start_tile = _start;
    end_tile = _end;
    open_tiles.Add(start_tile);
    Hex current = start_tile;
    start_tile.SetCost(-1, Mathf.Abs(start_tile.GetX - end_tile.GetX) + Mathf.Abs(start_tile.GetY - end_tile.GetY));

    while (open_tiles.Any())
    {
        current = open_tiles[0];
        foreach (Hex t in open_tiles)
        {
            if(t.GetF<current.GetF||t.GetF==current.GetF&& t._H<current._H)
            {
                current = t;
            }
        }
        if(current.ID==end_tile.ID)
        {
            GetPathList(end_tile);
            return path_tiles;
        }
        close_tiles.Add(current);
        open_tiles.Remove(current);
        SetMovePrice(current);
    }
    return null;
}
```

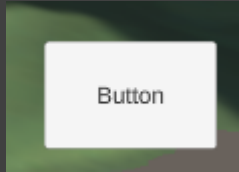
AStarYR.cs

```
참조 07개
public void OnClickPathFind()
{
    #region
    move_pos = a_star.PathFinding(start,end);

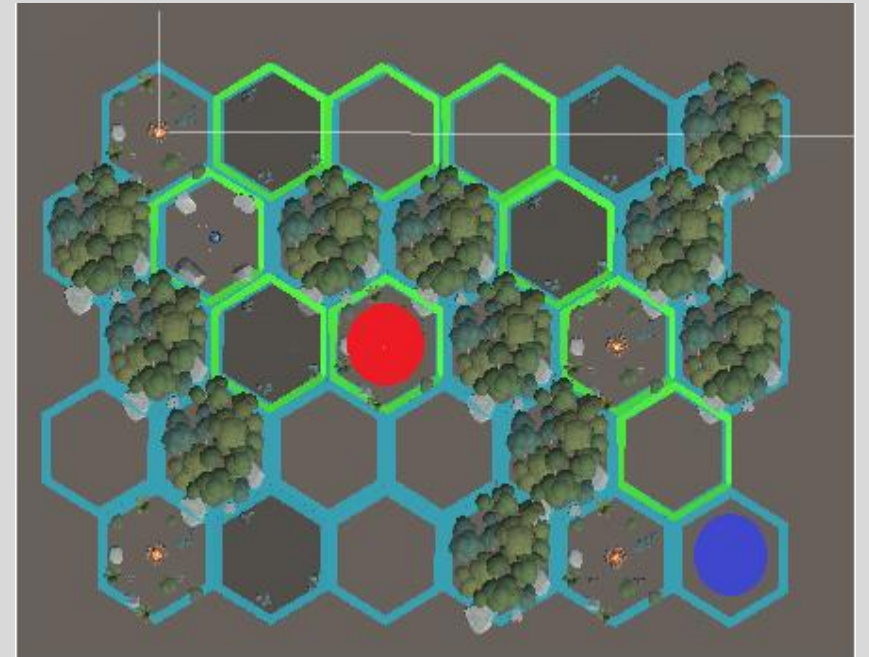
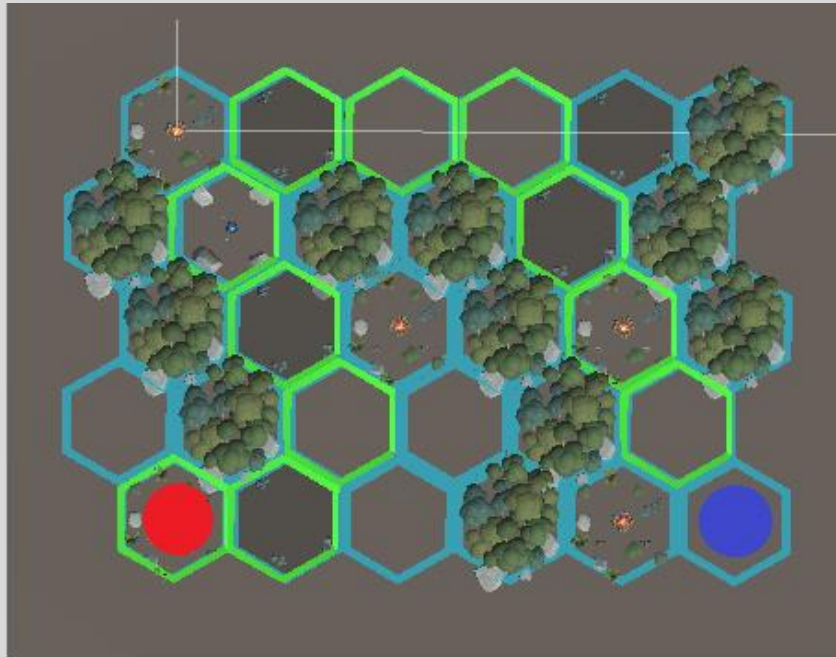
    foreach (var tile in move_pos)
    {
        MapManager.Instance.DrawLine(tile);
    }
}
```

GameGUIManager.cs

A*



테스트를 위해
버튼을 누르면
출발지에서 목적지까지
의 경로를 계산해서 출
력하게 했습니다.



빨간 원 = 목표 지점
파란 원 = 출발 지점

사용 언어 : C#

Mouse Manager

해당 기능 프로젝트 사용 X

Mouse event를
직접 구현해서 써보고
싶다는 생각에 공부 목적
으로 만들었습니다.

해당 기능을 토대로
임의의 인벤토리 기능을
만들어 사용하였습니다.

```
MouseManager.cs
using UnityEngine.EventSystems;
using UnityEngine.UI;
using System;

//원,오 다운,드래그 만 구현됨.
public enum TAG_TYPE
{
    NONE,
    MenuStart,
    MAX
}

public enum MOUSE_TYPE
{
    NONE,
    LEFTDOWN_BTN,
    LEFTUP_BTN,
    LEFTDRAG_BTN,
    RIGHTDOWN_BTN,
    RIGHTUP_BTN,
    MAX
}

// 이벤트 호출시 넘겨줄 마우스 정보 구조체.
public struct MouseArgs
{
    public Vector2 beforPos;
    public Vector2 currentPos;
    public GameObject beforGameObject;
    public TAG_TYPE befortag;
    public bool is_LeftDown;
    public bool is_RightDown;
}

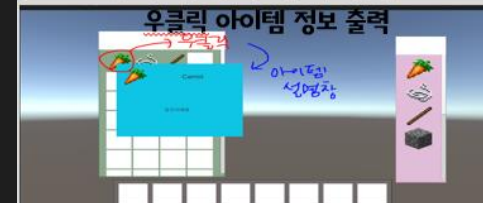
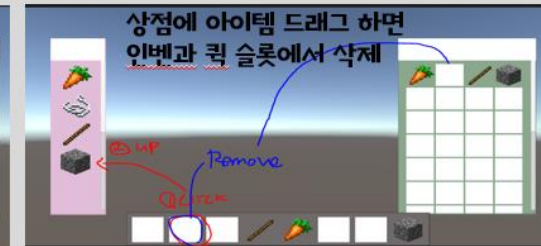
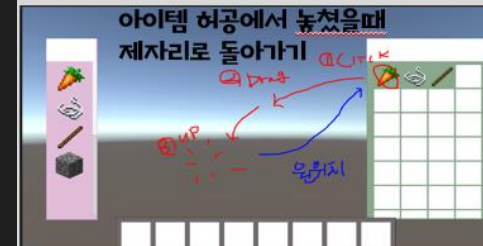
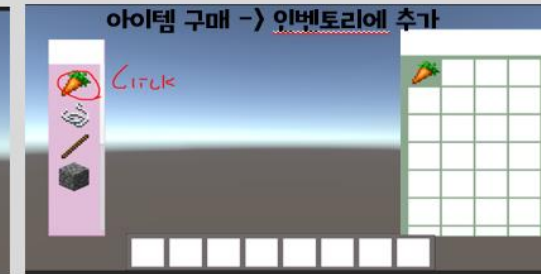
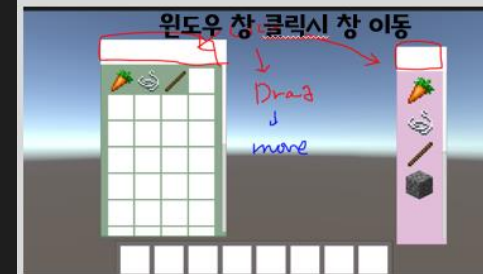
public class MouseManager : Singleton_Ver2.Singleton<MouseManager>
{
    [SerializeField]
    Canvas m_canvas;
    GraphicRaycaster m_Raycaster;
    PointerEventData m_PointerEventData;
    EventSystem m_EventSystem;

    public delegate void MouseEventHandler(GameObject _selected, MouseArgs args);
    Dictionary<MOUSE_TYPE, Dictionary<TAG_TYPE, List<MouseEventHandler>>>> MouseEvents;
    Dictionary<MOUSE_TYPE, List<TAG_TYPE>>> MouseTagList;

    GameObject m_currentGameObject;
    MouseArgs m_current_args;
    float current_mousedown_time;

    TAG_TYPE m_currenttag;
    bool m_isFind;
    object temptag;
    //마우스 이벤트 등록
    public void MouseEvent_Register(MOUSE_TYPE p_mouse, TAG_TYPE p_tag, MouseEventHandler p_event){...}

    public Vector3 MousePos{...}
    public void Init(){...}
    private void Awake(){...}
    void Start(){...}
    private void Update(){...}
}
```



감사합니다!