

# 포트폴리오

클라이언트 프로그래머 지원 최예람

# CONTENTS

---

---

## IOCP

전체 구조

비동기화

State 패턴

## Map (Red Black Tree)

## MemoryPool

## Render Sector

Sector 구조

Quad Tree

LRU Queue 구현

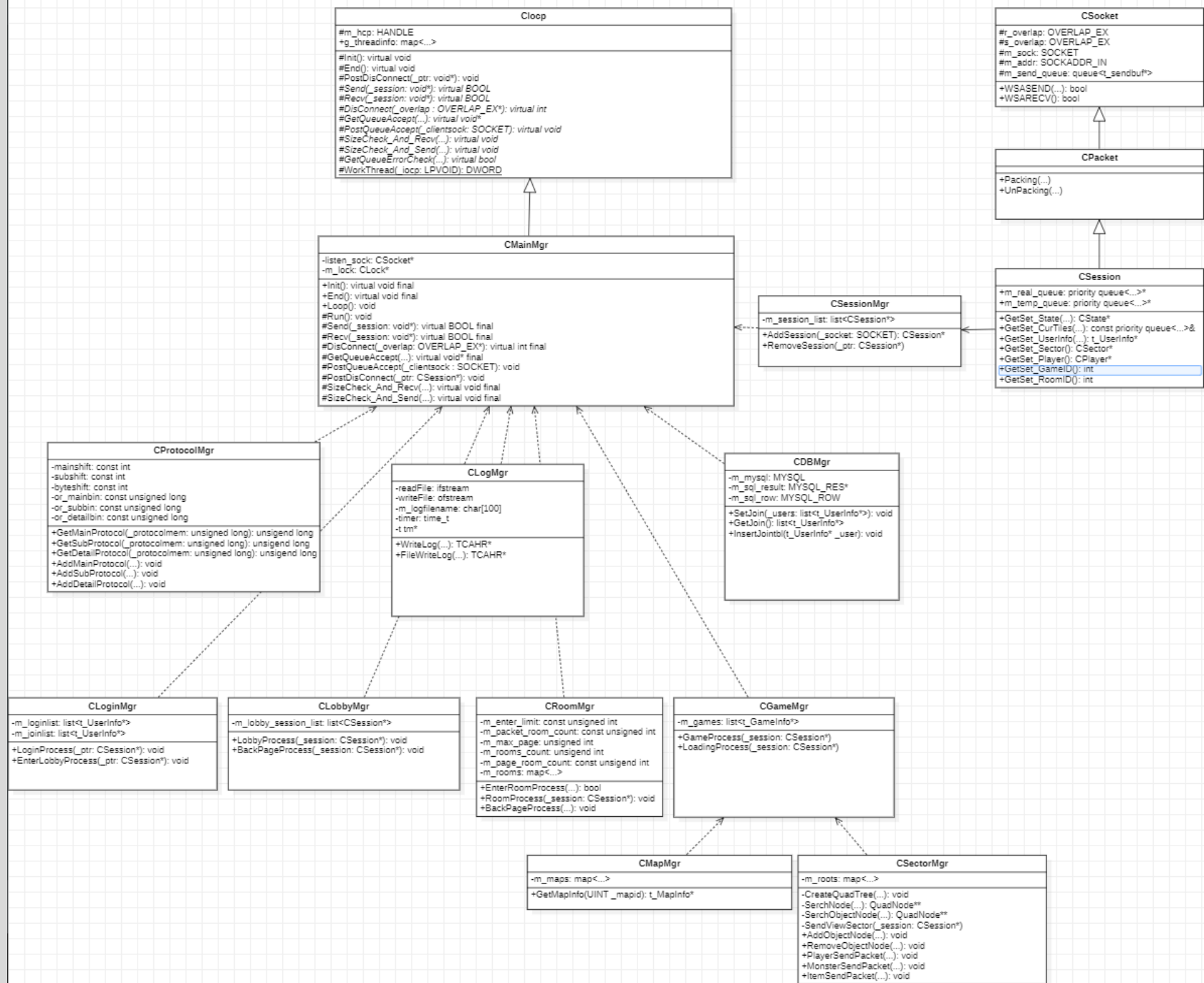
## Dump

## 팀 프로젝트 참여

---

# 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);

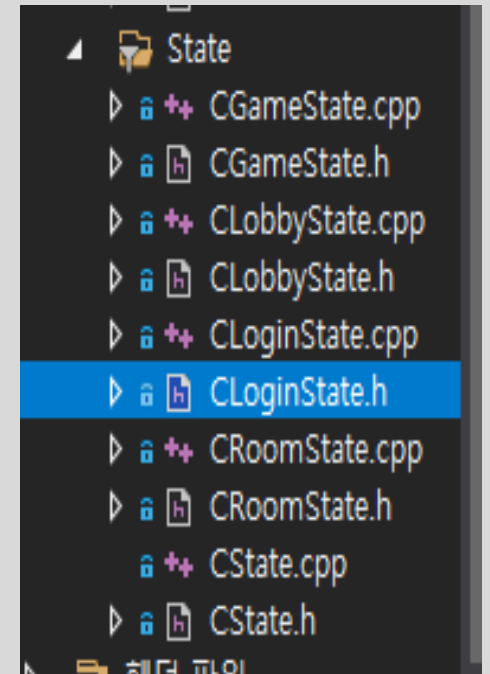
        bool check = ciocp->GetQueueErrorCheck(retval, cbTransferred, overlap_ptr);
        // check에서 예러체크후 오류발생시 type을 disconnected로 바꾸어준다.
        switch (overlap_ptr->type)
        {
            case IO_TYPE::ACCEPT:
                session = ciocp->GetQueueAccept(clientsock, overlap_ptr);
                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 명령  
을 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) { ... }
```

# Map

## Red-Black-Tree Insert

Insert 시 주의해야 할  
case 5가지에 대해  
구현하여 **RBT\_Insert()**에서  
처리 하도록 했습니다.

```
#pragma region insert func
//case 1 : 부모와 삼촌이 레드일 때
// => 부모와 삼촌을 black 으로 바꾸고 root node가 red 인지 검사, red 면 black으로 변경.
void insert_case1(Node<Key, Value>** _node){ ... }
//case 2 : 왼쪽에 push 됐을 때 부모의 오른쪽 push 그리고 부모가 레드 , 삼촌은 블랙
// => 왼쪽으로 회전.
void insert_case2(Node<Key, Value>** _node){ ... }
//case 3 : 왼쪽에 push 됐을 때 부모의 왼쪽 push 그리고 부모가 레드 , 삼촌은 블랙
void insert_case3(Node<Key, Value>** _node){ ... }
//case 4 : 오른쪽 push 됐을 때 부모의 왼쪽 push 그리고 부모가 레드 , 삼촌은 블랙
void insert_case4(Node<Key, Value>** _node){ ... }
//case 5 : 오른쪽 push 됐을 때 부모의 오른쪽 push 그리고 부모가 레드, 삼촌은 블랙
void insert_case5(Node<Key, Value>** _node){ ... }

void RBT_Insert(Node<Key, Value>** _node)
{
    if ((*_node) == nilnode)
    {
        return;
    }
    //root node 일 경우
    else if ((*_node)->parent == nullptr)
    {
        (*_node)->type = NodeType::Black;
        return;
    }

    Node<Key, Value>** grand_parent = GetGrandParent(_node);
    Node<Key, Value>** uncle = GetUncle(_node);

    if ((uncle != nullptr) && (*_node)->parent->type == NodeType::Red && (*uncle)->type == NodeType::Red)
    {
        insert_case1(_node);
    }
    else if ((uncle != nullptr) && (*_node)->parent->type == NodeType::Red && (*uncle)->type == NodeType::Black)
    {
        //왼쪽 부모에 push
        if ((*_node)->parent == (*grand_parent)->left)
        {
            //부모의 오른쪽에 push
            if ((*_node)->parent->right == (*_node))
            {
                insert_case2(_node);
            }
            //부모의 왼쪽에 push
            else
            {
                insert_case3(_node);
            }
        }
        //오른쪽 부모에 push
        else
        {
            if ((*_node)->parent->left == (*_node))
            {
                insert_case4(_node);
            }
            else
            {
                insert_case5(_node);
            }
        }
    }
}

#pragma endregion
```

# Map

## Red-Black-Tree Delete

Delete시 주의해야 할  
case 4가지에 대해  
구현하여 **RBT\_Delete()**에서  
처리 하도록 했습니다.

```
#pragma region delete func
/* ... */
//case 4 - r
void delete_case4_r(Node<Key, Value>** _node, Node<Key, Value>* parent, Node<Key, Value>* sibling){ ... }
//case 4 - l
void delete_case4_l(Node<Key, Value>** _node, Node<Key, Value>* parent, Node<Key, Value>* sibling){ ... }
#pragma endregion
#pragma region case 3
/* ... */
//case 3 - r
void delete_case3_r(Node<Key, Value>** _node, Node<Key, Value>* parent, Node<Key, Value>* sibling){ ... }
//case 3 - l
void delete_case3_l(Node<Key, Value>** _node, Node<Key, Value>* parent, Node<Key, Value>* sibling){ ... }
#pragma endregion
#pragma region case 2
/* ... */
//case 2
void delete_case2(Node<Key, Value>** _node, Node<Key, Value>* parent, Node<Key, Value>* sibling){ ... }
#pragma endregion
#pragma region case 1
/* ... */
//case 1 - r
void delete_case1_r(Node<Key, Value>** _node, Node<Key, Value>* parent, Node<Key, Value>* sibling){ ... }
//case 1 - l
void delete_case1_l(Node<Key, Value>** _node, Node<Key, Value>* parent, Node<Key, Value>* sibling){ ... }
#pragma endregion
void RBT_Delete(Node<Key, Value>** _node)
{
    Node<Key, Value>* parent = (*_node)->parent;
    Node<Key, Value>* sibling = nullptr;

    // 현재 노드가 부모기준 왼쪽이다.
    if (parent->left == (*_node))
    {
        sibling = parent->right;
        if (sibling->type == NodeType::Black)
        {
            /* 자식 둘다 Red
            if (sibling->left->type == NodeType::Red && sibling->right->type == NodeType::Red)
            {
                Children_Alter_Black(parent, sibling->left, sibling->right);
                delete_case1_l(_node, parent, sibling);
            }
            //오른쪽만 Red
            else if (sibling->right->type == NodeType::Red)
            {
                delete_case4_l(_node, parent, sibling);
            }
            //왼쪽만 Red
            else if (sibling->left->type == NodeType::Red)
            {
                delete_case3_l(_node, parent, sibling);
            }
            /* 형제: 검정
            형제의 자식1 : 검정
            형제의 자식 2 : 검정
            */
            else
            {
                delete_case2(_node, parent, sibling);
            }
        }
        else // 형제가 red 일 때 case 1
        {
            delete_case1_l(_node, parent, sibling);
        }
    }
    else
    {
        sibling = parent->left;
        if (sibling->type == NodeType::Black)
        {
            /* 자식 둘다 Red
            if (sibling->left->type == NodeType::Red && sibling->right->type == NodeType::Red)
            {
                Children_Alter_Black(parent, sibling->left, sibling->right);
                delete_case1_r(_node, parent, sibling);
            }
            //왼쪽만 Red
            else if (sibling->left->type == NodeType::Red)
            {
                delete_case4_r(_node, parent, sibling);
            }
            //오른쪽만 Red
            else if (sibling->right->type == NodeType::Red)
            {
                delete_case3_r(_node, parent, sibling);
            }
            /* 형제: 검정
            형제의 자식1 : 검정
            형제의 자식 2 : 검정
            */
            else
            {
                delete_case2(_node, parent, sibling);
            }
        }
        else // 형제가 red 일 때 case 1
        {
            delete_case1_r(_node, parent, sibling);
        }
    }
}
#pragma endregion
```

```
else
{
    sibling = parent->left;
    if (sibling->type == NodeType::Black)
    {
        /* 자식 둘다 Red
        if (sibling->left->type == NodeType::Red && sibling->right->type == NodeType::Red)
        {
            Children_Alter_Black(parent, sibling->left, sibling->right);
            delete_case1_r(_node, parent, sibling);
        }
        //왼쪽만 Red
        else if (sibling->left->type == NodeType::Red)
        {
            delete_case4_r(_node, parent, sibling);
        }
        //오른쪽만 Red
        else if (sibling->right->type == NodeType::Red)
        {
            delete_case3_r(_node, parent, sibling);
        }
        /* 형제: 검정
        형제의 자식1 : 검정
        형제의 자식 2 : 검정
        */
        else
        {
            delete_case2(_node, parent, sibling);
        }
    }
    else // 형제가 red 일 때 case 1
    {
        delete_case1_r(_node, parent, sibling);
    }
}
#pragma endregion
```

# Map

Red-Black-Tree  
Delete

구현한 STL을 map으로 사용.

Clocp.h

= 스레드ID, 스레드 정보

CMapMgr.h

= 맵ID, 맵 정보

CSectorMgr.h

= SectorID, root Node주소

MemoryPool\_2.h

= 메모리 크기,

생성된 memorypool 정보

```
UDLLC:  
    static RBT<DWORD, t_ThreadInfo*> g_threadinfo;  
protected:
```

Clocp.h ↗ ✕

```
RBT<UINT, t_MapInfo*> m_maps;
```

CMapMgr.h ↗ ✕

```
RBT<UINT, QuadNode*> m_roots; // key = gameinfo id, value = rootnode
```

CSectorMgr.h ↗ ✕

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

MemoryPool\_2.h ↗ ✕



# MemoryPool

## AssignSize()

AssignSize()로 들어온  
객체의 size를 보고  
어느 memorypool 블록을  
배정할지 결정해서  
Size를 반환 해줍니다.  
이 Size가 곧 map(RBT)의  
ID입니다.

```
#define MAXMEMORY_BYTE 32768

struct t_pool_info
{
    t_pool_info(const int _capacity, const int _size)
    {
        current = nullptr;
        char* next_ptr;
        char** ptr;
        char* ptr2;
        char* ptr3;

        current = (char*)malloc(_capacity);
        memset(current, 0, _capacity);

        next_ptr = (char*)current;
        startptr.push_back(current);

        for (int i = 1; i < _capacity/_size; i++)
        {
            ptr2 = 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*));
    }
    ~t_pool_info()
    {
        int size = startptr.size();
        for (int i = 0; i < size; i++)
        {
            char* item = startptr.back();
            startptr.pop_back();
            free(item);
        }
    }
    char* current;
    //이미 생성된 블록이 꼭 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;
};
```

```
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;
}
```

# MemoryPool

## New/Delete

### NEW

모든 블록은  
**MEMORY\_BYTE=32768**  
만큼 할당 받고 해당 블록에서  
쪼개서 사용합니다.

최초 할당  
or  
이미 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;
}
```

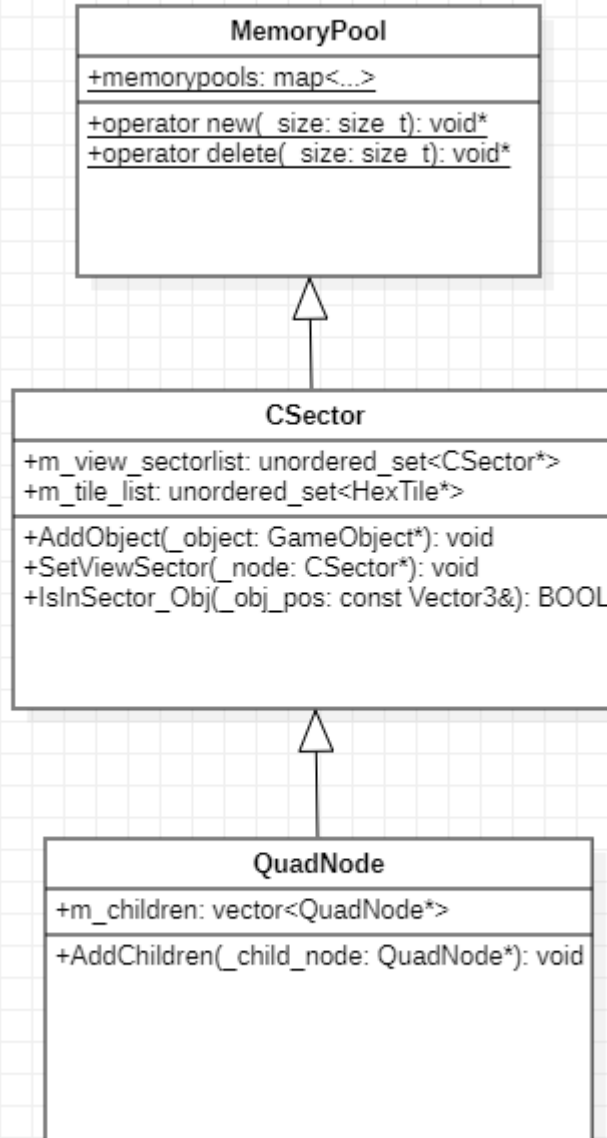
# Render Sector

## 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.



```
#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; // 밀변/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

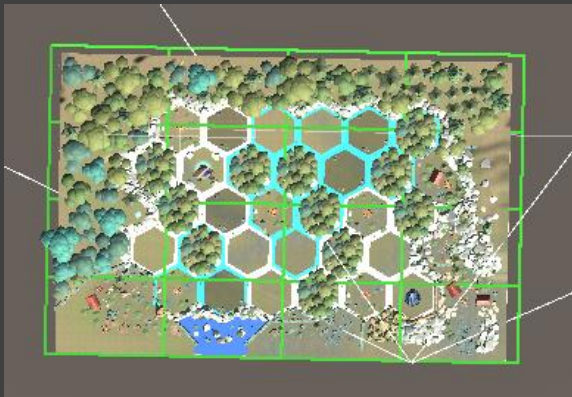
    int Child_Size();
private:
    static int create_count;
    int m_id;
    vector<QuadNode*> m_children; // 자식노드
    QuadNode* m_parent; // 부모노드
    BOOL m_is_culled; // 컬링 여부
    const float m_limit_depth = 4; // 트리의 최대 깊이
    E_NodeType m_type;
};
```

# Render Sector

## Quad Tree

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

Depth값을 주어 해당 값  
수정 시 트리가 4의 depth승  
만큼 구역이 쪼개집니다.  
유효한 데이터는 제일 깊이 있는  
노드들 이며, 상위 노드들은 구  
역의 의미만을 갖습니다.



초록선=sector / 현재 depth:2

```
void CSectorMgr::CreateQuadTree(t_GameInfo* _gameinfo, t_MapInfo* _mapinfo)
{
    Vector3 distance((_mapinfo->m_h_mapsize), 0, (_mapinfo->m_v_mapsize));
    Vector3 senter_pos(_mapinfo->m_start_position.x + (distance.x / 2), _mapinfo->m_default_y, _mapinfo->m_start_position.z - (distance.z / 2));
    distance.x /= 2;
    distance.z /= 2;
    QuadNode* root = new QuadNode(Vector3(), distance);

    CreateChildren(root, senter_pos, distance, 0, _gameinfo);

    {
        CLockGuard<CLock> lock(m_lock);
        m_roots.Push(_gameinfo->m_id, root);
    }

    //view sector list 도 setting 하는거 만들기
    SetViewNode(root, 0, _mapinfo, _gameinfo->m_id);
}

void CSectorMgr::CreateChildren(QuadNode* _parent, Vector3 _senterpos, Vector3 _distance, int _curdepth, t_GameInfo* _gameinfo)
{
    t_MapInfo* _mapinfo = _gameinfo->m_mapinfo;
    if (_mapinfo->m_depth == _curdepth)
    {
        //leaf node id
        _parent->SetID();
        /*Vector3 vec = _parent->GetDistance();
        vec.x += 1;
        vec.z += 1;
        _parent->SetDistance(vec);*/
        _gameinfo->m_leaf_nodes.Push(_parent->GetID(), _parent);
        return;
    }

    Vector3 distance(_distance.x / _mapinfo->m_squared_value, 0, _distance.z / _mapinfo->m_squared_value);
    Vector3* senterpos = nullptr;
    QuadNode* child = nullptr;
    for (int i = 0; i < _mapinfo->m_squared_value * 2; i++)
    {
        switch (i)
        {
            case 0:// left up
                senterpos = new Vector3(_senterpos.x - distance.x, _mapinfo->m_default_y, _senterpos.z + distance.z);
                child = new QuadNode(*senterpos, distance);
                break;
            case 1://right up
                senterpos = new Vector3(_senterpos.x + distance.x, _mapinfo->m_default_y, _senterpos.z + distance.z);
                child = new QuadNode(*senterpos, distance);
                break;
            case 2://left down
                senterpos = new Vector3(_senterpos.x - distance.x, _mapinfo->m_default_y, _senterpos.z - distance.z);
                child = new QuadNode(*senterpos, distance);
                break;
            case 3://right down
                senterpos = new Vector3(_senterpos.x + distance.x, _mapinfo->m_default_y, _senterpos.z - distance.z);
                child = new QuadNode(*senterpos, distance);
                break;
        }

        if (child == nullptr)
        {
            //메모리할당 실패
            return;
        }

        _parent->AddChildren(child);
        CreateChildren(child, *senterpos, distance, _curdepth + 1, _gameinfo);
        if (senterpos != nullptr)
            delete senterpos;
    }
}

#pragma region viewsector setting ver 1
void CSectorMgr::SetViewNode(QuadNode* _parent, int _curdepth, t_MapInfo* _mapinfo, UINT _gameid)
{
    if (_mapinfo->m_depth == _curdepth)
    {
        Vector3 start = _parent->GetStartPos();
        Vector3 distance = _parent->GetDistance();
        if (_parent->GetID() == 0)
        {
            printf("id=8");
        }
        Vector3 position;
        QuadNode** viewnode = nullptr;

        //시야의 최대치는 8이다.
        for (int i = 0; i < _mapinfo->m_eyesight; i++)
        {
            switch (static_cast<E_NodeType>(i))
            {
                case E_NodeType::Left:
                    //left node
                    position.x = start.x - distance.x*2;
                    position.y = start.y;
                    position.z = start.z;
                    break;
                case E_NodeType::LeftUp:
                    //left up node
                    position.z = start.z + distance.z*2;
                    break;
                case E_NodeType::LeftDown:
                    //left down node
                    position.z = start.z - distance.z*2;
                    break;
                case E_NodeType::Right:
                    //right node
                    position.x = start.x + distance.x*2;
                    position.y = start.y;
                    position.z = start.z;
                    break;
                case E_NodeType::RightUp:
                    //right up node
                    position.z = start.z + distance.z*2;
                    break;
                case E_NodeType::RightDown:
                    //right down node
                    position.z = start.z - distance.z*2;
                    break;
                case E_NodeType::Up:
                    //senter up
                    position.x = start.x;
                    position.y = start.y;
                    position.z = start.z + distance.z*2;
                    break;
                case E_NodeType::Down:
                    //senter down
                    position.z = start.z - distance.z*2;
                    break;
            }

            viewnode = SearchNode(m_roots[_gameid], position, 0, _mapinfo);
            if (viewnode == nullptr)
                continue;
            else
                _parent->SetViewSector(*viewnode);
        }

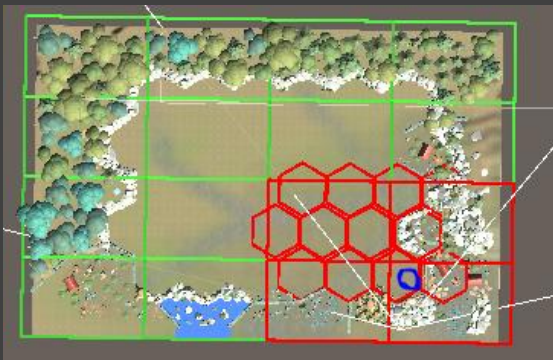
        _parent->SetViewSector(_parent);
        return;
    }
}
```

# 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;
}
```

```
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

```
void CSectorMgr::TestPlayerMove(CSession* _session, t_GameInfo* _gameinfo)
{
    CLockGuard<CLock> lock(m_lock);
    byte data[BUFSIZE];
    ZeroMemory(data, BUFSIZE);
    _session->UnPacking(data);
    Vector3 obj_pos;
    UnPacking(data, obj_pos);
    CPlayer* player = _session->GetPlayer();
    player->SetVector(obj_pos);
    QuadNode* sector = SerchObjectNode(_gameinfo, obj_pos);
    _session->SetSector(sector);
    TestSendViewSectorProcess(_session, _gameinfo, obj_pos);
    TestSendViewTileProcess(_session, _gameinfo, obj_pos);
}
```

```
void CSectorMgr::TestSendViewSectorProcess(CSession* _session, t_GameInfo* _gameinfo, Vector3 _objpos)
{
    /* ... */
    CLockGuard<CLock> lock(m_lock);
    unsigned long protocol = 0;
    CProtocolMgr::GetInst()->AddMainProtocol(&protocol, static_cast<unsigned long>(MAINPROTOCOL::TEST));
    CProtocolMgr::GetInst()->AddSubProtocol(&protocol, static_cast<unsigned long>(CGameMgr::SUBPROTOCOL::SECTOR));
    int test_sector_index = 0;

    list<Vector3> starts;
    Vector3 distance;

    QuadNode* sector = reinterpret_cast<QuadNode*>(*SerchObjectNode(m_roots[_gameinfo->m_id], _objpos, 0, _gameinfo->m_mapinfo));

    unordered_set<CSector*> viewlist = sector->GetViewSector();
    int count = 0;
    for (auto viewnode : viewlist)
    {
        if (count == 0)
        {
            distance = viewnode->GetDistance();
        }
        count++;
        starts.push_back(viewnode->GetStartPos());
    }
    Packing(protocol, starts, distance, _session);
}
```

```
void CSectorMgr::TestSendViewTileProcess(CSession* _session, t_GameInfo* _gameinfo, Vector3 _objpos)
{
    /* ... */
    CLockGuard<CLock> lock(m_lock);
    unsigned long protocol = 0;
    CProtocolMgr::GetInst()->AddMainProtocol(&protocol, static_cast<unsigned long>(MAINPROTOCOL::TEST));
    CProtocolMgr::GetInst()->AddSubProtocol(&protocol, static_cast<unsigned long>(CGameMgr::SUBPROTOCOL::Object));
    CProtocolMgr::GetInst()->AddDetailProtocol(&protocol, static_cast<unsigned long>(CGameMgr::DETAILPROTOCOL::Tile));
    list<Vector3> starts;
    Vector3 distance;

    /* ... */
    LRU_Queue<HexTile*, mygreater> render_queue = _session->GetCurTiles();
    int size = render_queue.Size();
    for (int i = 0; i < size; i++)
    {
        HexTile* tile = render_queue[i];
        Vector3 pos = tile->GetSenderPos();
        starts.push_back(pos);
    }
    Packing(protocol, starts, distance, _session);
}
```

Player 움직임에 따른 sector정보와 tile 정보 전송



# Render Sector

## LRU Queue

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

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

구현된 우선순위 큐는  
LRU\_Queue Class에서  
한번 더 감싸 줍니다.

```
template<typename T, typename _Pr = greater<T>>, <T>:IntelSense에 대한 설명, 댓글 및 인수를 제공합니다. </>
class PriorityQueue
{
protected:
    _Pr mycompare;
    Vector<T> contain;
private:
    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);
        }
    }
public:
    PriorityQueue(int capacity=40){...}
    void Push(T _data)
    {
        VecPush(_data);
        //VecSort();
    }
    void Pop()
    {
        int size = contain.size() - 1;
        T data = contain[size];
        for (auto itr = contain.begin(); itr != contain.end(); itr++)
        {
            if (data == *itr)
            {
                contain.erase(itr);
                break;
            }
        }
    }
    // 나중에 참조로 변경해야 함.
    T Front()
    {
        T temp = contain[0];
        int size = contain.size() - 1;
        contain[0] = contain[size];
        contain[size] = temp;
        int root = 0;
        int child = 1;
        do
        {
            child = root * 2 + 1;
            if (child < size - 1 && this->mycompare(contain[child], contain[child + 1]))
            {
                child++;
            }
            if (child < size && this->mycompare(contain[root], contain[child]))
            {
                temp = contain[root];
                contain[root] = contain[child];
                contain[child] = temp;
            }
            root = child;
        } while (child < size);

        T data = contain[size];
        return data;
    }
    size_t Size(){...}
    bool Empty(){...}
    const T& operator[](int _index)
    {
        return contain[_index];
    }
};
```

PriorityQueue.h

```
template<typename T,typename _Pr=greater<T>>
class LRU_Queue
{
public:
    LRU_Queue(int _capacity):capacity(_capacity){ queue = new PriorityQueue<T,_Pr>(_capacity); }
    bool Push(T _data, T& outdata)
    {
        bool flag = false;
        if (queue->Size() == capacity)
        {
            outdata = queue->Front();
            queue->Pop();
            flag = true;
        }
        queue->Push(_data);
        return flag;
    }
    bool Empty()
    {
        return queue->Empty();
    }
    T Front()
    {
        return queue->Front();
    }
    void Pop()
    {
        queue->Pop();
    }
    int Size()
    {
        return queue->Size();
    }
    /*컨테이너의 값만 빼오는것이기에 정렬 x, 삭제 x 요소의 값을 출력만 한다.
    lru에서 관리하는 render중간 타일에 대한 정보를 얻기 위해서 만들었다.*
    const T& operator[](int _index)
    {
        return (*queue)[_index];
    }
protected:
    PriorityQueue<T,_Pr>* queue;
private:
    int capacity;
};
```

LRU\_Queue.h

# Render Sector

## LRU Queue

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

오른쪽 이미지 두개는  
Queue에서 Tile 정보를 제외  
하는 이미지 입니다.

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

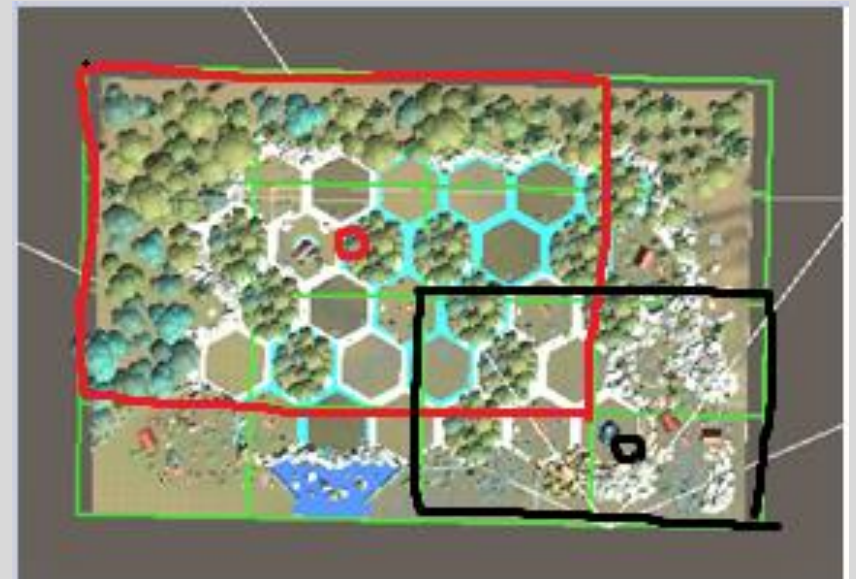
    bool same = false;
    unordered_set<HexTile*> tilelist;
    list<HexTile*> suc_tile;
    list<HexTile*> new_tile;
    while (m_real_queue->Empty() == false)
    {
        same = false;
        HexFile* temp = m_real_queue->Front();
        m_real_queue->Pop();
        for (auto sector : viewlist)
        {
            tilelist = sector->GetFileList();
            auto itr = tilelist.find(temp);
            if (itr != tilelist.end())
            {
                //리얼 큐랑 타일리스트 둘다 있는 경우
                //중복데이터면 시간경신해서 넣기
                temp->SetRenderTime(last_update_time);
                HexFile* data = nullptr;
                if (m_temp_queue->Push(temp, data))
                {
                    cout << data->GetTime() << " 이거 지워짐\n";
                    delete data;
                }
                suc_tile.push_back(temp);
                same = true;
                break;
            }
        }
        if (!same)
        {
            //리얼 큐에 있고 타일 리스트에 없는 경우-원래 시간으로 넣기.
            //다 놓았는데 tilelist에 없었다. real queue에만 있었다.
            HexFile* 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)
    {
        tilelist = sector->GetFileList();
        for (auto suc : suc_tile)
        {
            auto itr = tilelist.find(suc);
            if (itr != tilelist.end())
            {
                tilelist.erase(suc);
            }
        }
        for (auto tile : tilelist)
        {
            tile->SetRenderTime(last_update_time);
            HexFile* data = nullptr;
            if (m_temp_queue->Push(tile, data))
            {
                cout << data->GetID() << " | " << data->GetTime() << " 이거 지워짐\n";
                delete data;
            }
        }
    }

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

```
36|1663167027 이거 지워짐
37|1663167027 이거 지워짐
22|1663167027 이거 지워짐
23|1663167027 이거 지워짐
31|1663167027 이거 지워짐
38|1663167027 이거 지워짐
39|1663167027 이거 지워짐
12|1663167062 이거 지워짐
13|1663167062 이거 지워짐
14|1663167062 이거 지워짐
11|1663167062 이거 지워짐
21|1663167062 이거 지워짐
31|1663167062 이거 지워짐
```

TileID | 갱신 시간



# Dump

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

```
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(_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;
}
```



# 팀 프로젝트 참여

프로젝트 명: Yggdrasil  
개발 인원: 8명  
-클라이언트 프로그래머 3명  
-서버 프로그래머 2명

담당 업무:  
C++와 Unity의 서버 구현  
로그인/회원가입 DB 처리  
로그인 - 게임 입장까지의  
UI 클라이언트&서버 기능 구현.  
SectorManager 구현.

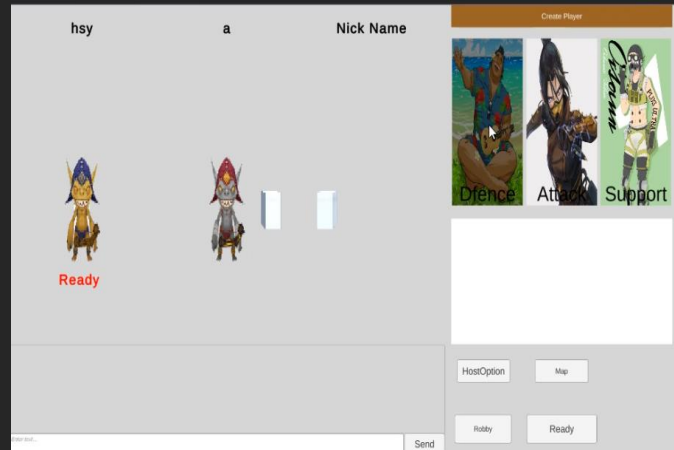


로그인/회원가입



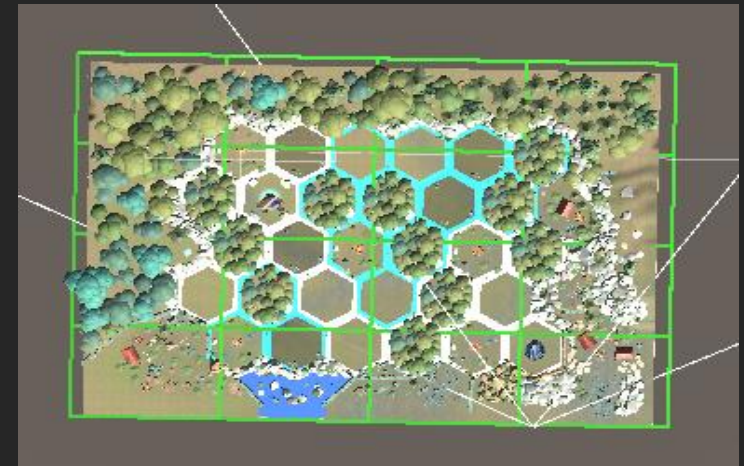
로비 기능

- 방 만들기
- 방입장
- 채팅
- 옵션 설정



방 기능

- 호스트/멤버 관리
- 레디
- 캐릭터 선택 등 처리..
- ( Ready 하면 해당 캐릭터 중복 선택 X)



게임 기능

- 로딩에 필요한 정보  
서버-클라 간의 교환 및  
데이터 설정.

# 팀 프로젝트 참여

Roka0105

분배된 역할에 맞게  
기능을 구현하며 팀원들과 협업.

서로 필요한 사항이 발생 시  
소통을 통해 조율 및 해결.

client update	게임 상태로 넘어가는 틀 완성 게임으로 씬 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 비추가	22 8 2022 20:55	roka0105 <cocapepsi123@daum.net>
client	버그 수정 다른 사람 채팅 출력 안되는 버그 수정 완료	22 8 2022 20:48	JangHan Kim <kimjh741963@hanmail.net>
UIScene	텍스트수정	22 8 2022 20:05	roka0105 <cocapepsi123@daum.net>
client	텍스트 프리랍 추가	22 8 2022 19:47	JangHan Kim <kimjh741963@hanmail.net>
Merge branch 'main'	of https://github.com/hamsoyeon/Yggdrasil	22 8 2022 19:29	roka0105 <cocapepsi123@daum.net>
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 차	22 8 2022 19:28	Jiny <jiny9789@gmail.com>
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 프	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	roka0105 <cocapepsi123@daum.net>
client update	일반 레디 명령 처리 렌더링 부분은 장한이 코드랑 합칠 예정	21 8 2022 17:26	JangHan Kim <kimjh741963@hanmail.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	roka0105 <cocapepsi123@daum.net>
[UIScene]	레디부분 수정 타 플레이어 캐릭선택시 그래픽 선택 비활성화	21 8 2022 5:21	hamsoyeon <chair3203@naver.com>
예람	할일 update	21 8 2022 2:37	JangHan Kim <kimjh741963@hanmail.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 플레이	21 8 2022 2:19	roka0105 <cocapepsi123@daum.net>
매인	씬 카메라 시네마틱 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>

**감사합니다!**