

# PORTFOLIO

---

## 클라이언트 프로그래머

한국산업기술대학교 게임공학과

김병석

# INDEX

---

자기 소개

수강 내역

주요 프로젝트 경험

1. Cartoon War
  2. The Legend of Zelda
  3. Mine Watch
  4. World of Tank
-

# 자기 소개

프로필



이름:	김병석
-----	-----

Phone Num:	010-6780-5510
------------	---------------

E-MAIL:	kbs5335586@gmail.com
---------	----------------------

Github	<a href="https://github.com/kbs5435586">https://github.com/kbs5435586</a>
--------	---

# 자기 소개

## 수강내역

---



### C/C++

절차치향, 객체지향 프로그래밍 학습



### 전산학개론



### 선형대수학



### 컴퓨터 구조



### 자료구조

단방향, 양방향 연결리스트 구현  
오목 AI 구현 (1인 프로젝트)



### 윈도우 프로그래밍 (Windows API)



### 컴퓨터 그래픽스 (OpenGL)



### 2D 게임 프로그래밍 (Python)



### 이산수학



### 알고리즘



### 운영체제



### 네트워크 기초

Network 이론 학습  
멀티 쓰레드 학습



### 스크립트 언어 (Python)

공공 API를 이용한 Python 프로젝트



### STL

Vector, List, Map 학습  
알고리즘 함수 및 람다 함수 학습



### 게임 수학

3D 게임 수학 이론 학습  
Bump Mapping, Skinning Animation  
Texture Sampling

# 자기 소개

## 수강내역

---



### 3D 게임 프로그래밍1 (DirectX12)

DirectX 이론  
DirectX12 초기화 과정 학습, HLSL 학습  
3D Simple RollerCoaster 구현  
Framework 제작



### 네트워크 게임 프로그래밍

World of Tank 리팩토링  
TCP 서버 구현 -> 3인용 2D 게임  
Multi Thread 구현, 클라이언트 동기화  
(2인 프로젝트)



### 데이터베이스

MySQL 실습



### 게임 소프트웨어 공학 (Git, OpenGL)

Git 이론 학습  
OpenGL 2D 간단한 RPG 게임 제작



### 게임 엔진 (Unity)

C# 학습  
게임 오브젝트 컴포넌트 구조 학습  
3D 유닛 디펜스 게임 제작 (1인 프로젝트)



### 셰이더 프로그래밍

GLSL 실습 및 Particle 제작



### 멀티 코어 프로그래밍

멀티 쓰레드, 쓰레드 동기화 학습

# 프로젝트 경험

---

## 01. Cartoon Wars 프로젝트 소개



장르 : 3D 대규모 전투

개발 시기 : 2021/01~2021/08

개발 목적 : 3D 그래픽스 이론을 비 실사 렌더링으로 접합하여 구현

개발 툴 : Visual Studio 2019, DirectX 12

개발 언어 : C++

팀원 : 3인

담당 역할 : 클라이언트

개발 중점 사항

- 프레임 워크 제작
- MFC를 이용한 맵틀 구현
- UI 제작
- Shader를 이용한 다양한 효과 구현



YouTube: [https://youtu.be/q\\_v3o8hCCKI](https://youtu.be/q_v3o8hCCKI) (client)

YouTube:

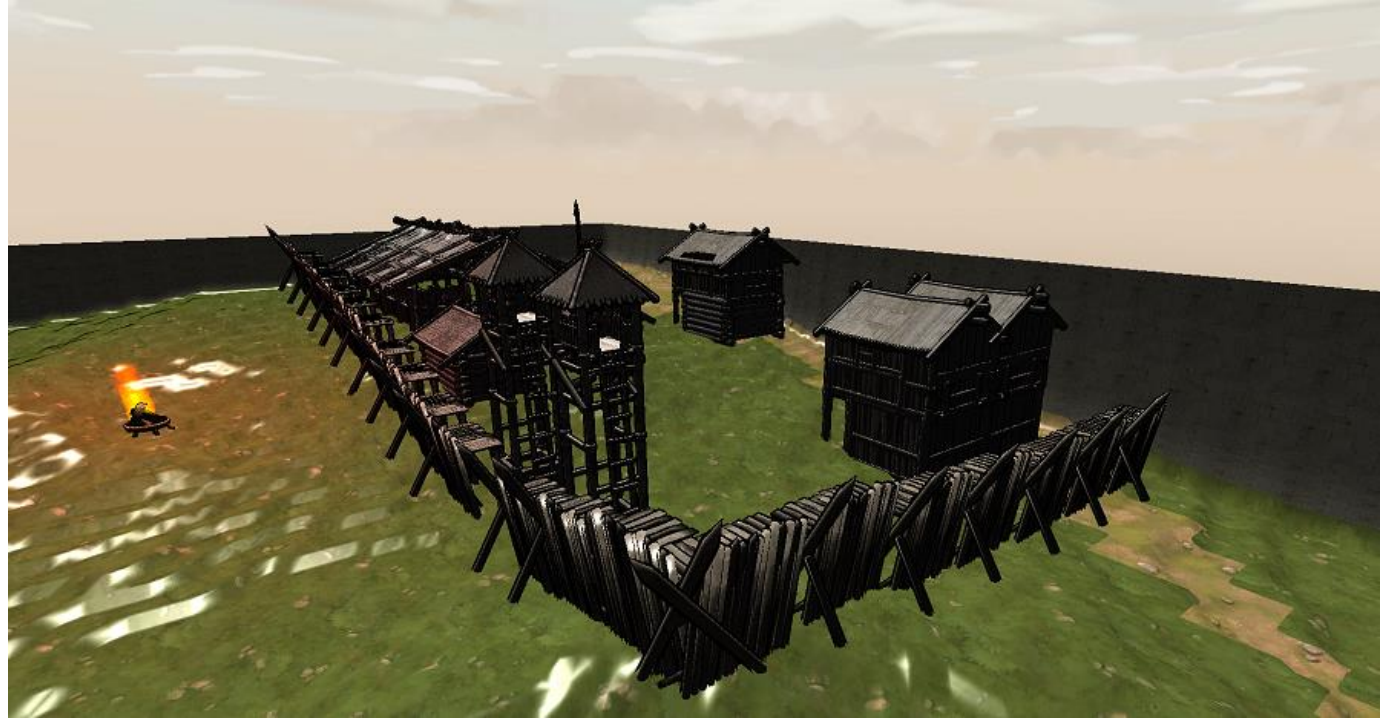
<https://www.youtube.com/watch?v=W28BjdZ42zE>  
(tool)

GitHub: <https://github.com/kbs5435586/Graduation>

# 프로젝트 경험

## 01. Cartoon Wars

### 프로젝트 소개



해당 게임은 각 플레이어끼리 경쟁을 통해서 골드를 얻고 골드를 바탕으로 각 플레이어 캐릭터의 진화, 아군 NPC 추가 및 진화를 하여 숨겨진 진지 또는 적군 진지를 탈환하는 게임입니다. 각각의 직업군마다 특성이 존재하여 항상 같은 전투가 아닌 지휘관의 능력에 따라 다른 전투가 진행되는 게임입니다.

# 프로젝트 경험

---

## 01. Cartoon Wars

### 프로젝트 소개

1. 게임 소개: <https://kbs5435586.tistory.com/7?category=1017616>
2. 디퍼드 렌더링: <https://kbs5435586.tistory.com/2?category=1017616>
3. 모션 블러: <https://kbs5435586.tistory.com/3?category=1017616>
4. Toon Shading: <https://kbs5435586.tistory.com/4?category=1017616>
5. Hatching Rendering: <https://kbs5435586.tistory.com/5?category=1017616>
6. 네비게이션 메쉬: <https://kbs5435586.tistory.com/6?category=1017616>
7. OBB 충돌: <https://kbs5435586.tistory.com/9>

링크 우클릭 후 새 탭에서 열기 부탁드립니다.



# 프로젝트 경험 서버 구조(1/5)

## 01. Cartoon Wars 세부 개발 내용

IOCP 서버를 이용하여 패킷을 관리하는 스레드를 4개를 만들어 줍니다.

```
vector<thread> worker_threads;
for (int i = 0; i < 4; ++i) // 컴퓨터가 쿼드코어라서 4를 넣었지만 본인 코어수만큼 넣어도 괜찮습니다.
{
    worker_threads.emplace_back([this]() {this->worker_thread(); });
}

while (rest_byte > 0) // 아직 남은 패킷이 있을때
{
    if (0 == packet_size) // 지금 우리가 처리하는 패킷이 처음이라면 포인터를 패킷 데이터 맨 앞을 가리키게 설정합니다.
        packet_size = *p;

    if (packet_size <= rest_byte + g_clients[user_id].m_prev_size) // 마지막으로 받은 데이터와 현재 데이터를 합쳐서 패킷 사이즈보다 크거나 같으면 패킷 완성됩니다.
    {
        memcpy(g_clients[user_id].m_packet_buf + g_clients[user_id].m_prev_size, p, packet_size - g_clients[user_id].m_prev_size); // p에 있는걸 packet_size만큼 m_packet_buf에 복사합니다.

        p += packet_size - g_clients[user_id].m_prev_size;
        rest_byte -= packet_size - g_clients[user_id].m_prev_size;
        packet_size = 0;
        process_packet(user_id, g_clients[user_id].m_packet_buf);
        g_clients[user_id].m_prev_size = 0;
    }
    else // 아직 패킷이 덜 왔을때
    {
        memcpy(g_clients[user_id].m_packet_buf + g_clients[user_id].m_prev_size, p, rest_byte); // p에 있는걸 rest_byte만큼 m_packet_buf에 복사합니다.
        // 혹시라도 2번이상 받았는데 패킷을 완성 못시킨 경우가 생길 수 있으니 이전에 받아놓은 크기 뒤부터 복사해오게 설정합니다.
        g_clients[user_id].m_prev_size += rest_byte;
        rest_byte = 0;
        p += rest_byte; // 처리한 데이터만큼 포인터 위치를 이동합니다.
    }
}
```

네트워크의 상태에 따라 전송되는 바이트가 달라지므로, 처리하게 되는 패킷의 사이즈를 누적해 처음 설정한 패킷의 사이즈와 같거나 크면 해당 패킷을 처리하게 됩니다. 위의 과정이 패킷을 재조합하는 과정입니다. 서버와 마찬가지로 클라이언트에서도 받은 패킷을 재조합하여 적용합니다.

# 프로젝트 경험 서버 구조 (2/5)

## 01. Cartoon Wars

### 세부 개발 내용

```
void Server::send_gold_packet(int user_id)
{
    sc_packet_gold packet; // 금 관련 패킷을 선언한다.
    packet.size = sizeof(packet);
    packet.type = SC_PACKET_GOLD;
    packet.gold = g_clients[user_id].m_gold;
    // 패킷에 정보를 담는다

    send_packet(user_id, &packet);
    // 해당 패킷과 클라이언트 번호를 패킷 전송 함수를 통해 보낸다.
}

void Server::send_packet(int user_id, void* packet)
{
    char* buf = reinterpret_cast<char*>(packet);

    OverEx* overEx = new OverEx; // 새로운 확장 오버랩 구조체를 할당한다.
    overEx->function = FUNC_SEND; // 오버랩 구조체 역할을 설정한다.
    ZeroMemory(&overEx->over, sizeof(overEx->over)); // 오버랩 구조체 메모리를 초기화한다.
    memcpy(overEx->io_buf, buf, buf[0]); // IOCP버퍼에 패킷 내용을 패킷 크기만큼 복사한다.
    overEx->wsabuf.buf = overEx->io_buf; // WSA버퍼에 IOCP버퍼를 복사한다.
    overEx->wsabuf.len = buf[0]; // 버퍼 사이즈를 설정한다.

    WSASend(g_clients[user_id].m_socket, &overEx->wsabuf, 1, NULL, 0, &overEx->over, NULL);
    // 해당 오버랩 구조체를 클라이언트에게 전송한다
}
```

이후, 해당 패킷에 영향을 받는 플레이어들에게 새로운 오버랩 구조체를 할당하여 결과값을 패킷에 담은 후 해당 클라이언트에 전달합니다.

# 프로젝트 경험 서버 구조 (3/5)

## 01. Cartoon Wars

### 세부 개발 내용

```
void Server::enter_game(int user_id, char name[])
{
    g_clients[user_id].m_cLock.lock(); // mutex lock 을 통한 데이터 레이스 방지
    strcpy_s(g_clients[user_id].m_name, name); // 플레이어의 이름
    g_clients[user_id].m_name[MAX_ID_LEN] = NULL;
    send_login_ok_packet(user_id); // 최초 플레이어 정보
    g_clients[user_id].m_status = ST_ACTIVE; // 플레이어의 로그인 상태
    g_clients[user_id].m_cLock.unlock();
}
```

```
void Server::do_revive(int id)
{
    g_clients[id].m_cLock.lock();
    g_clients[id].m_status = ST_ACTIVE;
    // 해당 데이터는 플레이어가 온라인인지 오프라인인지
    // 판단하는 데이터이므로 스레드 동기화가 되지 않아
    // 데이터 레이스가 일어날 경우 치명적이므로 mutex를 통해 동기화를 합니다.
    g_clients[id].m_cLock.unlock();
}
```

플레이어의 온라인, 오프라인을 판단하는 데이터

혹여나 이 과정에서 여러 스레드에서 동시에 건드릴 수 있는 데이터 값을 다룰 때에는 mutex lock, unlock을 이용하여 동기화해줍니다.

# 프로젝트 경험 서버 구조 (4/5)

## 01. Cartoon Wars 세부 개발 내용

```
#pragma pack(push,1)
struct sc_packet_move
{
    char size;
    char type;
    int id;
    float r_x, r_y, r_z;
    float u_x, u_y, u_z;
    float l_x, l_y, l_z;
    float p_x, p_y, p_z;
    int move_time;
};
#pragma pack (pop)
```

예) 패킷의 종류중 하나

```
#pragma pack(push,1)

struct sc_packet_login_ok
{
    char size;
    char type;
    int id;
    float p_x, p_y, p_z;
    float r_x, r_y, r_z;
    float u_x, u_y, u_z;
    float l_x, l_y, l_z;
    short hp;
    short level;
    int exp;
    short p_class;
};

#pragma pack (pop)
```

예) 패킷의 종류중 하나

프로토콜을 기능, 역할별로 최대한 세분화하여 패킷의 종류를 늘리는 대신 패킷 하나의 사이즈를 줄이고, 패킷 구조체를 #pragma pack(push,1)을 내부에 선언하여 구조체의 데이터를 1바이트 별로 정렬시켜 패딩 현상을 막았습니다. 또한 가장 많은 패킷 교환을 일으키는 플레이어 및 npc의 이동은 이동 시작, 멈춤, 회전 시작, 멈춤 시에만 교환하고 그 사이의 이동 연산은 클라이언트와 서버가 같은 계산식을 이용하여, 클라이언트는 자체 연산 결과를 렌더링에 반영하고 마지막 최종 결과값만 서버에서 받아 보정합니다.

# 프로젝트 경험 서버 구조 (5/5)

## 01. Cartoon Wars

### 세부 개발 내용

```
case GO_FORWARD:
    p.m_transform.Setup_Speed(p.m_move_speed, p.m_rotate_speed);
    p.m_transform.Go_Straight(TIME_DELTA);
    p.m_isAttack = false;
    break;
```

```
case GO_BACK:
    p.m_transform.Setup_Speed(p.m_move_speed, p.m_rotate_speed);
    p.m_transform.BackWard(TIME_DELTA);
    p.m_isAttack = false;
    break;
```

클라이언트에서 서버로 명령 신호를 보내면 해당 플레이어 속성 기반으로 이동연산을 진행합니다.

```
void Server::send_move_packet(int user_id, int other_id)
{
    sc_packet_move packet; // 위치 보정용 패킷을 선언합니다.
    packet.id = other_id;
    packet.size = sizeof(packet);
    packet.type = SC_PACKET_MOVE;

    _matrix pos = g_clients[other_id].m_transform.Get_Matrix();
    // 이동 연산이 다 끝나 최종 위치값이 저장된 객체의 매트릭스를 가져옵니다.
    packet.r_x = pos._11;
    packet.r_y = pos._12;
    packet.r_z = pos._13;
    packet.u_x = pos._21;
    packet.u_y = pos._22;
    packet.u_z = pos._23;
    packet.l_x = pos._31;
    packet.l_y = pos._32;
    packet.l_z = pos._33;
    packet.p_x = pos._41;
    packet.p_y = pos._42;
    packet.p_z = pos._43;
    // 해당 매트릭스 값을 패킷에 담아줍니다.

    send_packet(user_id, &packet);
}
```

계산이 완료된 위치값을 클라이언트로 보내줍니다.

# 프로젝트 경험 싱글톤 패턴 구현

## 01. Cartoon Wars 세부 개발 내용

CDevice.cpp

```
class CDevice :  
    public CBase  
{  
    _DECLARE_SINGLETON(CDevice)  
private:  
    CDevice();  
    virtual ~CDevice() = default;
```

```
#define NO_COPY(CLASSNAME)  
private:  
    CLASSNAME(const CLASSNAME&);  
    CLASSNAME& operator = (const CLASSNAME&);
```

```
#define _DECLARE_SINGLETON(CLASSNAME)  
NO_COPY(CLASSNAME)  
private:  
    static CLASSNAME* m_pInstance;  
public:  
    static CLASSNAME* GetInstance( void );  
    static unsigned long DestroyInstance( void );
```

```
#define _IMPLEMENT_SINGLETON(CLASSNAME)  
CLASSNAME* CLASSNAME::m_pInstance = NULL;  
CLASSNAME* CLASSNAME::GetInstance( void ) {  
    if(NULL == m_pInstance) {  
        m_pInstance = new CLASSNAME;  
    }  
    return m_pInstance;  
}  
unsigned long CLASSNAME::DestroyInstance( void ) {  
    unsigned long dwRefCnt = 0;  
    if(NULL != m_pInstance) {  
        dwRefCnt = m_pInstance->Release();  
        if(0 == dwRefCnt) m_pInstance = NULL;  
    }  
    return dwRefCnt;  
}
```

```
_IMPLEMENT_SINGLETON(CDevice)  
CDevice::CDevice()  
{  
}
```

- 싱글톤 패턴을 #defin을 사용한 매크로 작성했습니다.
- Device 클래스, Management 클래스 및 각 매니저들을 싱글톤 패턴으로 생성합니다.

# 프로젝트 경험

## Prototype 패턴 구현

### 01. Cartoon Wars

#### 세부 개발 내용

객체의 원형 생성 및 tag 부여합니다.

```
if (FAILED(pManagement->Add_Prototype_GameObject(L"GameObject_Player", CPlayer::Create())))  
    return E_FAIL;
```

```
HRESULT CGameObject_Manager::Add_GameObjectToLayer(const _tchar* pProtoTag,  
const _uint& iSceneID, const _tchar* pLayerTag, CGameObject** ppCloneObject, void* pArg, _uint iIdx)  
{  
    if (nullptr == m_pMapLayers)  
        return E_FAIL;  
  
    if (m_iNumScene <= iSceneID)  
        return E_FAIL;
```

```
    CGameObject* pPrototype = nullptr;
```

```
    pPrototype = Find_Prototype(pProtoTag);
```

```
    if (nullptr == pPrototype)  
        return E_FAIL;
```

원형 객체를 생성할 때 부여한 tag 검색

```
CGameObject* pGameObject = pPrototype->Clone_GameObject(pArg, iIdx);  
pGameObject->m_iLayerIdx = iIdx;  
if (nullptr == pGameObject)  
    return E_FAIL;
```

virtual CGameObject\*

Clone\_GameObject(void\* pArg, \_uint iIdx = 0) override;

해당 원형의 복사본 객체를 생성합니다.

```
    if (nullptr != ppCloneObject)  
        *ppCloneObject = pGameObject;
```

```
    CLayer* pLayer = Find_Layer(iSceneID, pLayerTag);
```

```
    if (nullptr == pLayer)  
    {
```

```
        pLayer = CLayer::Create();
```

```
        if (nullptr == pLayer)  
            return E_FAIL;
```

```
        if (FAILED(pLayer->Add_Object(pGameObject)))  
            return E_FAIL;
```

```
        m_pMapLayers[iSceneID].insert(MAPLAYERS::value_type(pLayerTag, pLayer));  
    }
```

```
    else
```

```
        if (FAILED(pLayer->Add_Object(pGameObject)))  
            return E_FAIL;
```

# 프로젝트 경험

## Component 패턴 구현(1/2)

### 01. Cartoon Wars

#### 세부 개발 내용

```
class CGameObject :
    public CBase
{
protected:
    map<const _tchar*, CComponent*> m_mapComponent;
}

class CPlayer :
    public CGameObject
{
private: 객체마다 필요한 Component들 추가
    CTransform*
    CRenderer*
    CMesh*
    CShader*
    CShader*
    CShader*
    CShader*
    CShader*
    CAnimator*
    CNavigation*
    CCollider*
    CTexture*
    CFrustum*
    m_pTransformCom = nullptr;
    m_pRendererCom = nullptr;
    m_pMeshCom[(_uint)CLASS::CLASS_END] = {nullptr};
    m_pShaderCom = nullptr;
    m_pComputeShaderCom = nullptr;
    m_pShaderCom_Shadow = nullptr;
    m_pShaderCom_PostEffect = nullptr;
    m_pShaderCom_Blur = nullptr;
    m_pAnimCom[(_uint)CLASS::CLASS_END] = {nullptr};
    m_pNaviCom = nullptr;
    m_pColiider[2] = { nullptr };
    m_pTextureCom[2] = {nullptr};
    m_pFrustumCom = nullptr;
```

추가된 Component들을 Map 컨테이너로 관리합니다.

- A is B인 커플링 관계가 아닌 A has B인 구조로 설계해 각 컴포넌트를 독립적으로 미리 구현합니다.
- 원하는 기능이 있을 시, 해당 기능을 하는 Component를 추가합니다.



# 프로젝트 경험

## Component 패턴 구현(2/2)

### 01. Cartoon Wars

#### 세부 개발 내용

Component들도 Prototype 패턴을 이용해 원본을 만든 후 객체 내부에서 사용시 복사본을 사용합니다.

```
if (FAILED(m_pManagement->Add_Prototype_Component((_uint)SCENEID::SCENE_STATIC, L"Component_Transform", CTransform::Create()))
    return E_FAIL;
    .
    .
    .

m_pTransformCom = (CTransform*)pManagement->Clone_Component((_uint)SCENEID::SCENE_STATIC, L"Component_Transform");
NULL_CHECK_VAL(m_pTransformCom, E_FAIL);
if (FAILED(Add_Component(L"Com_Transform", m_pTransformCom)))
    return E_FAIL;

HRESULT CGameObject::Add_Component(const _tchar* pComponentTag, CComponent* pComponent)
{
    if (nullptr == pComponent)
        return E_FAIL;

    if (nullptr != Find_Component(pComponentTag))
        return E_FAIL;

    m_mapComponent.insert(MAPCOMPONENT::value_type(pComponentTag, pComponent));

    pComponent->AddRef();

    return S_OK;
}
```

# 프로젝트 경험 Component(Renderer)

## 01. Cartoon Wars 세부 개발 내용

렌더링 하는 순서가 중요하므로 렌더링 순서 구분합니다.

```
enum RENDERGROUP { RENDER_PRIORITY, RENDER_NONEALPHA, RENDER_LIGHT, RENDER_ALPHA, RENDER_UI,  
RENDER_BLEND, RENDER_SHADOW, RENDER_POST, RENDER_BLUR, RENDER_END };
```

This 포인터를 이용해 자신을 RENDER\_NONALPHA 그룹에 추가합니다.

```
if (FAILED(m_pRendererCom->Add_RenderGroup(CRenderer: {RENDER_NONEALPHA, this})))  
    return -1;
```

```
HRESULT CRenderer::Add_RenderGroup(RENDERGROUP eGroup, CGameObject* pGameObject)
```

```
{  
    if (RENDER_END <= eGroup)  
        return E_FAIL;  
  
    if (nullptr == pGameObject)  
        return E_FAIL;  
  
    m_RenderList[eGroup].push_back(pGameObject);  
  
    pGameObject->AddRef();  
  
    return S_OK;  
}
```

설정한 렌더링 순서대로 렌더링 진행합니다.

```
_uint iSwapChainIdx = CDevice::GetInstance()->GetSwapChainIdx();  
pManagement->Get_RTT((_uint)MRT::MRT_SWAPCHAIN)->Clear(iSwapChainIdx);  
pManagement->Get_RTT((_uint)MRT::MRT_DEFERD)->Clear();  
pManagement->Get_RTT((_uint)MRT::MRT_LIGHT)->Clear();  
pManagement->Get_RTT((_uint)MRT::MRT_SHADOW)->Clear();  
pManagement->Get_RTT((_uint)MRT::MRT_BLUR)->Clear();  
pManagement->Get_RTT((_uint)MRT::MRT_INVEN)->Clear();
```

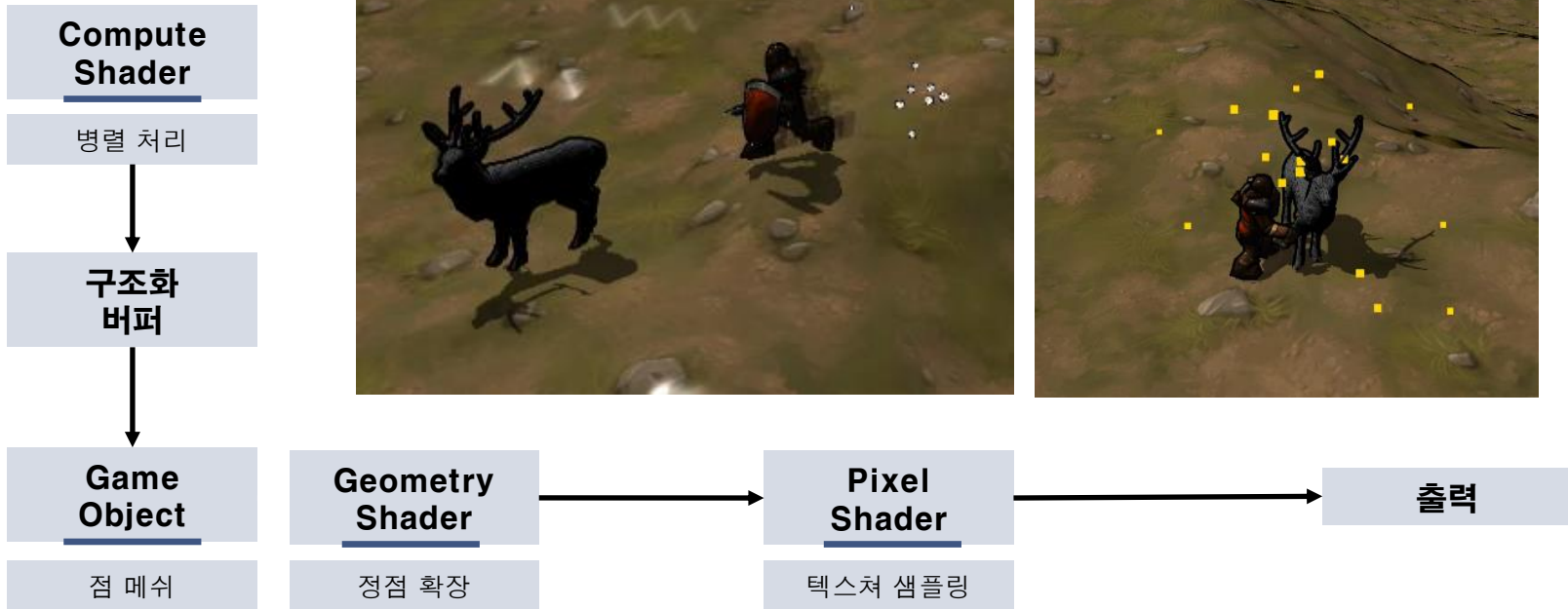
```
Render_Shadow(pManagement);  
Render_Blur();  
Render_Deffered(pManagement);  
Render_Light(pManagement);
```

```
iSwapChainIdx = CDevice::GetInstance()->GetSwapChainIdx();  
pManagement->Get_RTT((_uint)MRT::MRT_SWAPCHAIN)->OM_Set(1, iSwapChainIdx);
```

```
Render_Blend();  
Render_Priority();  
Render_OutLine();  
Render_Alpha();  
Render_Post_Effect();  
pManagement->Render_Font();  
Render_UI();
```

# 프로젝트 경험 파티클 시스템

## 01. Cartoon Wars 세부 개발 내용



컴퓨터 셰이더를 사용하여 파티클 각각의 시간, 운동 등의 정보를 계산하고 이를 구조화 버퍼에 저장하고 저장한 버퍼는 파티클 오브젝트의 셰이더에서 정보를 읽어와 위치와 크기를 계산합니다.

파티클 오브젝트는 점 하나의 점 메쉬를 사용하여 기하 셰이더를 통해 이를 사각형 평면으로 확장 후 픽셀 셰이더에서 파티클 텍스처를 샘플링하여 출력합니다.

### 컴퓨터 셰이더 코드

```
[numthreads(1024, 1, 1)]
void CS_Main(int3 _iThreadIdx : SV_DispatchThreadID)
{
    if (_iThreadIdx.x >= g_int_0)
        return;
    tRWSharedData[0].iAddCount = g_int_1;
    if (0 == tRWData[_iThreadIdx.x].iAlive)
    {
        int iOrigin = tRWSharedData[0].iAddCount;
        int iExchange = 0;
        while (0 < iOrigin)
        {
            int iInputValue = iOrigin - 1;
            InterlockedExchange(tRWSharedData[0].iAddCount, iInputValue, iExchange);
            if (iExchange == iOrigin)
            {
                tRWData[_iThreadIdx.x].iAlive = 1;
                break;
            }
            iOrigin = iInputValue;
        }
        if (1 == tRWData[_iThreadIdx.x].iAlive)
        {
            float2 vUV = float2(((float)_iThreadIdx.x / (float)g_int_0) + g_fAccTime, g_fAccTime);
            vUV.y += sin(vUV.x * 2 * 3.141592);
            if (vUV.x > 0)
                vUV.x = frac(vUV.x);
            else
                vUV.x = ceil(abs(vUV.x)) - abs(vUV.x);
            if (vUV.y > 0)
                vUV.y = frac(vUV.y);
            else
                vUV.y = ceil(abs(vUV.y)) - abs(vUV.y);
            vUV = vUV * g_vec2_0;
            float3 vNoise =
            {
                gaussian5x5Sample(vUV + int2(0, -100), g_texture0),
                gaussian5x5Sample(vUV + int2(0, 0), g_texture0),
                gaussian5x5Sample(vUV + int2(0, 100), g_texture0)
            };
            float3 vDir = (float4) 0.f;
            vDir = (vNoise - 0.5f) * 2.f;
            tRWData[_iThreadIdx.x].vWorldDir = normalize(vDir);
            tRWData[_iThreadIdx.x].vWorldPos = float3(0.f, 0.f, 0.f);
            tRWData[_iThreadIdx.x].m_fLifeTime = ((g_float_1 - g_float_0) * vNoise.x) + g_float_0;
            tRWData[_iThreadIdx.x].m_fCurTime = 0.f;
        }
    }

    else // Particle Update 하기
    {
        tRWData[_iThreadIdx.x].m_fCurTime += g_fDT;
        if (tRWData[_iThreadIdx.x].m_fLifeTime < tRWData[_iThreadIdx.x].m_fCurTime)
        {
            tRWData[_iThreadIdx.x].iAlive = 0;
            return;
        }

        float fRatio = tRWData[_iThreadIdx.x].m_fCurTime / tRWData[_iThreadIdx.x].m_fLifeTime;
        float fSpeed = (g_float_3 - g_float_2) * fRatio + g_float_2;
        float3 vNormalizeDir = normalize(tRWData[_iThreadIdx.x].vWorldDir);
        tRWData[_iThreadIdx.x].vWorldPos += vNormalizeDir * fSpeed * g_fDT;
        tRWSharedData[0].iCurCount = 0;
        InterlockedAdd(tRWSharedData[0].iCurCount, 1);
    }
}
```

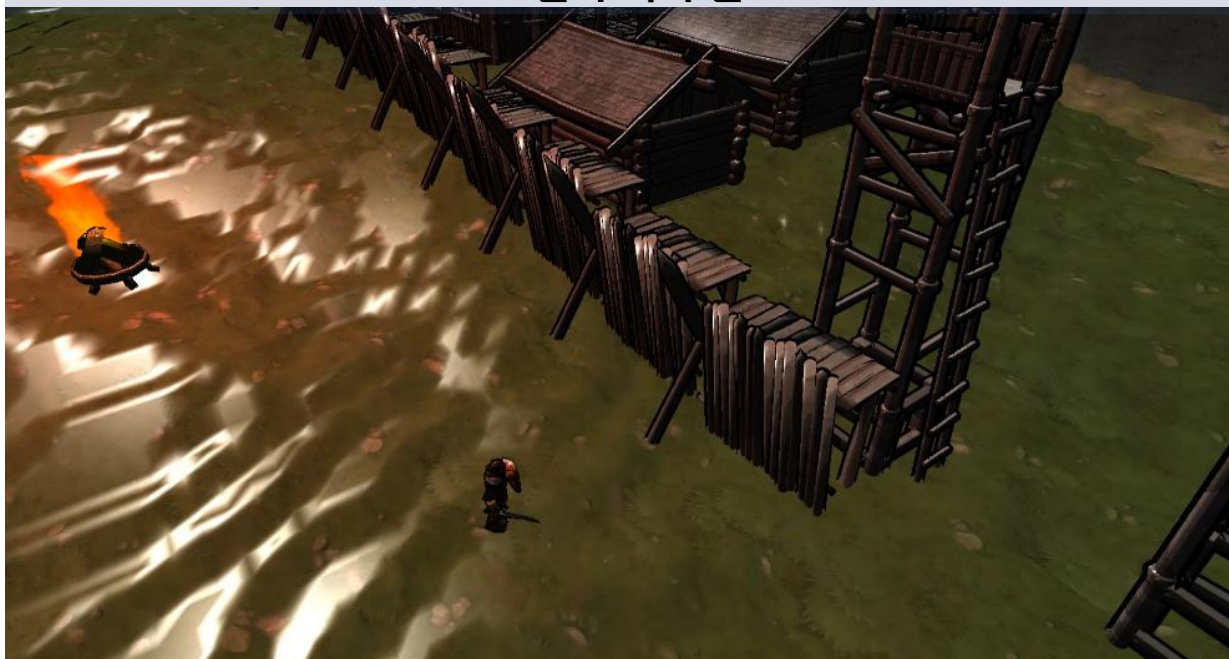
# 프로젝트 경험

## 실시간 그림자 처리(1/2)

### 01. Cartoon Wars

#### 세부 개발 내용

그림자 처리 전



그림자가 존재하지 않아 입체감이 부족한 모습

그림자 처리 후



광원에 카메라를 추가하여 해당 카메라는 Shadow Map에 오브젝트의 깊이값을 저장  
디퍼드 렌더링의 Merge Light 과정에서 해당 Shadow Map을 리소스로 가져와 그림자 출력

광원의 카메라는 플레이어의 위치를 반영하여 플레이어 기준으로 정해진 반경의 그림자를  
그려냅니다.



# 프로젝트 경험 실시간 그림자 처리(2/2)

## 01. Cartoon Wars 세부 개발 내용

```
HRESULT CScene_Stage::Ready_Layer_Light_Camera(const _tchar* pLayerTag, CManagement* pManagement)
{
    CLight_Camera* pCameraObject = nullptr;
    if (FAILED(pManagement->Add_GameObjectToLayer(L"GameObject_Camera_Light", (_uint)SCENEID::SCENE_STAGE, pLayerTag,
        (CGameObject**) &pCameraObject)))
        return E_FAIL;
    CAMERADesc tCameraDesc;
    ZeroMemory(&tCameraDesc, sizeof(CAMERADesc));
    tCameraDesc.vEye = _vec3(-1000.f, 1000.f, 0.f);
    tCameraDesc.vAt = _vec3(1.f, -1.f, 0.f);
    tCameraDesc.vAxisY = _vec3(0.f, 1.f, 0.f);

    PROJDesc tProjDesc;
    ZeroMemory(&tProjDesc, sizeof(tProjDesc));
    tProjDesc.fFovY = XMConvertToRadians(60.f);
    tProjDesc.fAspect = _float(WINCX) / WINCY;
    tProjDesc.fNear = g_Near;
    tProjDesc.fFar = 300.f;

    if (FAILED(pCameraObject->SetUp_CameraProjDesc(tCameraDesc, tProjDesc, true)))
        return E_FAIL;

    return S_OK;
}
```

### 광원 시점에 카메라 추가

```
if (dot(tCurCol.vDiffuse, tCurCol.vDiffuse) != 0.f)
{
    float4 vWorldPos = mul(vPosition, matViewInv);
    float4 vShadowProj = mul(vWorldPos, g_mat_0);
    float fDepth = vShadowProj.z / vShadowProj.w;
    float2 vShadowUV = float2((vShadowProj.x / vShadowProj.w) * 0.5f + 0.5f,
        (vShadowProj.y / vShadowProj.w) * -0.5f + 0.5f);

    if (0.01f < vShadowUV.x && vShadowUV.x < 0.99f && 0.01f < vShadowUV.y && vShadowUV.y < 0.99f)
    {
        float fShadowDepth = g_texture2.Sample(Sampler0, vShadowUV).r;

        // 그림자인 경우 빛을 약화시킨다.
        if (fShadowDepth && (fDepth > fShadowDepth + 0.00001f))
        {
            tCurCol.vDiffuse *= 0.01f;
            tCurCol.vSpecular = 0.f;
        }
    }
}
```

### Light Merge

```
void CPlayer::Render_GameObject_Shadow()
{
    CManagement* pManagement = CManagement::GetInstance();
    if (nullptr == pManagement)
        return;
    pManagement->AddRef();

    _uint iSubsetNum = m_pCurMeshCom->GetSubsetNum();
    for (_uint i = 0; i < iSubsetNum; ++i)
    {
        MAINPASS tMainPass = {};
        _matrix matWorld = m_pTransformCom->Get_Matrix();
        _matrix matView = CCamera_Manager::GetInstance()->GetShadowView();
        _matrix matProj = CCamera_Manager::GetInstance()->GetShadowMatProj();

        REP tRep = {};
        tRep.m_arrInt[0] = 1;
        tRep.m_arrInt[1] = m_pCurAnimCom->GetBones()->size();

        m_pShaderCom_Shadow->SetUp_OnShader(matWorld, matView, matProj, tMainPass);

        _uint iOffset = pManagement->GetConstantBuffer((_uint)CONST_REGISTER::b0)->SetData((void*)&tMainPass);
        CDevice::GetInstance()->SetConstantBufferToShader(pManagement->GetConstantBuffer(
            (_uint)CONST_REGISTER::b0)->GetCBV().Get(), iOffset, CONST_REGISTER::b0);

        iOffset = pManagement->GetConstantBuffer((_uint)CONST_REGISTER::b8)->SetData((void*)&tRep);
        CDevice::GetInstance()->SetConstantBufferToShader(pManagement->GetConstantBuffer(
            (_uint)CONST_REGISTER::b8)->GetCBV().Get(), iOffset, CONST_REGISTER::b8);

        m_pCurAnimCom->UpdateData(m_pCurMeshCom, m_pComputeShaderCom);
        CDevice::GetInstance()->UpdateTable();
        m_pCurMeshCom->Render_Mesh(i);
    }
    Safe_Release(pManagement);
}
```

### ShadowMapTexture 에 깊이 기록

```
ComPtr<ID3D12DescriptorHeap> pShadowTex = CManagement::GetInstance()->Get_RTT((_uint)MRT::MRT_SHADOW)->Get_RTT(0)->pRtt->GetSRV().Get();
CDevice::GetInstance()->SetTextureToShader(pShadowTex.Get(), TEXTURE_REGISTER::t2);
```

### Light Texture에 ShadowMapTexture을 Merge

```
VS_OUT VS_Main(VS_IN vIn)
{
    VS_OUT output = (VS_OUT)0.f;

    if (g_int_0)
    {
        Skinning(vIn.vPosition, vIn.vWeight, vIn.vIndices, 0);
    }

    output.vPosition = mul(float4(vIn.vPosition, 1.f), matWVP);
    output.vProj = output.vPosition;

    return output;
}

float4 PS_Main(VS_OUT vIn) : SV_Target
{
    return float4(vIn.vProj.z / vIn.vProj.w, 0.f, 0.f, 0.f);
}
```

### 그림자 셰이더

# 프로젝트 경험

---

## 02. Mine Watch 프로젝트 소개



장르 : FPS

개발 시기 : 2018/11~2019/01

개발 목적 : 3D 이론과 서버를 연동한 멀티 플레이 게임 개발

개발 툴 : Visual Studio 2015, DirectX 9

개발 언어 : C++

팀원 : 4인

담당 역할 : 클라이언트(UI) / Map tool 제작

개발 중점 사항

- 프레임 워크 제작
- MFC를 이용한 Map tool 구현
- UI 제작
- 충돌처리 및 게임 콘텐츠 개발



---

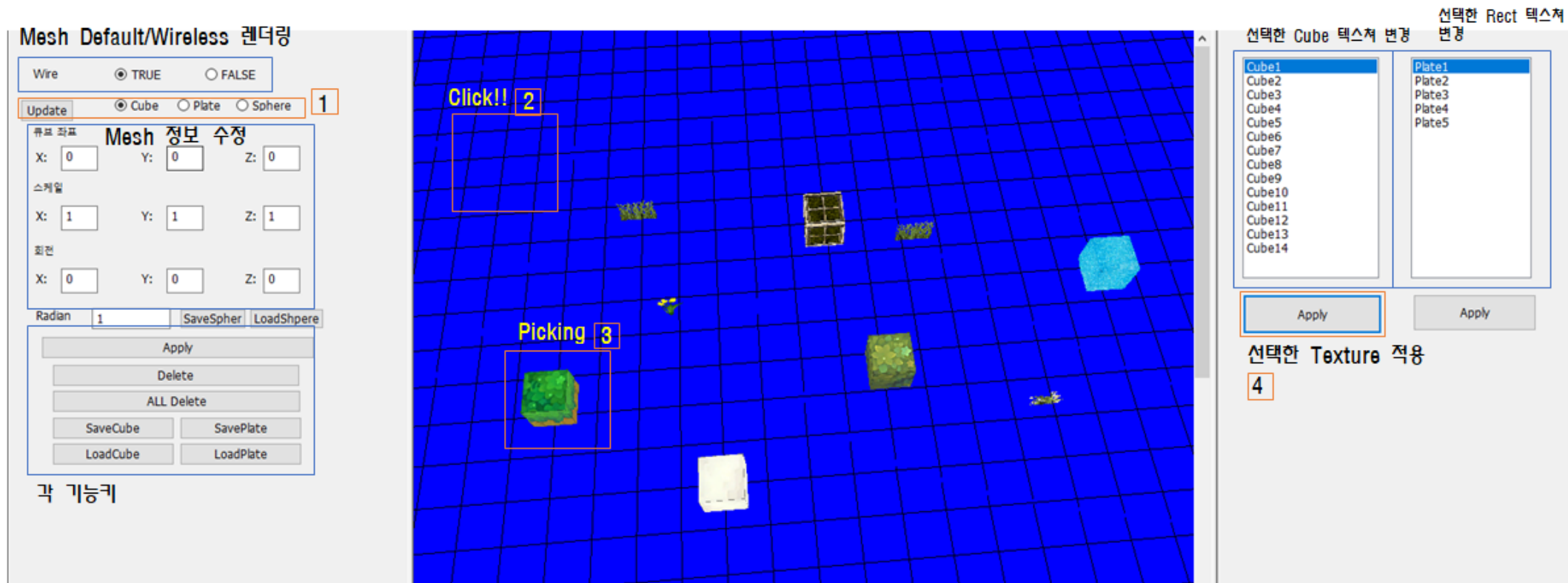
**YouTube:** <https://www.youtube.com/watch?v=h8RRAiZFsxk>

# 프로젝트 경험

## Map Tool(1/3)

### 02. Mine Watch

#### 세부 개발 내용



1. 배치하려는 Mesh 선택(Cube, Plane, Sphere)
2. Grid 위 클릭
3. 설치된 Mesh Picking
4. 텍스처 적용

# 프로젝트 경험

## Map Tool(2/3)

### 02. Mine Watch

#### 세부 개발 내용

3D공간을 표시하기 위해 행렬들을 이용해서 화면으로 출력한 것처럼 화면에서 클릭했을 때 어떤 것을 클릭했는지 확인하기 위해서는 화면의 x, y 좌표를 변환해야 합니다.

```
public:
```

```
    D3DXVECTOR3    m_vOrg;
    D3DXVECTOR3    m_vDir;
```

m\_vOrg : 광선의 출발 지점(마우스로 클릭한 좌표)  
m\_vDir: 광선의 방향(화면 안쪽으로 향하게 됨)

RayAtViewSpace함수: 현재 투영변환까지 된 화면을 ViewSpace로 변환한다.

1. ViewPort를 얻어온 뒤 저장합니다.

2. ProjectionMatrix를 얻어온 뒤 저장합니다.

3-1)  $X = ((+2.0f * \text{마우스X좌표}) / \text{뷰포트 넓이} - 1.0f) / \text{투영행렬중 1행 1열}$

(만약 배열, 벡터로 행렬이 되어 있다면 MatProjection[0][0]에 해당한다)  
->이는 투영행렬의 1행 1열이 X값이 변하는데 영향을 주기 때문입니다.

3-2)  $Y = ((-2.0f * \text{마우스Y좌표}) / \text{뷰포트 높이} + 1.0f) / \text{투영행렬중 2행 2열}$

(배열, 벡터라면 [1][1]에 해당) ->투영행렬의 2행 2열이 Y값 변화에 영향을 줍니다.

3-3)  $Z = 1.0f$  투영 윈도우 값이다. DirectX는 평면  $z = 1$  과 일치하도록 정의하고 있습니다.

```
CRay CRay::RayAtViewSpace(int x, int y)
{
    D3DVIEWPORT9 vp;
    CDevice::GetInstance()->GetDevice()->GetViewport(&vp);

    D3DXMATRIXA16 matProj;
    CDevice::GetInstance()->GetDevice()->GetTransform(D3DTS_PROJECTION, &matProj);

    CRay ray;
    ray.m_vDir.x = (x * 2.0f / vp.Width - 1.0f) / matProj._11;
    ray.m_vDir.y = (-y * 2.0f / vp.Height + 1.0f) / matProj._22;
    ray.m_vDir.z = 1.0f;

    ray.m_eRaySpace = E_VIEW;

    return ray;
}
```



# 프로젝트 경험

## Map Tool(3/3)

### 02. Mine Watch

#### 세부 개발 내용

```
CRay CRay::RayAtWorldSpace(int x, int y)
{
    CRay ray = CRay::RayAtViewSpace(x, y);

    D3DXMATRIXA16 matView, matInvView;
    CDevice::GetInstance()->GetDevice()->GetTransform(D3DTS_VIEW, &matView);
    D3DXMatrixInverse(&matInvView, NULL, &matView);

    D3DXVec3TransformCoord(&ray.m_vOrg, &ray.m_vOrg, &matInvView);
    D3DXVec3TransformNormal(&ray.m_vDir, &ray.m_vDir, &matInvView);

    D3DXVec3Normalize(&ray.m_vDir, &ray.m_vDir);
    ray.m_eRaySpace = E_WORLD;

    return ray;
}
```

**RayAtWorldSpace함수:** 위에서 변환된 ViewSpace상의 좌표를 WorldSpace상 공간으로 변환시킵니다.

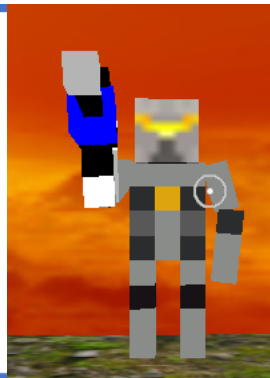
- 1) ray에 뷰포트, 투영행렬을 이용해 변환한 방향벡터를 얻어옵니다.
- 2) ViewMatrix 와 그 역행렬을 구한 뒤 저장합니다.
- 3) ViewMatrix 의 역행렬을 이용해서 광선의 출발점을 변환합니다.
- 4) 뷰의 역행렬을 이용해서 방향벡터를 변환합니다.
- 5) 방향벡터를 정규화(행렬안의 모든 값이 1보다 작도록) 합니다.

# 프로젝트 경험 캐릭터

## 02. Mine Watch 세부 개발 내용



- 메이
- 보통 공격 속도
- 보통 공격력
- Skill1: 얼음벽 세우기
- Skill2: 플레이어 본인 얼음으로 변함, HP회복, 일정 시간 무적
- 궁극기: 하늘에서 눈을 흘날려 범위 내 있는 적에게 데미지 가함



- 라인하르트 + 위도우메이커
- 보통 공격 속도
- 보통 공격력
- 점진, 줌아웃 가능
- Skill1: 일정 시간 공중에 날 수 있음
- Skill2: 플래시뱃,
- 궁극기: 투시



- 정크랫
- 느린 공격 속도
- 강한 공격력
- Skill1: 높이 뛰기
- Skill2: 연사 가능한 총 사용(4발)
- 궁극기: 느린 발사 속도를 가진 폭탄을 발사, 다른 플레이어와 충돌 시 즉사.



- 겐지
- 빠른 공격 속도
- 약한 공격력
- Skill1: 돌진, 돌진 중 다른 플레이어와 충돌시 피해 가함
- Skill2: 순간이동,
- 궁극기: Skill1, Skill2의 쿨타임을 비약적으로 감소 시킴

# 프로젝트 경험 UI

## 02. Mine Watch 세부 개발 내용



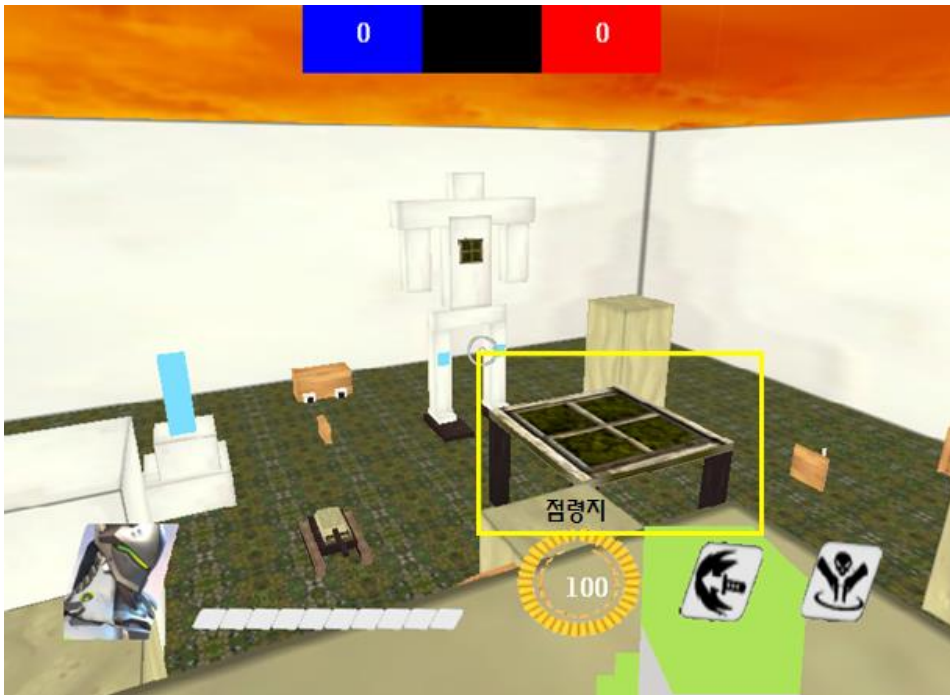
- 궁극기 및 스킬 사용시 쿨타임이 표시 되도록 설계했습니다.

캐릭터 초상화: 특정 장소에서 'H'키를 누를 시 캐릭터 선택창 나타나며 캐릭터 선택 시 초상화 변경  
HP: 피해를 입으면 수치만큼 칸이 줄어들도록 설계  
궁극기 게이지: 시간값을 사용하여 궁극기 게이지 시간 계산  
Skill: 캐릭터 선택 시 해당 캐릭터의 스킬로 변경

# 프로젝트 경험 모드

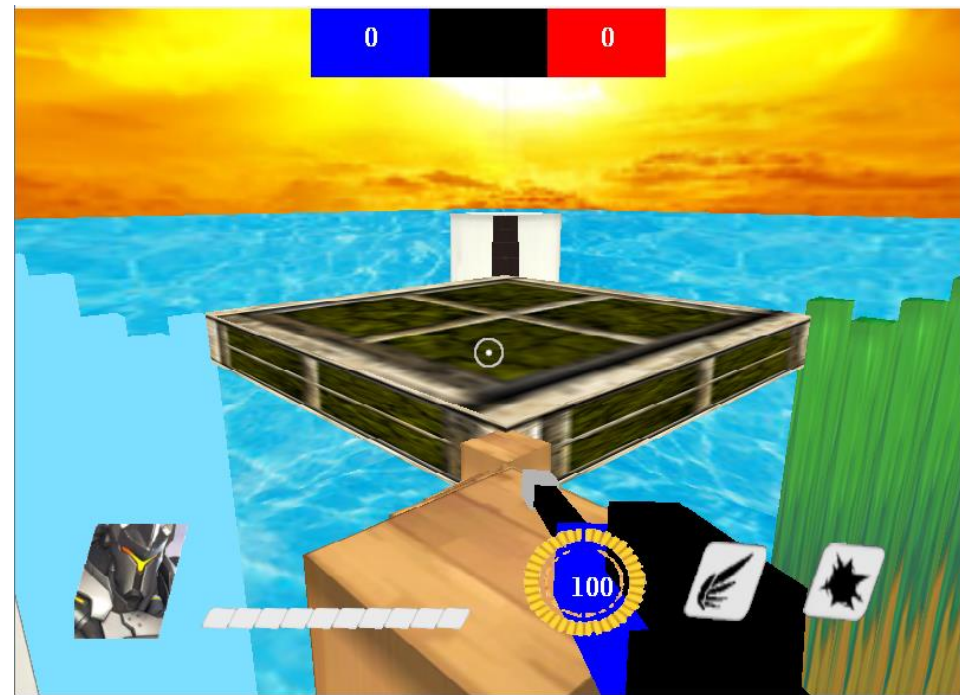
## 02. Mine Watch

### 세부 개발 내용



#### 점령전

- 2 : 2로 싸워 점령지를 오래 점령하는 팀이 승리하는 모드
- 총 3라운드로 되어있으며 2번 먼저 승리하는 팀이 최종 승리하게 됩니다.



#### 개인전

- 사각형 각 꼭지점에는 다른 플레이어들이 리스폰 됩니다.
- 바닥으로 떨어지면 바로 사망하게 됩니다.
- 끝까지 버티는 플레이어가 우승하게 됩니다.

# 프로젝트 경험

---

## 03. The Legend Of Zelda 프로젝트 소개



장르 : 2D RPG

개발 시기 : 2018/10~2018/11

개발 툴 : Visual Studio 2015, DirectX 9

개발 언어 : C++

팀원 : 1인

담당 역할 : 클라이언트

개발 중점 사항

- 프레임 워크 제작
- MFC를 이용한 Map tool 구현
- UI 제작
- 충돌처리 및 게임 콘텐츠 개발



---

YouTube: [https://www.youtube.com/watch?v=6\\_e72E0zrrw](https://www.youtube.com/watch?v=6_e72E0zrrw)

---



# 프로젝트 경험 추상화 패턴(1/2)

## 03. The Legend Of Zelda 세부 개발 내용

```
CObj* pObj = CAbstractFactory<CMyLogo>::CreateObj();

class CObj;
template <typename T>
class CAbstractFactory
{
public:
    static CObj* CreateObj()
    {
        CObj* pObjject = new T;

        if (FAILED(pObjject->Init()))
        {
            SafeDelete(pObjject);
            return nullptr;
        }

        return pObjject;
    }

    static CObj* CreateObj(float f, float f1) { ... }
    static CObj* CreateObj(D3DXVECTOR3 Pos) { ... }
};
```

Factory Class 선언

CObj에 Init함수를 호출

```
class CObj
{
public:
    CObj();
    virtual ~CObj();
public:
    virtual HRESULT Init() PURE;
};
```

### 추상 팩토리 패턴

- 공통된 부모로부터 파생되는 객체를 생성할 때마다 작성되고 공통된 로직을 추상화하는 디자인 패턴입니다.
- 특징
  - Template을 사용합니다.
  - Factory 클래스를 이용하여 객체를 생성합니다.
  - 인자에 따라 생성되는 팩토리의 종류가 달라집니다.

### 장점

- 객체의 자료형이 하위 클래스에 의해서 결정됨으로 확장에 용이합니다.
- 동일한 형태로 프로그래밍이 가능합니다.
- 확장성이 있는 전체 프로젝트 구성이 가능합니다.

### 단점

- 객체가 늘어날 때마다 하위 클래스 재정의로 인한 불필요한 많은 클래스 생성 가능성이 있습니다.

# 프로젝트 경험 추상화 패턴(2/2)

## 03. The Legend Of Zelda 세부 개발 내용




```
CCollisionMgr::GetInstance()->CollisionRects(m_ObjLst[EQUIP], m_ObjLst[MONSTER]);  
CCollisionMgr::GetInstance()->CollisionRects(m_ObjLst[EQUIP], m_ObjLst[MID_BOSS]);  
CCollisionMgr::GetInstance()->CollisionRects(m_ObjLst[EQUIP], m_ObjLst[BOSS]);  
CCollisionMgr::GetInstance()->CollisionRects(m_ObjLst[EQUIP], m_ObjLst[FINALBOSS]);  
  
CCollisionMgr::GetInstance()->CollisionRects(m_ObjLst[BULLET], m_ObjLst[MONSTER]);  
CCollisionMgr::GetInstance()->CollisionRects(m_ObjLst[BULLET], m_ObjLst[MID_BOSS]);  
CCollisionMgr::GetInstance()->CollisionRects(m_ObjLst[BULLET], m_ObjLst[BOSS]);  
CCollisionMgr::GetInstance()->CollisionRects(m_ObjLst[BULLET], m_ObjLst[FINALBOSS]);
```

- AABB 충돌 처리를 이용했습니다.
- 선택한 Object 그룹끼리 충돌 할 수 있으며 그 종류에 따라 다른 효과를 줄 수 있습니다.

```
void CCollisionMgr::CollisionRects(OBJLIST & dstLst, OBJLIST & srcLst)  
{  
    float fMoveX = 0.f, fMoveY = 0.f;  
  
    for (CObj*& pDst : dstLst)  
    {  
        for (CObj*& pSrc : srcLst)  
        {  
            if (CheckRects(pDst, pSrc, &fMoveX, &fMoveY, 1))  
            {  
                pDst->SetBulletLife(1);  
                pSrc->SetHp(pDst->GetInfo());  
                if (fMoveX > fMoveY)  
                {  
                    if (pDst->GetInfo().vPos.y >= pSrc->GetInfo().vPos.y)  
                        fMoveY *= -1.f;  
  
                    pSrc->SetPos(pSrc->GetInfo().vPos.x, pSrc->GetInfo().vPos.y + fMoveY*0.1f);  
                }  
                else  
                {  
                    if (pDst->GetInfo().vPos.x >= pSrc->GetInfo().vPos.x)  
                        fMoveX *= -1.f;  
  
                    pSrc->SetPos(pSrc->GetInfo().vPos.x + fMoveX*0.1f, pSrc->GetInfo().vPos.y);  
                }  
            }  
        }  
    }  
}  
  
bool CCollisionMgr::CheckRects(CObj *& pDst, CObj *& pSrc, float * pMoveX, float * pMoveY, int num)  
{  
    float fSumRadX = (pDst->GetCollSize().x + pSrc->GetCollSize().x)* 0.5f;  
    float fSumRadY = (pDst->GetCollSize().y + pSrc->GetCollSize().y)* 0.5f;  
  
    float fDistX = fabs(pDst->GetInfo().vPos.x - pSrc->GetInfo().vPos.x);  
    float fDistY = fabs(pDst->GetInfo().vPos.y - pSrc->GetInfo().vPos.y);  
  
    if (fSumRadX >= fDistX && fSumRadY >= fDistY)  
    {  
        *pMoveX = fSumRadX - fDistX;  
        *pMoveY = fSumRadY - fDistY;  
  
        return true;  
    }  
  
    return false;  
}
```

# 프로젝트 경험 몬스터

## 03. The Legend Of Zelda 세부 개발 내용

		
Snake	Octo	Moblin
일정 거리 내에 플레이어가 있으면 플레이어를 향해 전진	일정 거리 내에 플레이어가 있으면 플레이어를 향해 전진 범위 밖에 플레이어가 있을 경우 플레이어를 향해 탄환 발사	일정 거리 내에 플레이어가 있으면 플레이어를 향해 돌진

```

if (m_pTarget->GetInfo().vPos.x+vScroll.x >= m_tInfo.vPos.x - 150 &&
    m_pTarget->GetInfo().vPos.x +vScroll.x<= m_tInfo.vPos.x + 150
    && m_pTarget->GetInfo().vPos.y+vScroll.y >= m_tInfo.vPos.y - 150 &&
    m_pTarget->GetInfo().vPos.y+vScroll.y <= m_tInfo.vPos.y + 150)
{
    m_bMonsterPattern = true;
    D3DXMatrixScaling(&matScale, m_tInfo.vSize.x, m_tInfo.vSize.y, 0.f);
    D3DXMatrixTranslation(&matTrans, m_tInfo.vPos.x - vScroll.x, m_tInfo.vPos.y - vScroll.y, 0.f);

    m_tInfo.matWorld = matScale * matTrans;

    m_tInfo.vDir = m_pTarget->GetInfo().vPos+vScroll - m_tInfo.vPos;

    D3DXVec3Normalize(&m_tInfo.vDir, &m_tInfo.vDir);

    m_tInfo.vPos += m_tInfo.vDir * m_fSpeed; Player을 향해 전진
}

```

Snake Pattern

```

if (m_pTarget->GetInfo().vPos.x + vScroll.x >= m_tInfo.vPos.x - 100 &&
    m_pTarget->GetInfo().vPos.x + vScroll.x <= m_tInfo.vPos.x + 100
    && m_pTarget->GetInfo().vPos.y + vScroll.y >= m_tInfo.vPos.y - 100 &&
    m_pTarget->GetInfo().vPos.y + vScroll.y <= m_tInfo.vPos.y + 100)
{
    D3DXMatrixScaling(&matScale, m_tInfo.vSize.x, m_tInfo.vSize.y, 0.f);
    D3DXMatrixTranslation(&matTrans, m_tInfo.vPos.x-vScroll.x, m_tInfo.vPos.y-vScroll.y, 0.f);

    m_tInfo.matWorld = matScale * matTrans;

    m_tInfo.vDir = m_pTarget->GetInfo().vPos+vScroll - m_tInfo.vPos;

    D3DXVec3Normalize(&m_tInfo.vDir, &m_tInfo.vDir);

    m_tInfo.vPos += m_tInfo.vDir * m_fSpeed; Player을 향해 전진
}
else
{
    m_bCheckDist = true;
}

```

범위 밖에 있을 경우  
 if (m\_bCheckDist)  
     FireBullet();  
     투사체를 발사함



# 프로젝트 경험

## 아이템

### 03. The Legend Of Zelda

세부 개발 내용



필드에서 떨어지는 장비들을 얻어 인벤토리에 저장됩니다.

Item을 특정 키에 장착 할 수 있습니다.



# 프로젝트 경험

---

## 04. World of Tanks 프로젝트 소개



장르 : FPS

개발 시기 : 2019/10~2019/11

개발 목적 : 3D 이론과 서버를 연동한 멀티 플레이 게임 개발

개발 툴 : OpenGL, VS2019, GitHub

개발 언어 : C++

팀원 : 2인

담당 역할 : 클라이언트 및 서버

개발 중점 사항

- 프레임 워크 제작
- MFC를 이용한 Map tool 구현
- TCP/IP와 멀티쓰레드를 사용한 서버 연동
- 충돌처리 및 게임 콘텐츠 개발



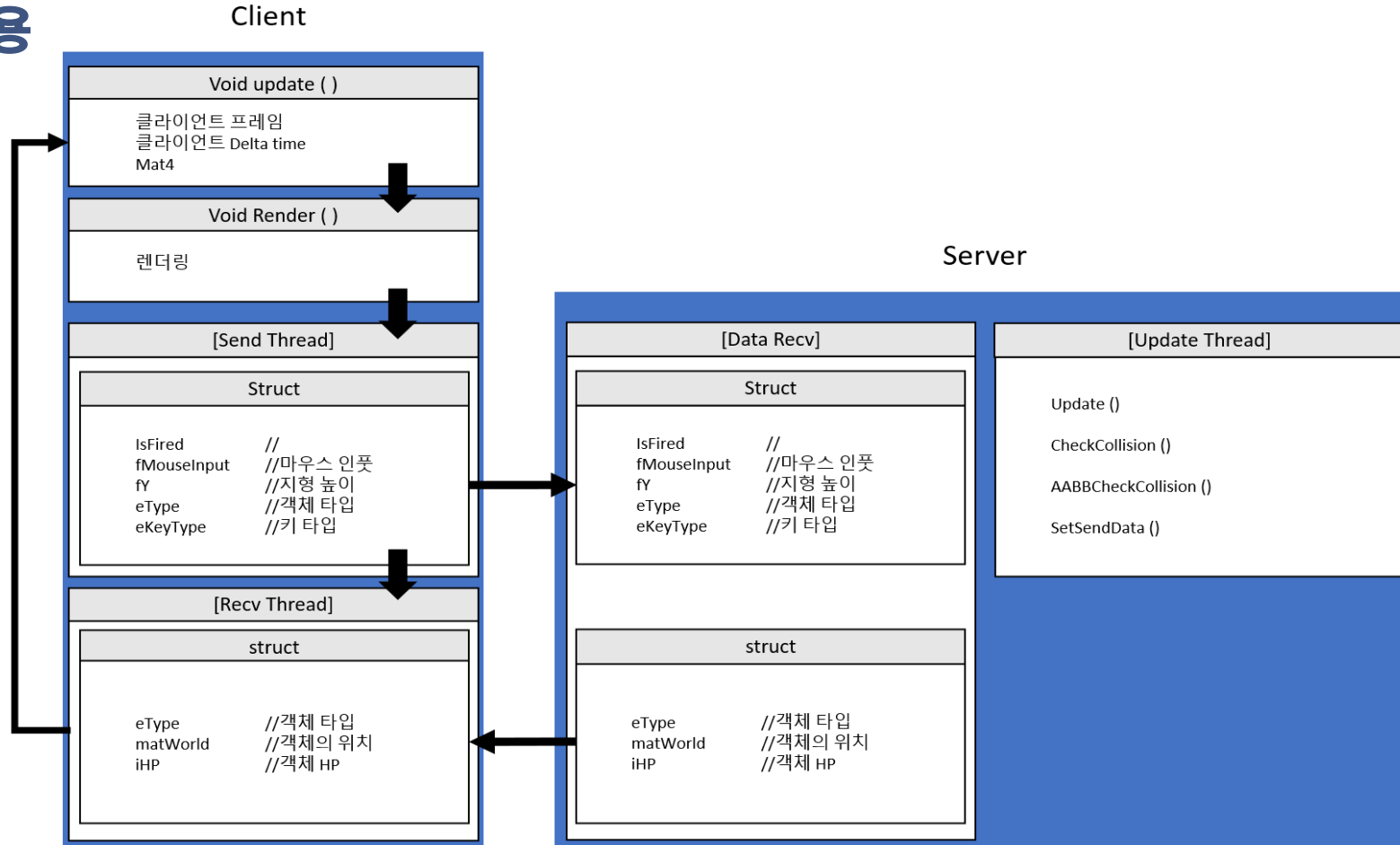
---

YouTube: <https://www.youtube.com/watch?v=vSf8Tr1ftjw>

# 프로젝트 경험 프레임워크

## 04. World of Tanks

### 세부 개발 내용



- TCP/IP 서버를 활용
- 업데이트 쓰레드를 통해 게임의 충돌처리 및 기타 연산 수행
- Mutex를 이용한 동기화 처리

# 프로젝트 경험 패킷(1/2)

## 04. World of Tanks

### 세부 개발 내용

```
#pragma pack(push,1)
typedef struct tagServer_Send
{
    bool        IsFired = false;
    float       fMouseInput = 0.f;
    float       fY = 0.f;
    CONTROLL_TYPE eType = TYPE_END;
    KEY_TYPE     eKeyType = KEY_END;
}SERVERSEND;

typedef struct tagSever_Recv
{
    CONTROLL_TYPE eType = CONTROLL_TYPE::TYPE_END;
    mat4          matWorld = mat4(1.f);
    int           iHP = 0;
}SERVERRECV;

typedef struct tagLogin_SendData
{
    char          szPlayerID[256] = {};
    CONTROLL_TYPE eType;
}LOGINDATA;
#pragma pack(pop)
```

[프로토콜 정의]

```
CServerMgr::GetInstance()->GetSendData().push_back(m_tSendData);
```

각 오브젝트들의 업데이트 끝날 때 서버 매니저에 vector에 데이터를 저장합니다.

```
DWORD WINAPI SendThread(void* pArg)
{
    while (1)
    {
        WaitForSingleObject(CServerMgr::GetInstance()->GetSendHandle(), INFINITE);

        int iLen = CServerMgr::GetInstance()->GetSendData().size();
        send(CServerMgr::GetInstance()->GetSocket(), (char*)&iLen, sizeof(int), 0);

        for (int i = 0; i < iLen; ++i)
        {
            int size = sizeof(SERVERSEND);
            SERVERSEND temp = CServerMgr::GetInstance()->GetSendData()[i];
            int iRetVal = send(CServerMgr::GetInstance()->GetSocket(), (char*)&temp, sizeof(SERVERSEND), 0);
        }

        CServerMgr::GetInstance()->GetSendData().clear();
        ResetEvent(CServerMgr::GetInstance()->GetSendHandle());
        SetEvent(CServerMgr::GetInstance()->GetRecvHandle());
    }

    return 0;
}
```

[클라이언트의 패킷 송신]

# 프로젝트 경험 패킷(2/2)

## 04. World of Tanks 세부 개발 내용

```
DWORD WINAPI RecvThread(void* pArg)
{
    while (1)
    {
        int iLen = 0;

        //CServerMgr::GetInstance()->GetRecvData().clear();
        WaitForSingleObject(CServerMgr::GetInstance()->GetRecvHandle(), INFINITE);

        recvn(CServerMgr::GetInstance()->GetSocket(), (char*)&iLen, sizeof(int), 0);

        for (int i = 0; i < iLen; ++i)
        {
            SERVERRECV temp = {};
            recvn(CServerMgr::GetInstance()->GetSocket(), (char*)&temp, sizeof(SERVERRECV), 0);

            if (temp.eType == TYPE_BULLET)
            {
                CServerMgr::GetInstance()->GetRecvData().push_back(temp);
            }

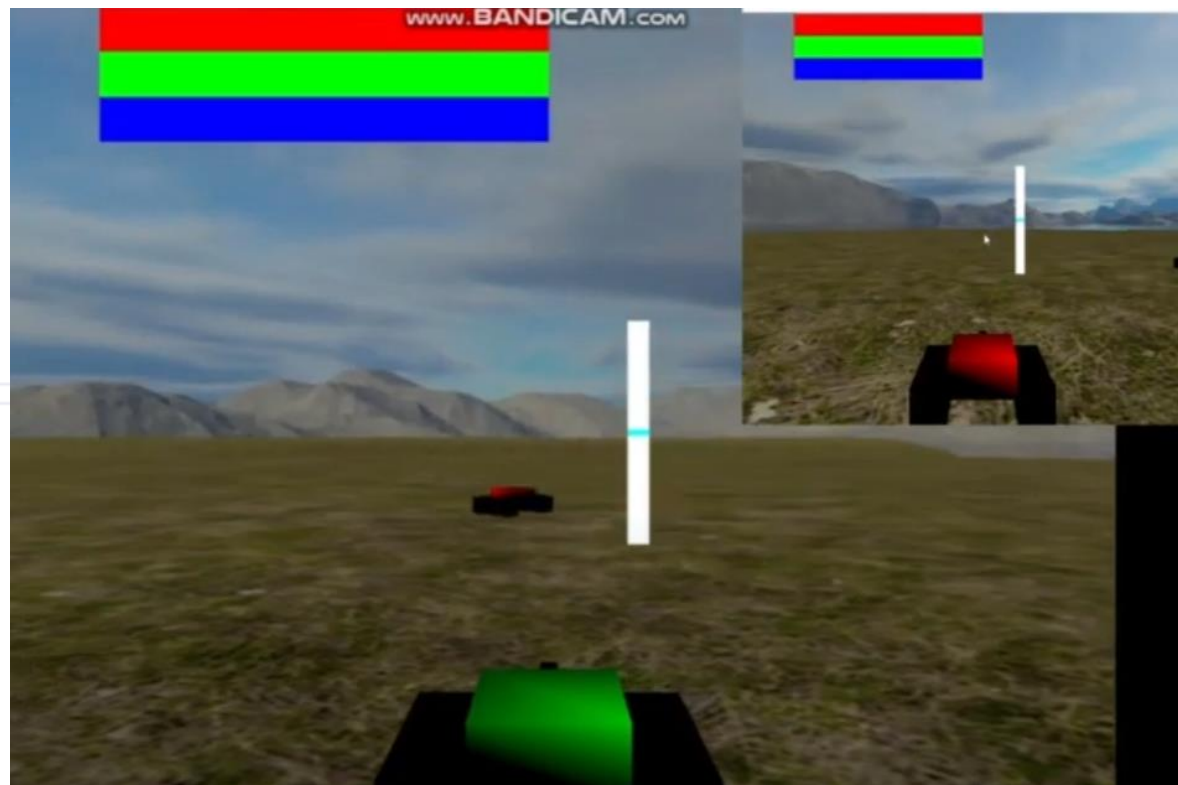
            if (temp.eType == TYPE_PLAYER_A)
            {
                CObjectMgr::GetInstance()->GetObject_List(OBJECT_PLAYER_A)[0]->GetTransform()->SetMatrix(temp.matWorld);
                CObjectMgr::GetInstance()->GetObject_List(OBJECT_PLAYER_A)[0]->SetHP(temp.iHP);
            }
            else if (temp.eType == TYPE_PLAYER_B)
            {
                CObjectMgr::GetInstance()->GetObject_List(OBJECT_PLAYER_B)[0]->GetTransform()->SetMatrix(temp.matWorld);
                CObjectMgr::GetInstance()->GetObject_List(OBJECT_PLAYER_B)[0]->SetHP(temp.iHP);
            }
            else if (temp.eType == TYPE_PLAYER_C)
            {
                CObjectMgr::GetInstance()->GetObject_List(OBJECT_PLAYER_C)[0]->GetTransform()->SetMatrix(temp.matWorld);
                CObjectMgr::GetInstance()->GetObject_List(OBJECT_PLAYER_C)[0]->SetHP(temp.iHP);
            }
        }

        ResetEvent(CServerMgr::GetInstance()->GetRecvHandle());
    }

    return 0;
}
```

[클라이언트의 패킷수신]

•Recv 쓰레드의 종료를 기다린 뒤, 서버에서 각 플레이어의 Matrix와 체력을 계산한 후 각 플레이어에게 전달합니다.



# 프로젝트 경험 오브젝트 구조

## 04. World of Tanks 세부 개발 내용



```
class CObj;
class CObjectMgr
{
    __DECLARE_SINGLETON(CObjectMgr)
private:
    CObjectMgr();
    ~CObjectMgr();
public:
    void    Add_Object(CObj* pObj, OBJECT eID);
    void    Update(_float fTimeDelta);
    void    Render();
    void    Release();
public:
    void    Group_Release(OBJECT eID);

public:
    vector<CObj*>  GetObject_List(OBJECT eID) { return m_vecObj[eID]; }
    CObj*         Find_Light(string strLightName);
public:
    vector<CObj*>  m_vecObj[OBJECT_END];
};
```

- 모든 오브젝트들은 CObj 클래스를 상속 받습니다.
- 모든 오브젝트들은 CObjectMgr를 통해 생성 및 관리 합니다.

**감사합니다.**

---