포트폴리오

서버프로그래머 지원

목차
CONTENTS

HACKED!

목적: HACKED 네트워크 지원

개발 기간 : 2019년 1월 ~

사용 도구 : C++(서버), UE4.22(클라이언트)

개발 인원 : 13명 (서버 1명, 클라이언트 2명)

담당 업무 : IOCP-EPOLL 크로스 플랫폼 서버 구현.
스팀 매칭과 레플리케이션 시스템 지원.
클라이언트측 서버 대응.
클라이언트 서포트.

HACKED!

https://youtu.be/K72234nFn2E

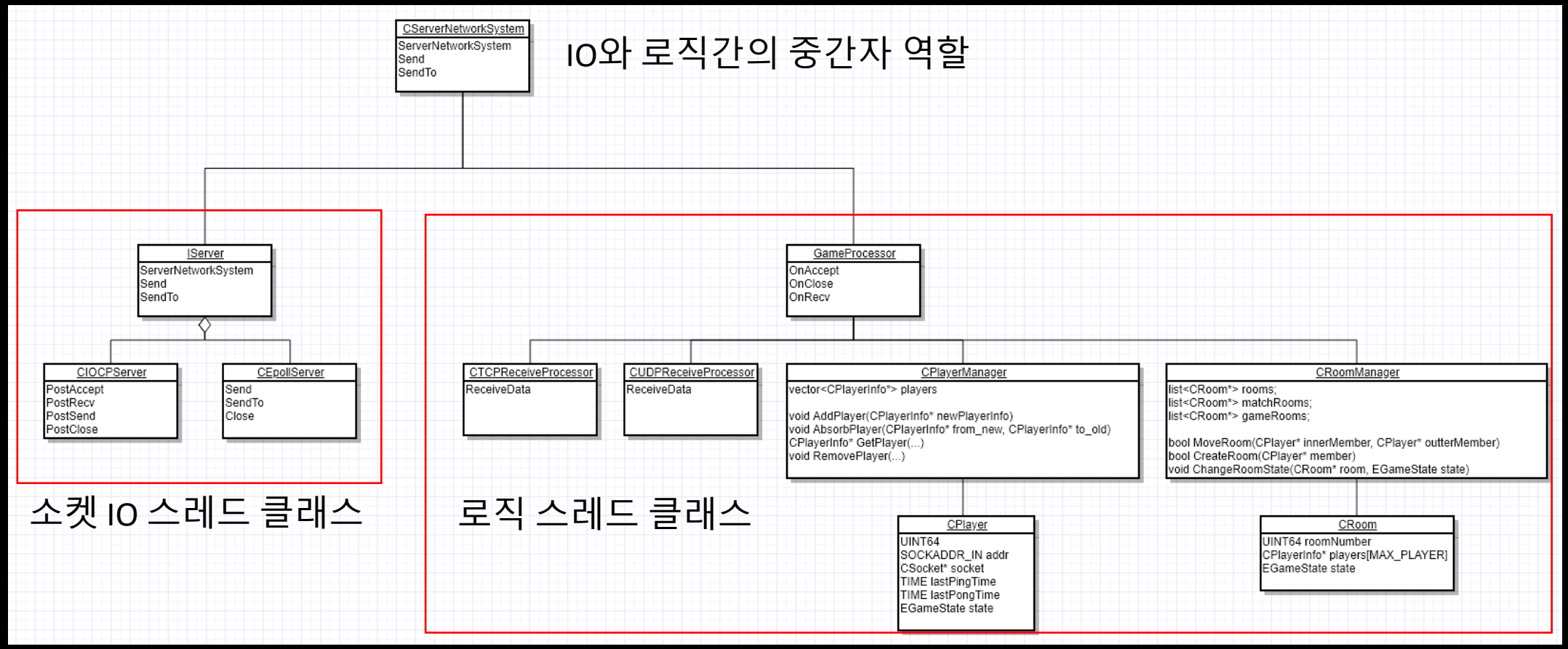
HACKED! 서버
데디게이트 서버 요약

중앙 서버로 스팀 친구를 초대하거나, 모르는 사람과 매칭하여
게임을 시작하는 것을 중재함.

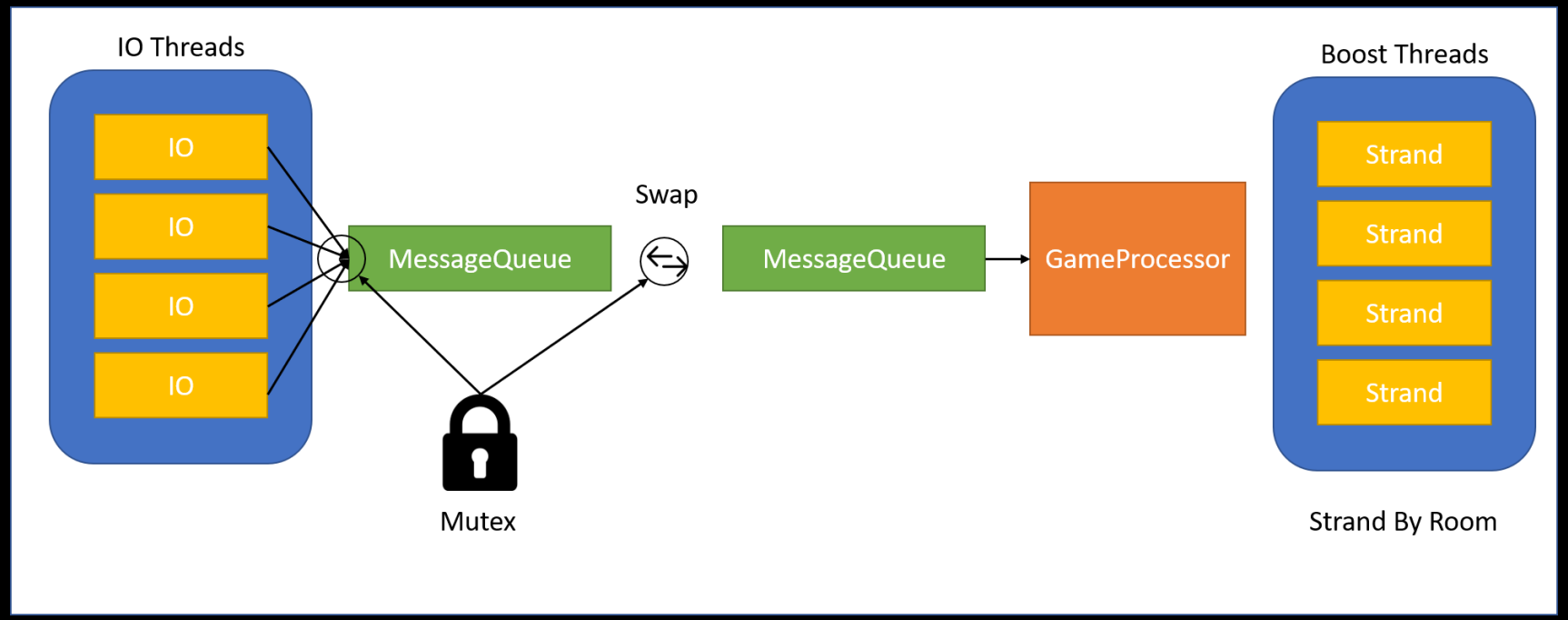
또한, 인게임 리플리케이션 시스템을 중재함.

HACKED! 서버
서버 다이어그램

+ 하트비트 서버 (서버가 죽으면 다시 실행시켜줌.)



HACKED! 서버
서버 도식



IO와 로직을 분리

IO 부분은 IOCP로 메시지 큐에 넣음.
로직 부분은 Boost::Strand를 사용하여 Room 별로 동기화.
로직에서 메시지 큐가 비었으면 메시지 큐를 교체.

HACKED! 서버
IOCP 사용

IOCP를 사용하여
소켓 함수들을
(Send, SendTo, Accept,
Close, Recv, RecvFrom)
비동기화

```
while (true) {
    // Wait until Async IO end
    retval = GetQueuedCompletionStatus(hcp, &cbTransferred, &client_sock,
        (LPOVERLAPPED *)&overlap, INFINITE);

    try {
        switch (overlap->m_type)
        {
        case EOverlappedType::ACCEPT:
        {
            owner->_AcceptProc((FAcceptOverlapped*)overlap);
            break;
        }
        case EOverlappedType::TCP_RECV:
        {
            owner->_RecvProc((FBuffuerableOverlapped*)overlap, cbTransferred, true);
            break;
        }
        case EOverlappedType::UDP_RECV:
        {
            owner->_RecvProc((FBuffuerableOverlapped*)overlap, cbTransferred, false);
            break;
        }
        case EOverlappedType::SEND:
        {
            owner->_SendProc((FSendOverlapped*)overlap);
            break;
        }
        case EOverlappedType::CLOSE:
        {
            owner->_CloseProc((FCloseOverlapped*)overlap);
            break;
        }
        }
    }
```

HACKED! 서버
로비 구현

파티장 위임, 강퇴, 파티 나가기, 친구 초대등을 구현

HACKED! 서버
직렬화/역직렬화

```cpp
struct FC_Reconnect_Server : FPacketStruct {
    FC_Reconnect_Server() {
        type = EMessageType::C_Reconnect_Server;
    }
    UINT64 slot[MAX_PLAYER];
    bool isGameRoom;
    UINT64 myID;

    PACK_STRUCT(slot, isGameRoom, myID);
};
FC_Reconnect_Server msg;
msg.Serialize(buf);
msg.Deserialize(buf);
```

```cpp
// PACK_STRUCT
#define PACK_STRUCT(...)\
virtual size_t GetSize() {\
    return GetSizeOf(__VA_ARGS__) + FPacketStruct::GetSize();\
}\
virtual size_t Serialize(char* buf) {\
    int len = (int)FPacketStruct::Serialize(buf);\
    __Serialize(buf, len, __VA_ARGS__);\
    return len;\
}\
virtual void Deserialize(char* buf, int* cursor) {\
    FPacketStruct::Deserialize(buf, cursor);\
    __Deserialize(buf, cursor, __VA_ARGS__);\
}
```

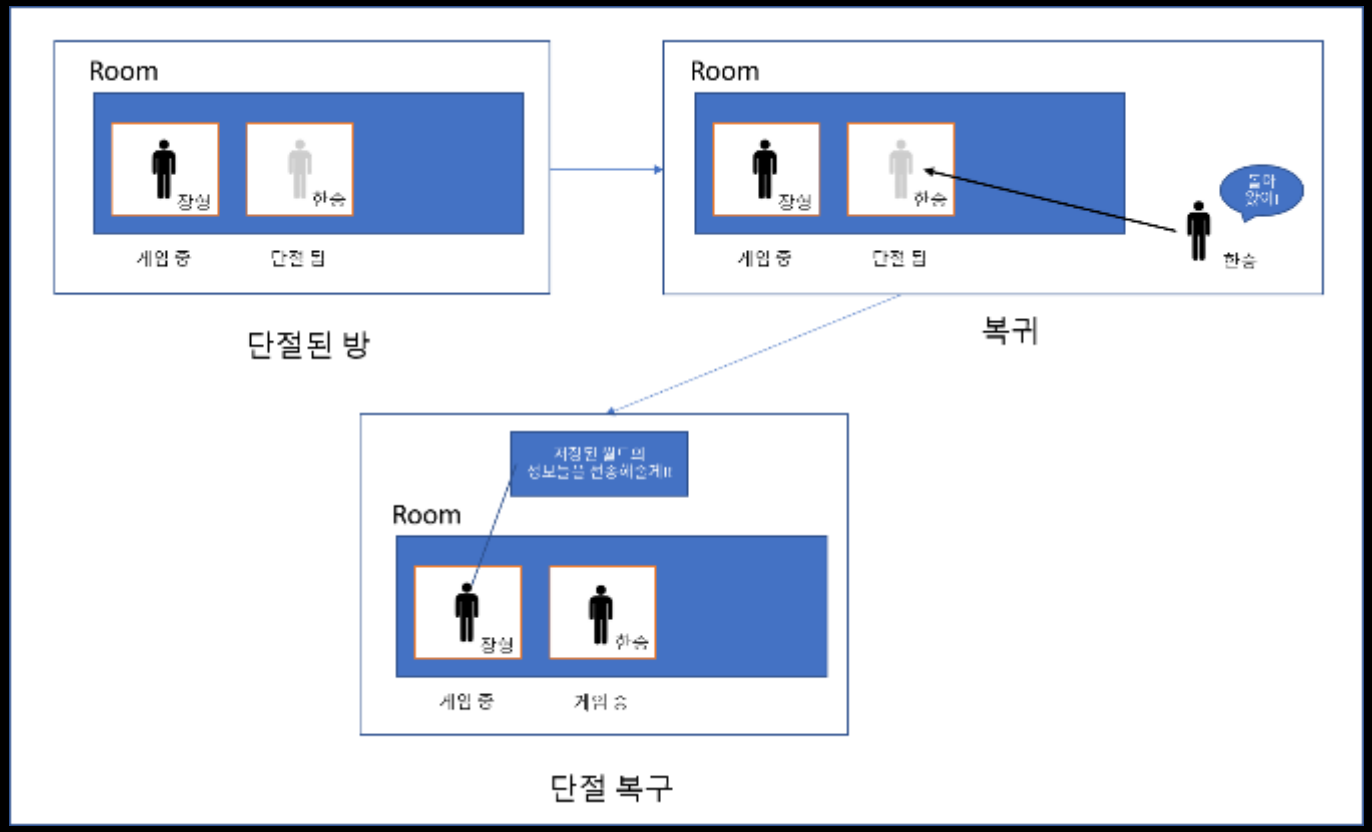
네트워크로 보낼 메세지를 구조체화 하여 사용.
바이트 오더를 고려하여 직렬화/역직렬화하는 템플릿 함수와 매크로 구현.

HACKED! 서버
오브젝트풀

```
// 초대자의 이름을 담아 보낸다.
FS_Lobby_InviteFriend_Request sLobbyInviteFriendRequest;
sLobbyInviteFriendRequest.senderID = senderId;

// 버퍼를 꺼냄
char* buf = CServerNetworkSystem::GetInstance()->bufObjectPool->PopObject();

// 직렬화하여 버퍼에 담음.
sLobbyInviteFriendRequest.Serialize(buf);

// 전송
ServerNetworkSystem->Send(targetUser, buf, (int)sLobbyInviteFriendRequest.GetSize(), true);
WriteLog(ELogLevel::Warning, CLog::Format("C_Lobby_InviteFriend_Request : Invite Send Success.\n"));

// 버퍼 반납
CServerNetworkSystem::GetInstance()->bufObjectPool->ReturnObject(buf);
```

소켓, 버퍼, IOCP Overlapped를 담는 객체들의
생성/삭제 부하를 줄이기 위해 **오브젝트풀 구현.**

HACKED! 서버 재접속



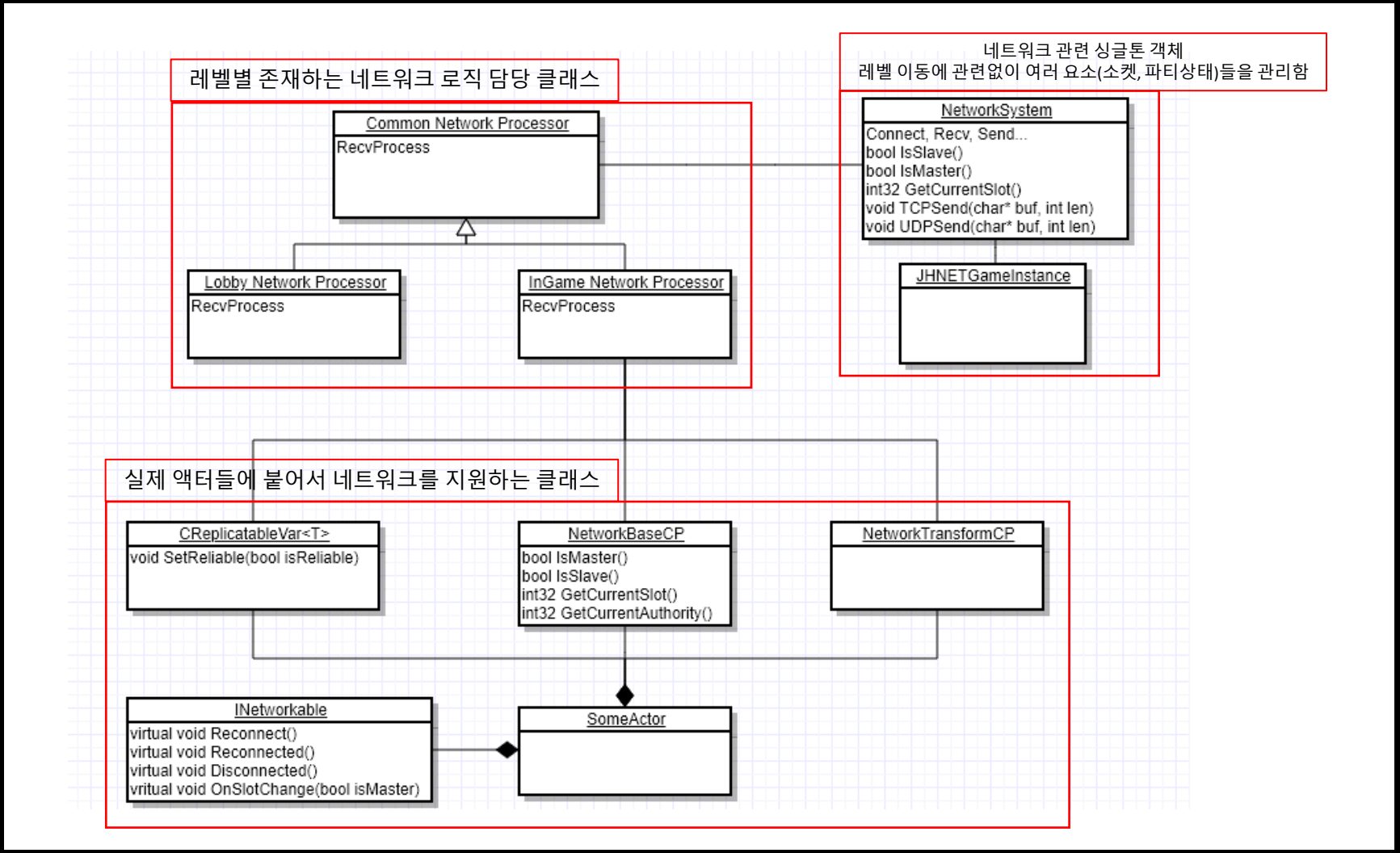
플레이어가 나가면 ConnectionLost 상태로 전환,
같은 SteamID로 재접속시 방을 다시 찾아줌.
클라이언트 부분에서는 마스터 클라이언트의 모든 네트워크 액터에서 복구 로직을 실행함.

HACKED! 클라이언트
클라이언트 요약

함수를 RPC화 하는 매크로 구현.
RPC를 사용하여 위치를 동기화하는 컴포넌트 구현.
RPC를 사용하여 스폰을 동기화하는 함수 구현.

변수를 동기화하는 클래스 구현.

HACKED! 클라이언트
클라이언트 다이어그램



HACKED! 클라이언트
JHNET 모듈화

모듈을 사용하여 구현한 프로젝트
https://github.com/LJH960101/FlockingGame.git

HACKED 게임과 **네트워크 로직**을 완전히 **모듈화**하여
다른 게임에서도 사용할 수 있도록 구현.

HACKED! 클라이언트 RPC

```
RPC_FUNCTION(AInGameNetworkProcessor, _RPCNetworkSpawn, uint64, int, FTransform)
void _RPCNetworkSpawn(uint64 currentSpawnCount, int type, FTransform transform);
void AInGameNetworkProcessor::_RPCNetworkSpawn(uint64 currentSpawnCount, int type, FTransform transform)
{
    RPC(NetBaseCP, AInGameNetworkProcessor, _RPCNetworkSpawn, ENetRPCType::MULTICAST, true, currentSpawnCount, type, transform);
}
void AInGameNetworkProcessor::BeginPlay()
{
    Super::BeginPlay();
    BindRPCFunction(NetBaseCP, AInGameNetworkProcessor, _RPCSetSlotSteamName);
    BindRPCFunction(NetBaseCP, AInGameNetworkProcessor, _RPCNetworkSpawn);
```

매크로를 사용하여 함수를 동기화하는 RPC 시스템 구현.

호출된 함수의 인자를 Buf, len으로 직렬화해서 보내고,
받은 Buf, len을 역직렬화 한 뒤
해당 함수에 꽂아주는 매크로를 만들어서 구현함.

HACKED! 클라이언트 RPC

Buf, len으로 원본 함수를 호출하는 대리자 역할을 하는 함수를 만들어주는 매크로

함수의 인자를 Buf, len으로 직렬화 해서 RPC 요청하는 매크로

buf, len으로 호출이 가능한 함수를 수신을 담당하는 클래스에 바인딩하는 매크로

```
#define RPC_FUNCTION_1(CLASS_NAME, FUNCTION_NAME, ARG1)\
bool CLASS_NAME##_##FUNCTION_NAME##_INLOOP = false;\
UFUNCTION()\
void FUNCTION_NAME##_##(int len, char* buf){\
    CLASS_NAME##_##FUNCTION_NAME##_INLOOP = true;\
    if(len != GetSizeOfBuf<ARG1>(buf)){\
        JHNET_LOG(Error, "Argment Error!");\
        CLASS_NAME##_##FUNCTION_NAME##_INLOOP = false;\
        return;\
    }\
    auto func = std::bind(&CLASS_NAME::FUNCTION_NAME, this, std::placeholders::_1);\
    auto arg1 = JHNETSerializer::TDeserialize<ARG1>(buf, nullptr);\
    func(arg1);\
    CLASS_NAME##_##FUNCTION_NAME##_INLOOP = false;\
}
#define RPC_1(NETBASECP, CLASS_NAME, FUNCTION_NAME, CLASS_AND_FUNCTION_NAME, E_NET_RPC_TYPE, IS_RELIABLE, ARG1)\
{\
    if(NETBASECP && !CLASS_NAME##_##FUNCTION_NAME##_INLOOP){\
        char* __newBuf = NETBASECP->GetNetworkSystem()->bufObjectPool->PopObject();\
        int __len = JHNETSerializer::TSerialize(__newBuf, ARG1);\
        NETBASECP->ExecuteNetFunc(\
            TEXT(#FUNCTION_NAME),\
            E_NET_RPC_TYPE, __len, __newBuf, IS_RELIABLE, false);\
        NETBASECP->GetNetworkSystem()->bufObjectPool->ReturnObject(__newBuf);\
        if(E_NET_RPC_TYPE == ENetRPCType::MASTER && NETBASECP->IsSlave()) return;\
    }\
}
// BindRPCFunction
// You muse use this in PostInitializeComponents or BeginPlay to register function.
// BindRPCFunction(NetBaseCP, AHACKEDCharacter, Move)
#define BindRPCFunction(NETBASECP, CLASS_NAME, FUNCTION_NAME)\
{\
    JHNET_CHECK(this);\
    JHNET_CHECK(NETBASECP);\
    auto rpcDelegate = NETBASECP->CreateBindableDelegateFunction(TEXT(#FUNCTION_NAME));\
    JHNET_CHECK(rpcDelegate);\
    rpcDelegate->BindUObject(this, &CLASS_NAME::FUNCTION_NAME##_##);\
}
```

HACKED! 클라이언트 스폰, 트랜스폼 동기화

```
RPC_FUNCTION(AInGameNetworkProcessor, _RPCNetworkSpawn, uint64, int, FTransform)
void _RPCNetworkSpawn(uint64 currentSpawnCount, int type, FTransform transform);
```

```
UCLASS( ClassGroup=(Custom), meta=(BlueprintSpawnableComponent) )
class JHNET_API UNetworkTransformCP : public UActorComponent
{
public:
    // Sets default values for this component's properties
    UNetworkTransformCP();

    RPC_FUNCTION(UNetworkTransformCP, __RPCSyncLocation, FVector)
        void __RPCSyncLocation(FVector location);

    RPC_FUNCTION(UNetworkTransformCP, __RPCSyncRotation, FVector)
        void __RPCSyncRotation(FVector rotation);

    RPC_FUNCTION(UNetworkTransformCP, __RPCSyncControlRotation, FVector)
        void __RPCSyncControlRotation(FVector targetVector);
```

RPC 시스템을 사용하여 네트워크 스폰 동기화,
위치 회전 동기화 컴포넌트 구현.

HACKED! 클라이언트 변수 동기화

```
CReplicatableVar<float> _currentSpeed;
CReplicatableVar<bool> _bOnJump;
CReplicatableVar<float> _currentForwardSpeedRate;
CReplicatableVar<float> _currentRightSpeedRate;
```

```
_bOnJump.Init(&NetBaseCP, TEXT("bOnJump"));
_bOnJump.SetReliable(true);

_currentSpeed.Init(&NetBaseCP, TEXT("currentSpeed"));
_currentSpeed.SetReliable(false);

_currentForwardSpeedRate.Init(&NetBaseCP, TEXT("currentForwardSpeedRate"));
_currentForwardSpeedRate.SetReliable(false);

_currentRightSpeedRate.Init(&NetBaseCP, TEXT("currentRightSpeedRate"));
_currentRightSpeedRate.SetReliable(false);
```

변수를 동기화하는 템플릿 클래스 구현.

HACKED! 클라이언트 변수 동기화

```cpp
CReplicatableVar<T>& operator=(const CReplicatableVar<T>& rhs) {
    ChangeProcess(rhs._value);
    return *this;
}
CReplicatableVar<T>& operator=(const T& rhs) {
    ChangeProcess(rhs);
    return *this;
}
```

```cpp
void ChangeProcess(const T& newValue) {
    if (_netBaseCP == nullptr) { ... }
    if (*_netBaseCP == nullptr) { ... }
    if (IsChanged(newValue)) {
        // Something is change.
        if (_handle.IsEmpty() || !(*_netBaseCP)->JHNET_CHECKAlreadyUseSyncVar(_handle)) { ... }

        // SyncVar sync proc...
        if (!_handle.IsEmpty()) {
            char* buf = (*_netBaseCP)->GetNetworkSystem()->bufObjectPool->PopObject();
            int len = JHNETSerializer::TSerialize<T>(buf, _value);
            (*_netBaseCP)->SetSyncVar(_handle, sizeof(newValue), buf, _bReliable);
            _netBaseCP->GetNetworkSystem()->bufObjectPool->ReturnObject(buf);
        }
    }
}
```

값이 변하면 전송하도록 구현.

HACKED! 클라이언트 변수 동기화

위에서 구현한 내용들을 바탕으로

캐릭터 동기화(이동 공격 죽음등),
몬스터 동기화(이동 공격 죽음등),
게임 플로우 동기화,
스포너(몬스터 발생기) 구현 및 동기화,
웨이브 시스템 구현 및 동기화,
퀘스트 자막 시스템 구현 및 동기화,
캐릭터 선택 구현 및 동기화

등을 하였습니다.

INFO

목적: 학사관리 시스템 구현

개발 기간 : 2016년 9월 ~ 2017년 3월

사용 도구 : Angular2, DreamFactory (REST), Oracle, HTML, CSS

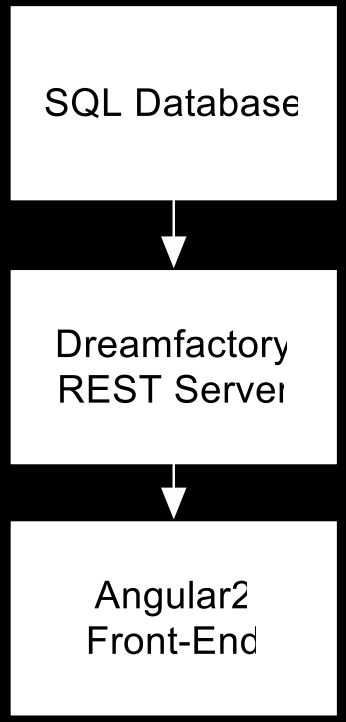
개발 인원 : 6명

담당 업무 : DreamFactory Back-End Script 및 Angular2 Front-End Script

INFO


대시보드
CKINFO
주요 링크
버스 예약
도서관
학사 일정
증명서 발급
공지사항
장학 제도
실시간 상담
출결 상황
주차 정보
오늘의 강의정보
< 성공적인프로젝트를위한대인관계 >
주제 : 성공적 갈등관리 방법
< 초보자를위한영어 >
주제 : 중간고사
과제 : be동사 과거형 문제풀기
오늘의 식단
< 휘당 >
고깃국 옥수수밥 깐김자
성적 그래프
나의 성적
학년 성적
전체 성적
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
1학년1학기
1학년2학기
2학년1학기
도서 대여 정보
청강문화산업대학교
대학홈페이지 도서관 개인정보처리방침 영상정보
실시간 상담-메시지 남기기

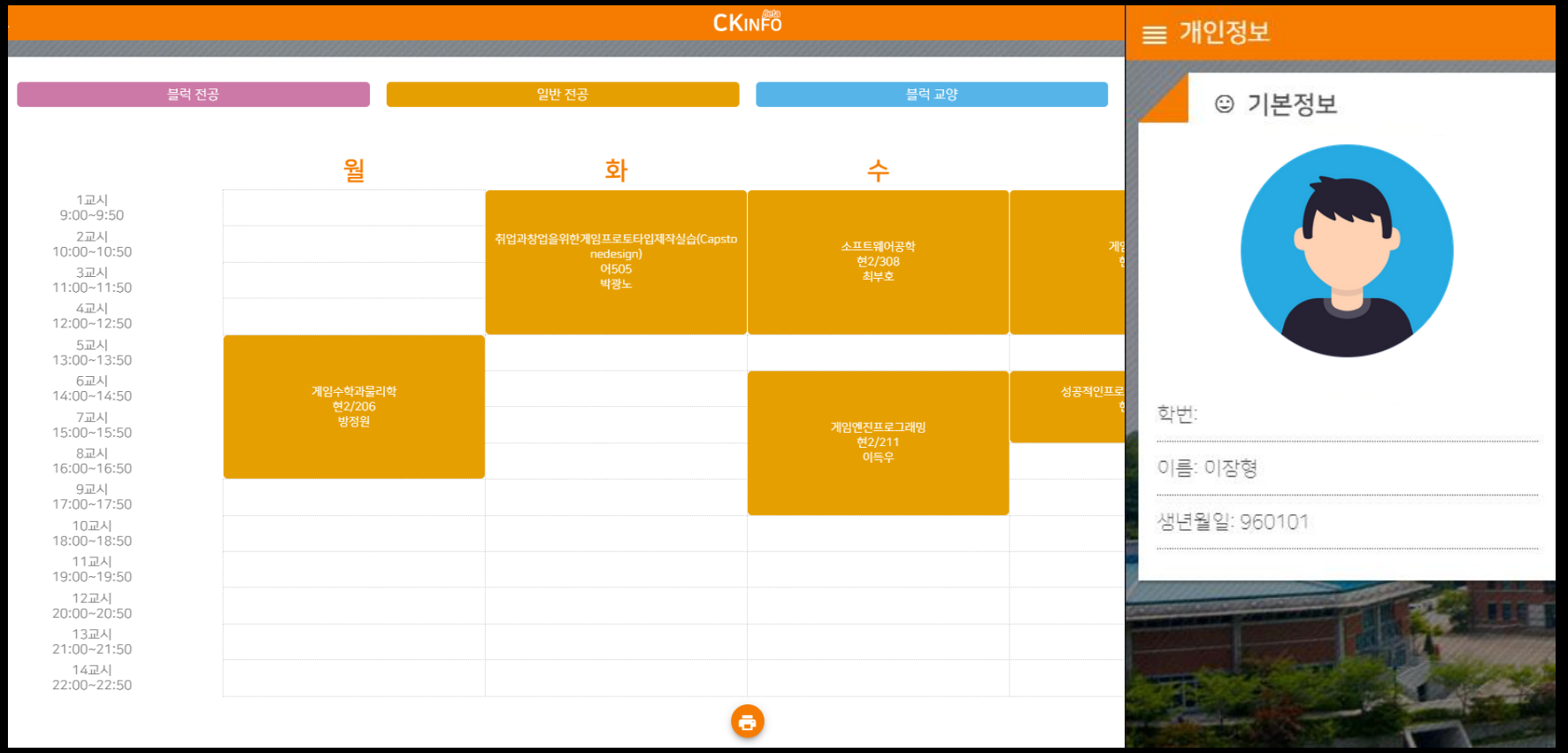
INFO



기존 ActiveX를 사용하는 학생 시스템을
대체하려는 목적으로 시작된 프로젝트.

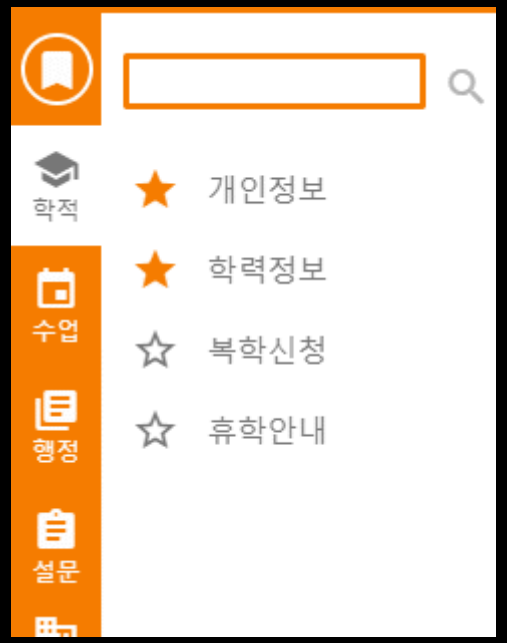
SPA 웹으로 Route 이동 간의 로딩이 없음.

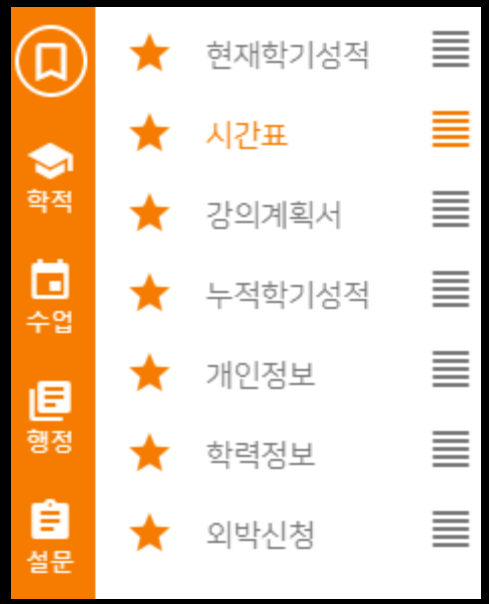
INFO



단순한 Select 쿼리나 뷰 프로시저 등을
REST API를 사용하여 구현한 조회 업무

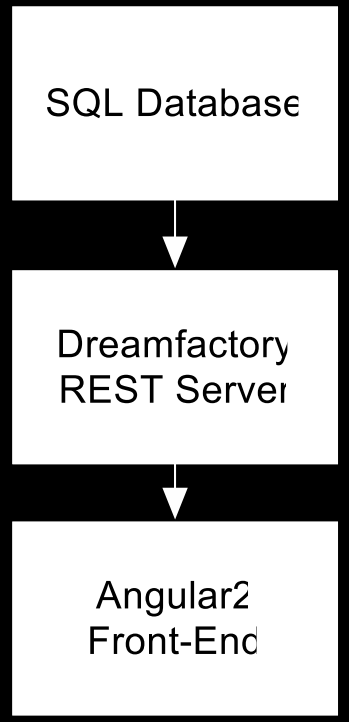
INFO

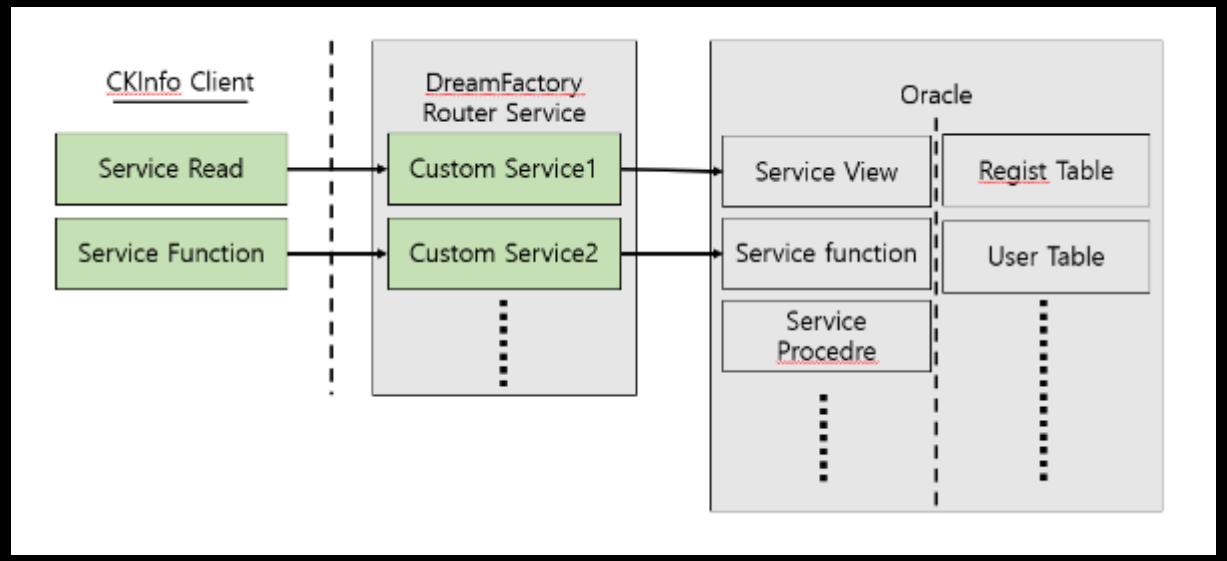




테이블을 직접 만들고 **Insert, Update**와 **Function**을
활용하여 **게시판과 즐겨 찾기 시스템** 등을 **구현**

INFO





DreamFactory 단에서는 PHP 스크립트를 사용하여 기존 DB등의 시스템과 연동하는 서비스들을 구현함.

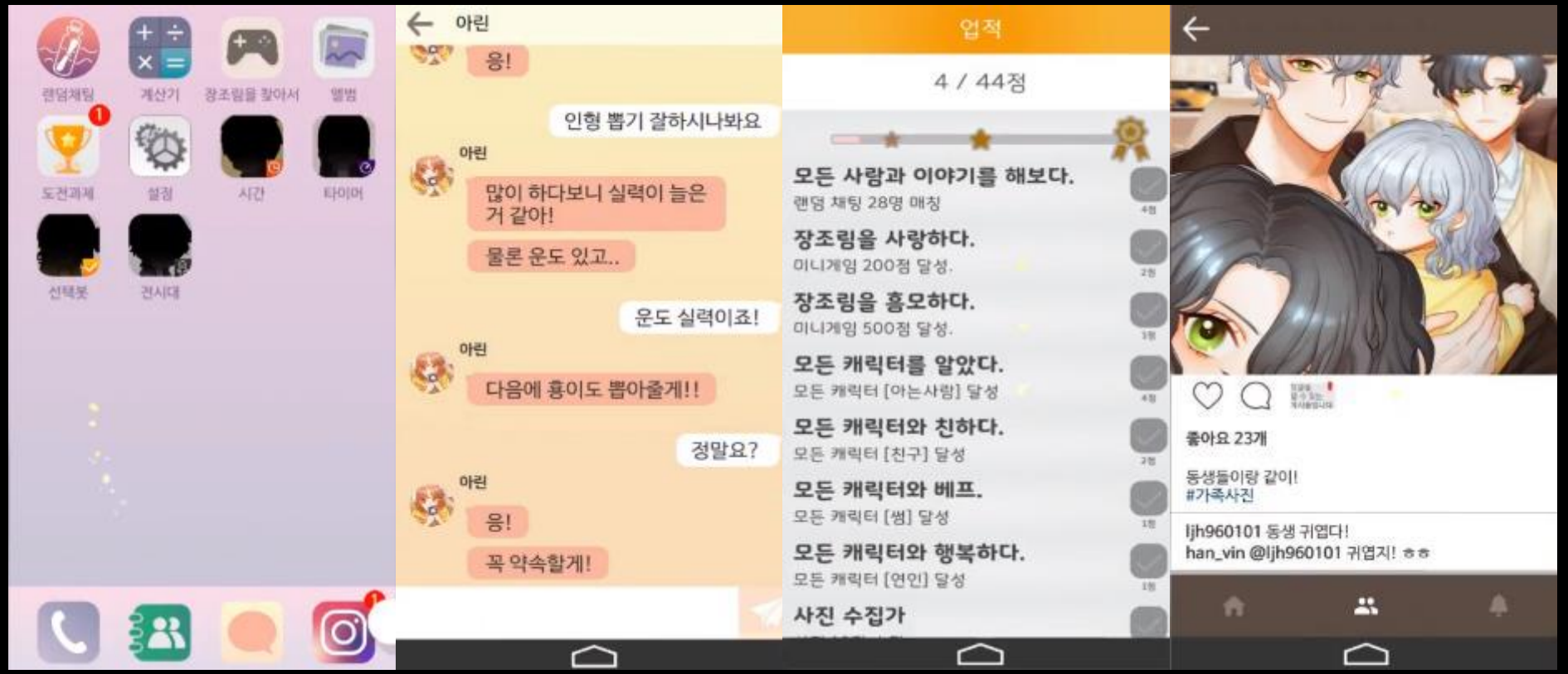
내폰속로맨스

목적: 스마트폰을 모작하기
개발 기간 : 2018년 3월 ~ 2018년 6월
사용 도구 : C# Form, Unity
개발 인원 : 5명 (프로그래머 1명)
담당 업무 : 게임 구현, 에디터 제작

내폰속로맨스

https://youtu.be/HUcwjtxZJF8

내폰속로맨스



아이폰과 유사한 UX를 구현.

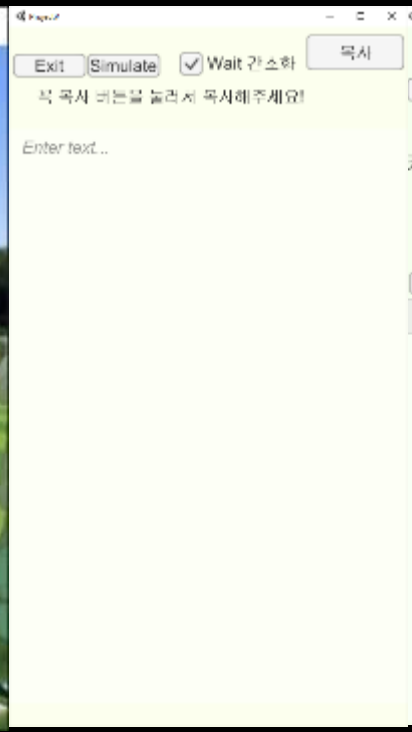
홈화면, 어플, 푸시알람, 어플간의 통신 구현

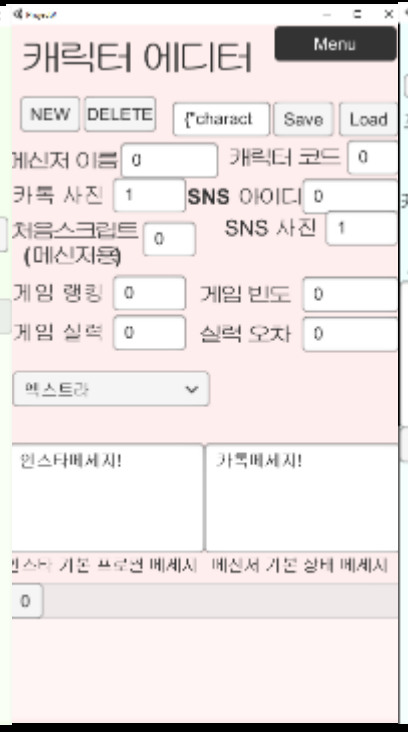
개별어플 14개 구현 (랜덤채팅, 메신저, 인스타, 사진수집, 도전과제 등)

내폰속로맨스

캐릭터 호감도 시스템,
채팅 스크립트 시스템,
SNS 시스템,
기타 시스템 구현

내폰속로맨스





스크립트 편집을 돕기 위한 **툴 제작**

PPAP

목적: 언리얼 네트워크 시스템 공부

개발 기간 : 2018년 10월 1주 ~
2018년 10월 4주

사용 도구 : 언리얼

개발 인원 : 9명 (프로그래머 2명)

담당 업무 : 네트워크 시스템 구현, UI 구현
로비 구현, 미니게임 1종 구현

PPAP

https://youtu.be/CHpKUIAikBY

PPAP

언리얼 네트워킹 시스템과 Steam API를 연동하여
Steam 친구와 P2P 게임 구현

PPAP

언리얼 네트워킹 시스템을 사용하여 월드 이동 및 여러
네트워킹 로직과 동기화 UI를 구현.

팝콘랜드

목적: 유니티 네트워크 시스템 공부

개발 기간 : 2018년 11월 2주 ~
2018년 11월 4주

사용 도구 : 유니티, UNET

개발 인원 : 6명 (프로그래머 1명)

담당 업무 : 多vs多 슈팅게임 구현

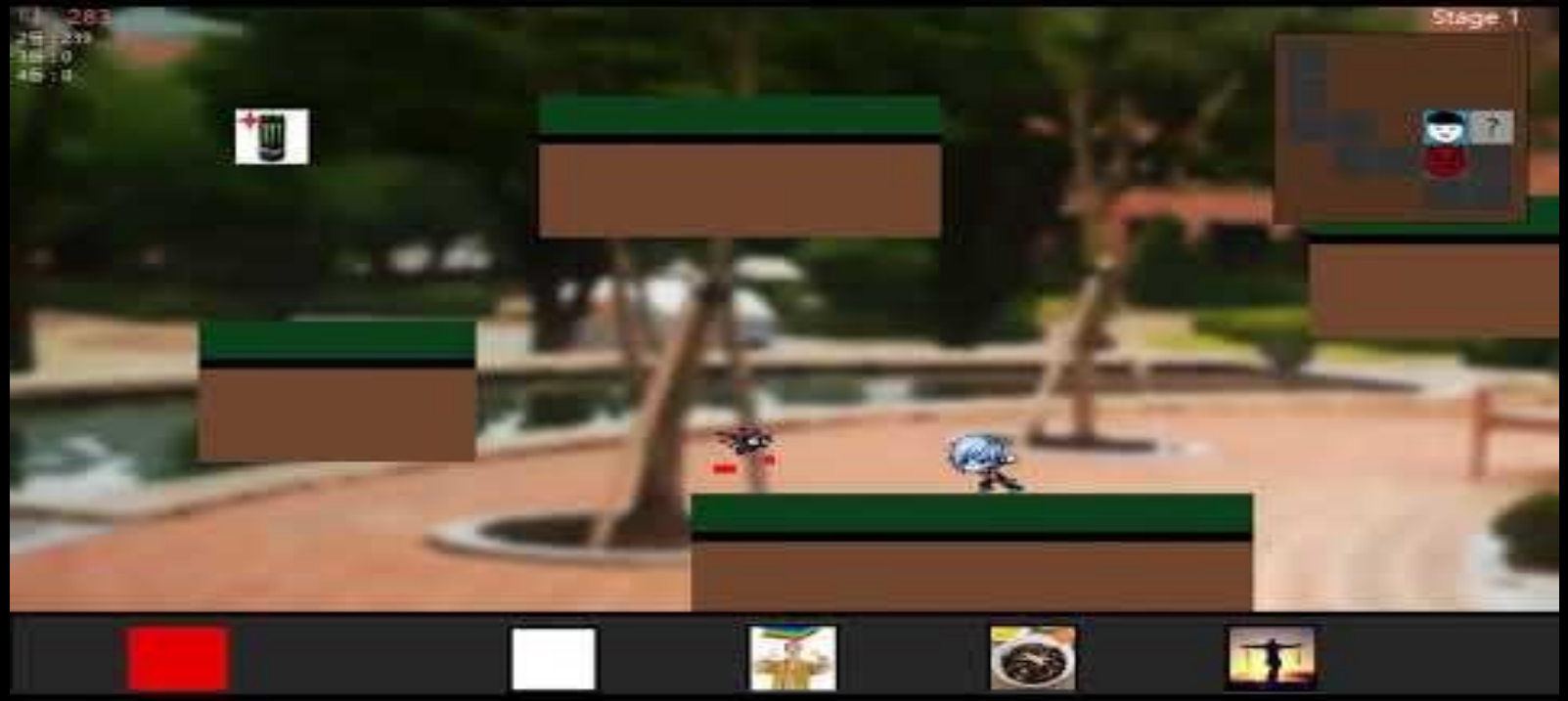
팝콘랜드

https://youtu.be/ZHTIcU0Tiu0

팝콘랜드

즉시 발사하는 총알에 대한 보간,
캐릭터의 데드레커닝 등 그럴듯하게 보이는 화면 연출에 초점.
점수, 장비, 필살기, 중립 몬스터 등 여러 네트워크 요소 단기간 구현.

기타이력

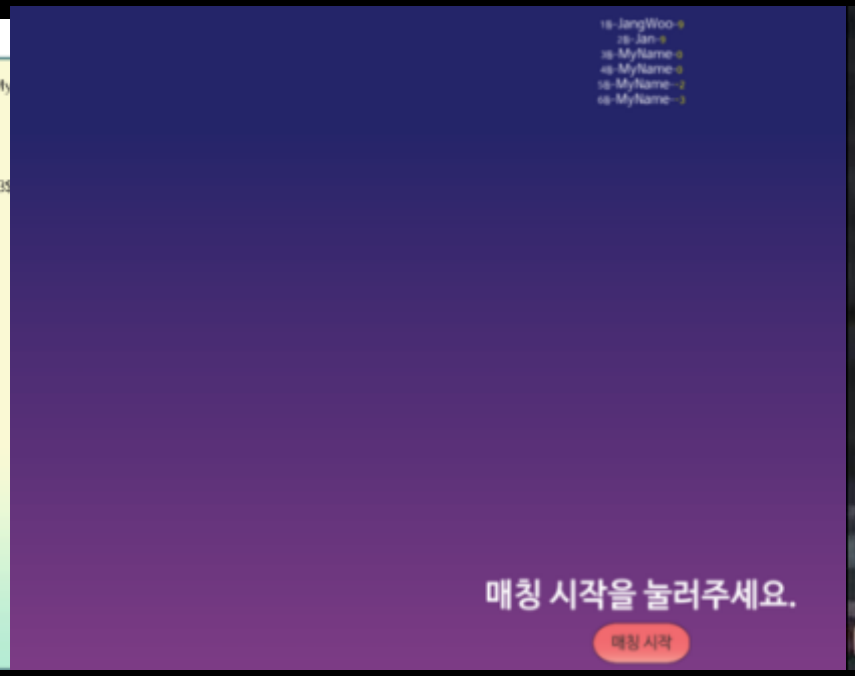


Winapi로 구현한 로그라이크형 2D 슈팅 게임

20종 가까이 되는 무기를 람다를 사용하여 빠르게 구현함. (1학년 2학기 기말과제 2주)

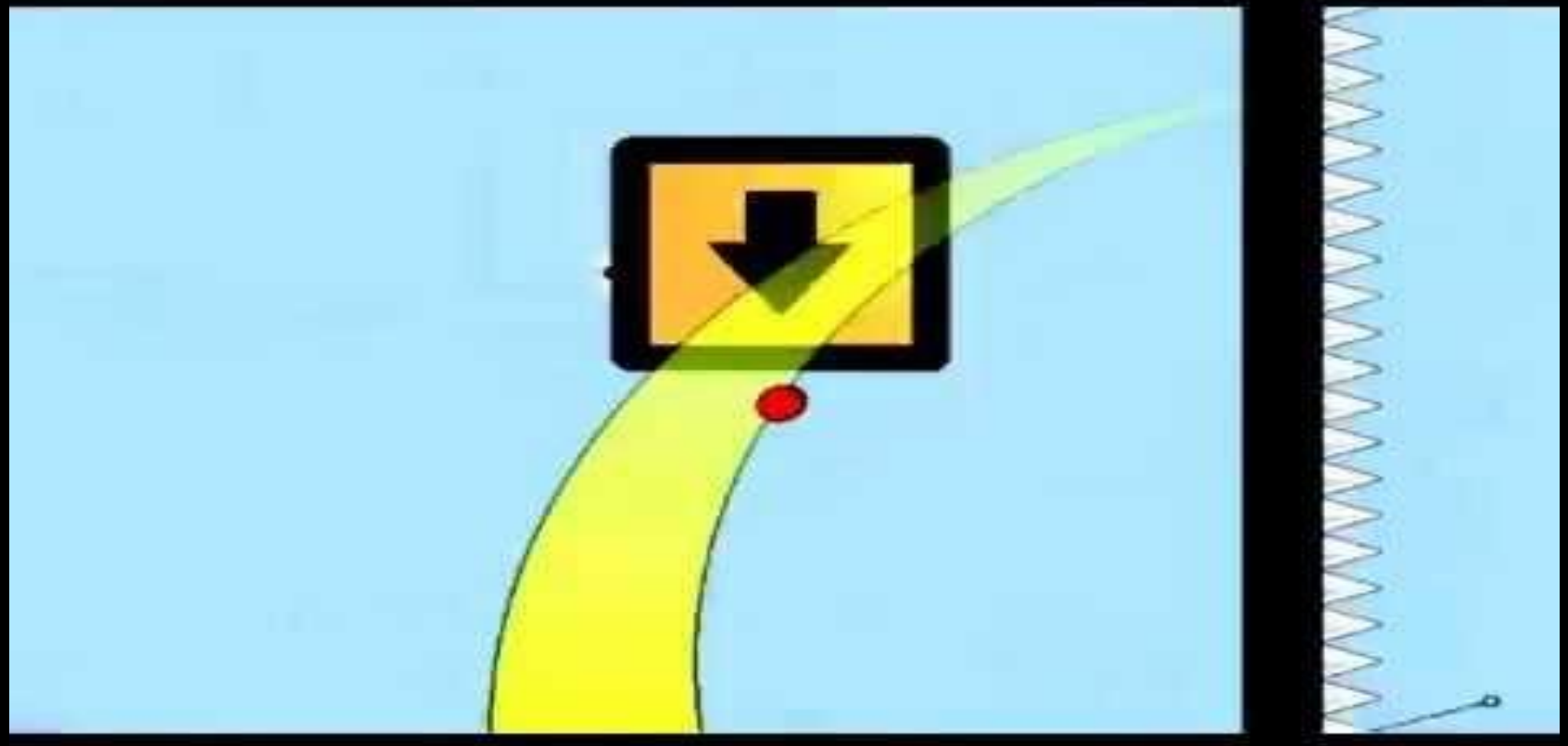
https://youtu.be/Zh3F4uMQqd0

기타이력



매칭과 랭킹을 DB에 담는 역할을 하는 서버를 구현,
UDP 통신으로 3가지 무기를 사용하는 1:1 FPS 대전 구현.
UDP에 최소한의 신뢰성을 보태기 위해, 열차 형식으로 메시지 전달. (2학년 1학기 기말 - 2주)
https://youtu.be/4InaaCtPlFw

기타이력



공을 튕기면서 스테이지를 깨는 퍼즐 형식의 게임.

혼자서 하나의 프로젝트를 완성하고, Play Store에 등록 및 광고와 결제 시스템 적용 공부 목적.

https://youtu.be/BBvYsbzPoa8

https://play.google.com/store/apps/details?id=com.Pepe.ProjectP

기타이력

오큘러스 리프트를 활용한 체감형 교육게임.
순찰자들을 피해 단서를 모으고 퀴즈를 풀어 탈출하는 게임.
언리얼 엔진 기초 공부 목적.

감사합니다