**네트워크 게임 프로그래밍**

**Term Project 추진 계획서**

게임공학과

2016182039 정지우

2017180016 서진석

2017180021 어수혁

**목차**

1. 애플리케이션 기획
2. High-Level 디자인
3. Low-Level 디자인
4. 팀원 역할 분담
5. 개발환경
6. 개발일정

**1. 애플리케이션 기획**

* 게임 : 들고 튀어라!
* 2인용 아케이드 술래잡기 게임
* 설명

게임 시간이 종료될 때 코인을 들고 있는 사람이 이기는 술래잡기 게임으로서, 게임이 시작되면 맵의 어딘가에 생성되는 코인을 주워서 상대방으로부터 멀리 도망치는 2인용 게임이다.

각각의 플레이어는 일반 공격, 스킬, 대쉬를 갖고 있으며 각각의 쿨타임은 캐릭터마다 다르다. 공격이나 스킬을 상대에게 적중시킬 경우 상대는 잠시동안 움직일 수 없으며 들고있던 코인을 떨어뜨리게 된다.

맵은 이동 가능 타일, 속도가 빨라지는 가속 타일, 지나갈 수 없는 이동 불가 타일로 이루어져 있다. 단, 피격시 밀려나는 효과와 대시로는 이동 불가 타일을 지나갈 수 있다.

화면의 절반은 자신의 캐릭터 주변이, 나머지 절반은 상대의 캐릭터 주변이 보인다.

게임 시간이 종료될 때 아무도 코인을 갖고 있지 않다면 게임이 끝나지 않으며, 먼저 코인을 줍는 사람이 즉시 승리하게 된다.

* 조작

방향키 : 이동

Z : 스킬

X : 공격

C : 대쉬

게임 스크린샷

캐릭터 선택창

텍스트이(가) 표시된 사진

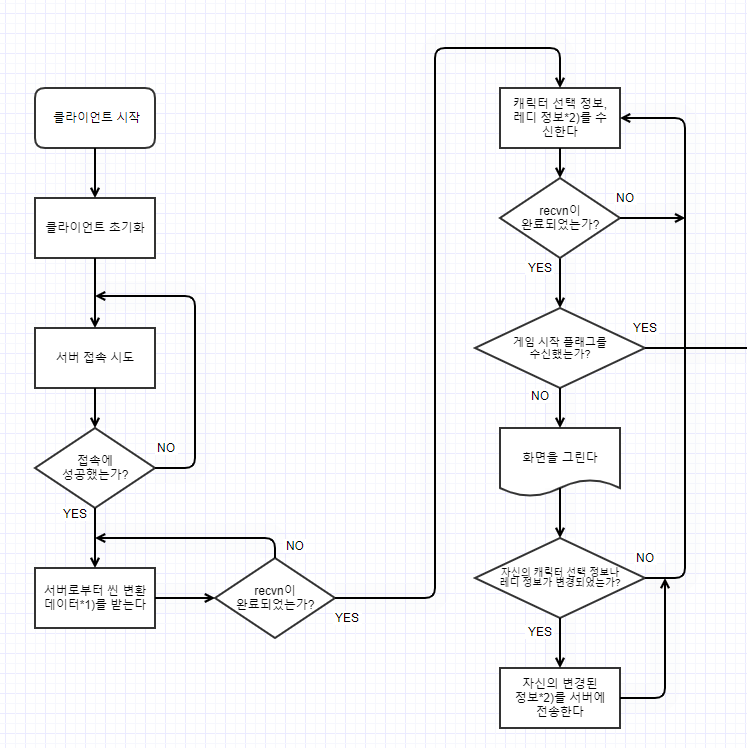
자동 생성된 설명

인게임

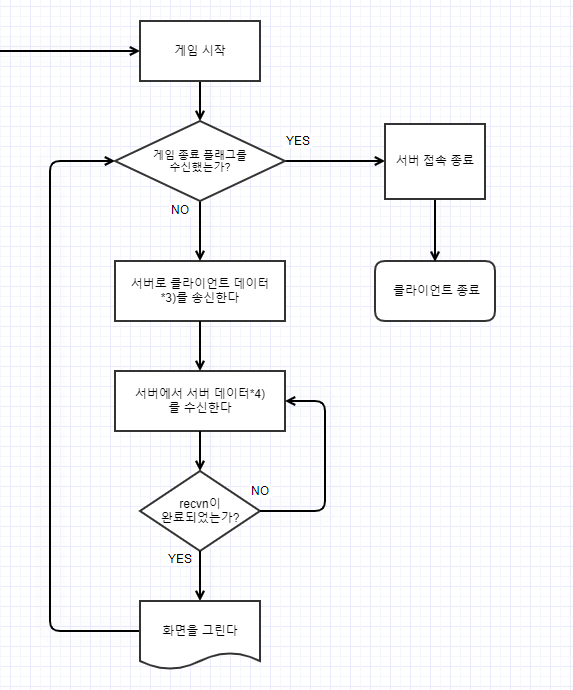


**2. High-Level 디자인**

클라이언트

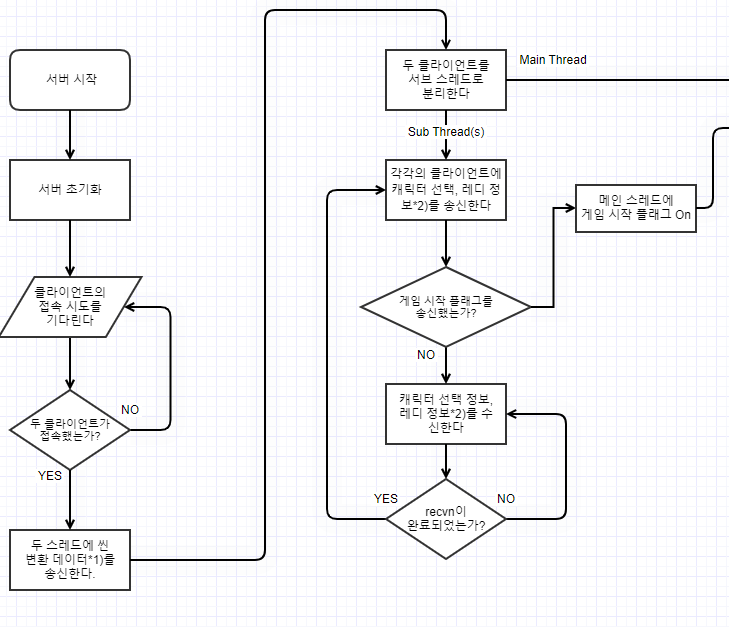


서버로 접속 –> 캐릭터 선택창 플로우 차트

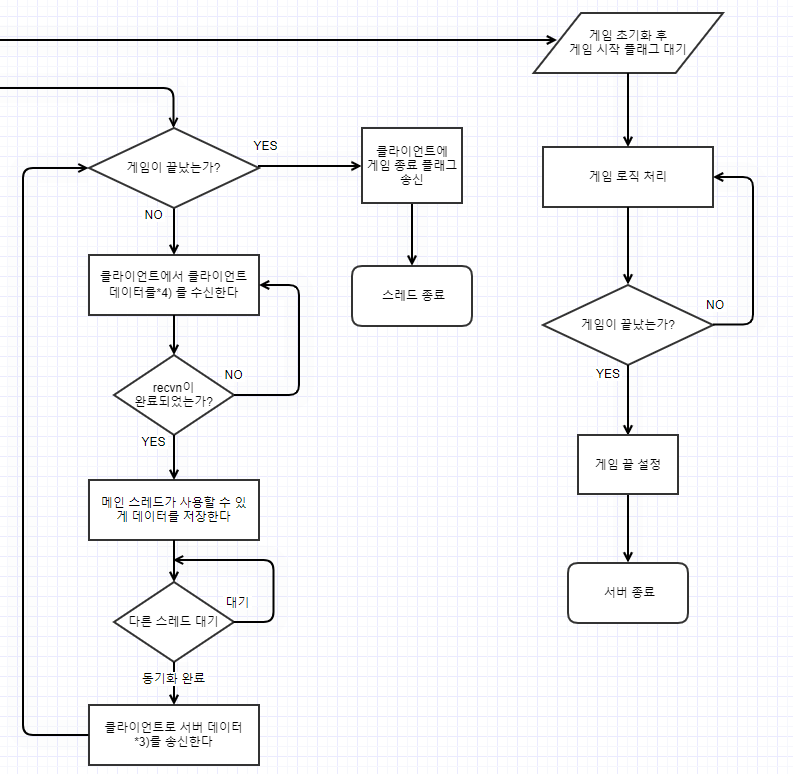


인게임 플로우 차트

서버



클라이언트 접속 –> 캐릭터 선택 창 플로우 차트



인게임 플로우 차트

**3. Low-Level 디자인**

-통신 프로토콜;

\*1)

int start; //맨 처음 서버가 클라이언트에게 게임이 시작되었다고 보내는 신호

\*3)

**struct ClientToServer**

{

Int PlayerNum //플레이어 번호

int x //플레이어의 x좌표

int y //플레이어의 y좌표

int DrawState // \*2일때는 characterNum, 플레이어 상태

~~bool attack~~

char AttackedPlayerNum[3] //맞은 플레이어의 번호

// \*2일때는 ready, 다른 캐릭터에게 공격을 적중시켰는지 확인하는 플래그

}

//인 게임에서 클라이언트가 서버에 보내는 데이터

//동기 TCP 사용, bool 사용하므로 #pragma pack 사용해야 함

\*4)

**struct ServerToClient**

{

~~int EnemyX //상대의 x좌표~~

~~int EnemyY //상대의 y좌표~~

~~int EnemyDrawState // \*2일때는 characterNum, 상대의 상태~~

~~bool attacked // \*2일때는 ready, 피격 당했는지 알려주는 플래그~~

ClientToServer PlayerData[3] // 플레이어 데이터

int CoinX //현재 코인의 X좌표

int CoinY //현재 코인의 Y좌표

int CoinState //코인의 상태 (누가 갖고 있는지)

int Time //남은 시간

}

//인 게임에서 서버가 클라이언트에 보내는 데이터

//동기 TCP 사용

-클라이언트 네트워크 클래스;

**class NetworkManager**

{

NetworkManager () //소켓 생성

NetworkManager () //소켓 정리

void connect () //네트워크 연결

void sendData (const ClientToServer)) //게임 로직 처리부분에서 넘겨주는 데이터

ClientToServer CreateClientToServer() //player데이터를 전송할 프로토콜로 만들기

void send(ClientToServer&) //전송

void recvData (int Scene, &ServerToClient)) //게임 로직 처리 부분으로 넘길 데이터

void recvn(ServerToClient\*) //데이터를 받기

//return 값 ServerToClient

void setData(ServerToClient&) //시간 플레이어 전체 코인

private:

SOCKET socket; //자신의 소켓

Player\*\* players // 플레이어 배열 포인터

Int playerNum // 플레이어 번호

Int\* time // 시간 포인터

Coin\* //코인 포인터

}

-서버 네트워크 클래스;

**class PlayerNetworkManager**

{

PlayerNetworkManager() //소켓 생성

~PlayerNetworkManager() //소켓 정리

void setSocket(SOCKET&) //소켓 지정, recvData 쓰레드 생성

static void recvData () // 현재 클라이언트로부터 데이터 받기 (<-socket)

static DWORD WINAPI recvData (LPVOID) // 쓰레드 함수

// 현재 클라이언트로부터 데이터 받기 (<-socket)

void recvn(ClientToServer\*) //데이터를 받기

void setData(CientToServer&) //ClientToServer 등록

void sendData (const ServerToClient&) // 현재 클라이언트로 데이터 보내기

private:

SOCKET socket; //현재 클래스가 가리키는 플레이어의 소켓

SOCKET othersocket; //현재 클래스가 가리키지 않는 플레이어의 소켓

ClientToServer Data; //받은 플레이어 데이터 저장

}

**class MainStream** //서버 게임 로직 처리 클래스

{

MainStream() //게임 초기화

~MainStream() //게임 끝

~~bool initiate() //두 클라이언트 접속 확인하는 부분~~

void waitForClientToConnect() //3개의 클라이언트 listen, accept

//접속하면 PlayernetworkManager에 socket 생성

//3명 다 접속 완료하면 sendData() 쓰레드 생성

void playerSelectStart() // 캐릭터 선택 창으로 넘어가기

void gameLogic() //인게임 로직 함수

void TimeElapse() // 시간 계산

void coinUpdate() // 코인 계산

static DWORD WINAPI sendData (LPVOID arg) // 쓰레드 함수

private:

~~int coinX; //현재 코인의 X좌표~~

~~int coinY; //현재 코인의 Y좌표~~

~~int coinState; //현재 코인의 상태~~

~~int time; //남은 시간~~

~~PlayerNetworkManager p[2]; //두 플레이어를 가리키는 클래스 배열~~

Socket listenSock

ServerToClient Data //플레이어, 시간, 코인 데이터

PlayerNetworkManager p[3]; //플레이어를 가리키는 클래스 배열

}

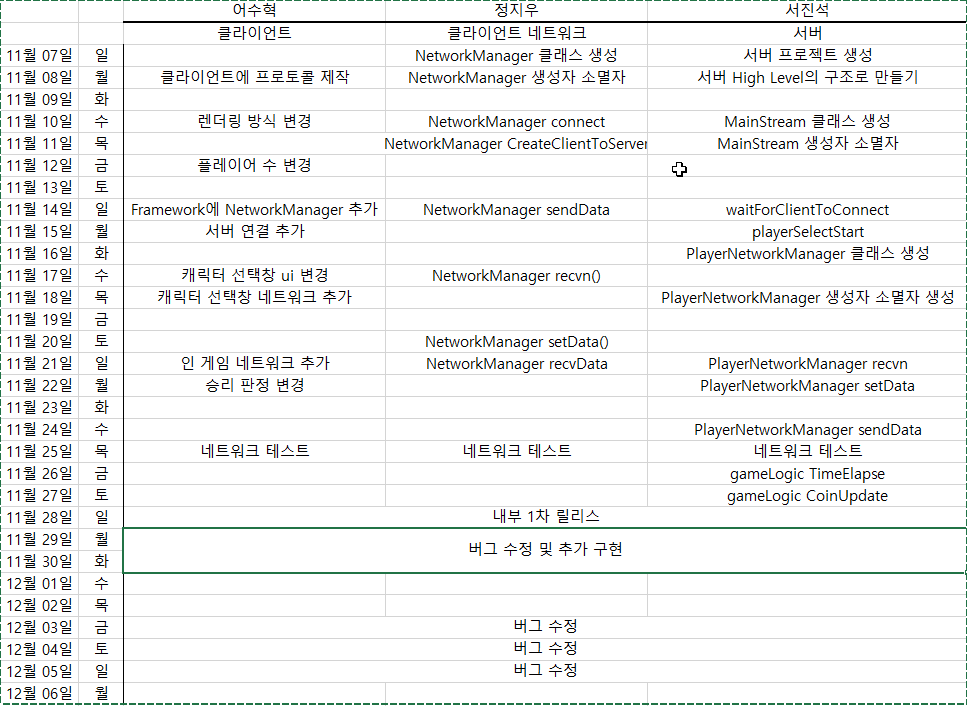
**4. 팀원 역할 분담**

* 정지우 : 어플리케이션 프로토콜 작성, 서버 로직 제작
* 서진석 : 서버 사이드 네트워크 클래스 제작
* 어수혁 : 클라이언트 사이드 네트워크 클래스 제작, 최적화

**5. 개발 환경**

* Visual Studio 2019
* Windows SDK 10.0.19041.0
* GitHub

**6. 개발 일정**



별 1)개발 일정표.xlsx