네트워크 게임 프로그래밍

추진 계획서

Term Project - Agr.io

2017180004 고선민

2017180004 남주영

2017180004 김재원

**Agr.io**

목차

**애플리케이션 기획**

게임 소개2

게임 플레이3

조작 방법3

**High-Level 디자인**

서버 구조5

클라이언트 구조6

**Low-Level 디자인**

장 제목 입력(수준 2)5

장 제목 입력(수준 2)6

**개발 환경**10

**팀원 별 역할분담**10

**High-Level 디자인**10

**애플리케이션 기획**

* **게임 소개**

****

|  |  |
| --- | --- |
| **Agr.io** | |
| **장르** | 서바이벌 슈팅 게임 |
| **최대 인원** | 4명 |
| **플레이 시간** | 5분~8분 |
| **조작** | 키보드, 마우스 |
| **시점** | 탑뷰, 2D |
| **개발언어** | C++, 윈도우API |
| **플랫폼** | 윈도우 |

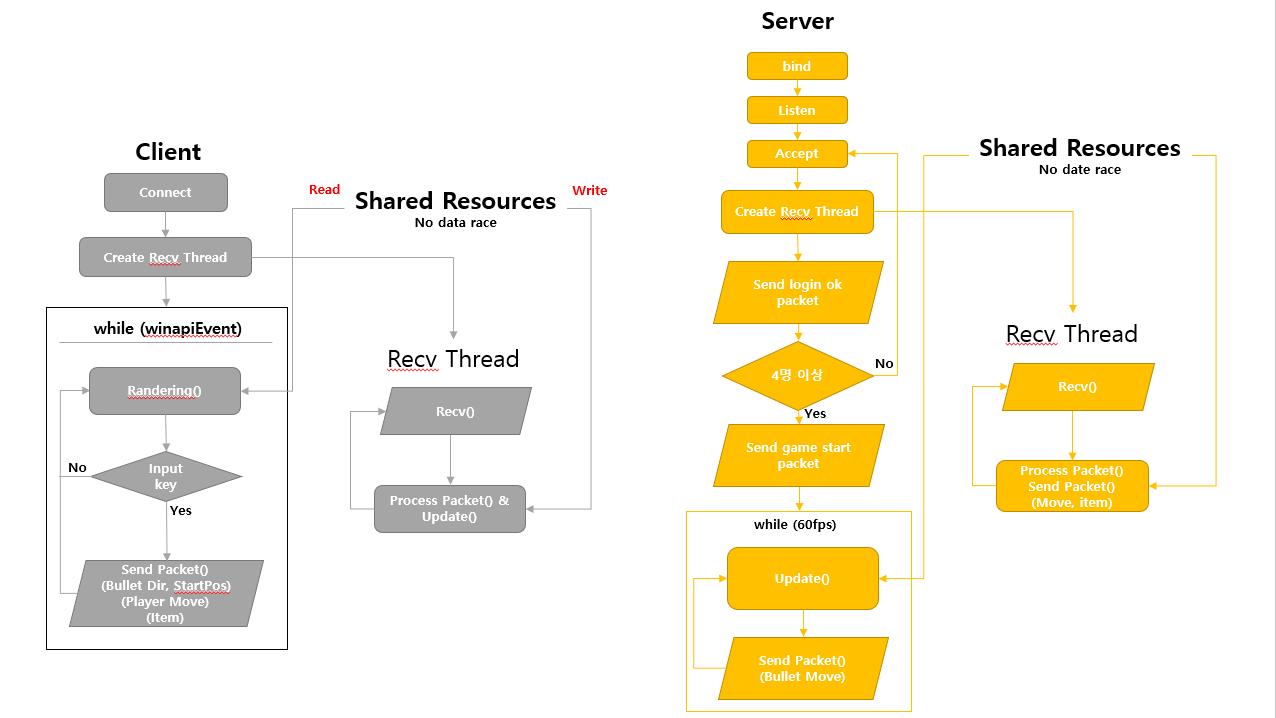
Agr.io는 PVP 서바이벌 슈팅 게임입니다.

지도이(가) 표시된 사진

자동 생성된 설명

* **게임 플레이**
* **조작 방법**

**High Level Design**

****

**Low Level Design**

1. **Protocol**

* **서버 주소**

const short SEVER\_PORT = 4000;

const char SEVER\_ADDR = “127.0.0.1”;

* **최대 접속 가능 클라이언트**

const char MAX\_USER = 4;

* **플레이어 상태**

const enum PLAYERSTATE {

IDLE, RUN, SHOOT, HIT

}

* **총알 발사 방향**

const enum DIR {

UP, DOWN, LEFTUP, RIGHTUP, LEFTDOWN, RIGHTDOWN, LEFT, RIGHT

}

* **아이템 ID**

const enum ITEM {

pistol, uzi, shotgun, hpPotion, wall

}

* **오브젝트 ID**
* **패킷 타입 (Client -> Server)**

const char CS\_PACKET\_LOGIN = 1;

const char CS\_PACKET\_PLAYER\_MOVE = 2;

Const char CS\_PACKET\_PALYER\_STATE = 3;

const char CS\_PACKET\_SHOOT\_BULLET = 4;

const char CS\_PACKET\_USED\_ITEM = 5;

* **패킷 타입 (Server -> Client)**

const char SC\_PACKET\_LOGIN\_OK = 1;

const char SC\_PACKET\_CHANGE\_SCENE = 2;

const char SC\_PACKET\_OBJ\_MOVE = 3;

const char SC\_PACHET\_PLAYER\_STATE = 4;

const char SC\_PACKET\_PUT\_OBJ = 5;

const char SC\_PACKET\_REMOVE\_OBJ = 6;

const char SC\_PACKET\_CHANGE\_HP = 7;

const char SC\_PACKET\_GET\_ITEM = 8;

const char SC\_PACKET\_CHAGE\_WEAPON = 9;

* **패킷 정의(Client -> Server)**

struct cs\_packet\_login{

unsigned char packetSize;

char packetType;

char playerSkin;

}

struct cs\_packet\_move{

unsigned char packetSize;

char packetType;

char x, y;

char dir;

}

struct cs\_packet\_player\_state

unsigned char packetSize;

char packetType;

char playerState;

}

struct cs\_packet\_shoot\_bullet{

unsigned char packetSize;

char packetType;

char shootX, shootY;

char dir;

}

struct cs\_packet\_used\_item {

unsigned char packetSize;

char packetType;

char itemNum;

}

* **패킷 정의(Server -> Client)**

struct sc\_packet\_login\_ok{

unsigned char size;

char packetType;

char playerID;

char x, y;

}

struct sc\_packet\_change\_scene{

unsigned char size;

char packetType;

char sceneNum;

}

struct sc\_packet\_obj\_move{

unsigned char size;

char packetType;

char objectID;

char lookDir;

char x, y;

}

struct sc\_packet\_player\_state{

unsigned char size;

char packetType;

char objectID;

char playerState;

}

struct sc\_packet\_put\_obj {

unsigned char size;

char packetType;

char objectID;

char sprite;

char x, y;

unsigned char width, height;

}

struct sc\_packet\_remove\_obj {

unsigned char size;

char packetType;

char objectID;

}

struct sc\_packet\_change\_hp {

unsigned char size;

char packetType;

char playerID;

unsigned char hp;

}

struct sc\_packet\_get\_item {

unsigned char size;

char packetType;

… char playerID;

char itemID;

}

struct sc\_packet\_change\_weapon {

unsigned char size;

char packetType;

… char playerID;

char gunID; // == itemID

}

1. **Client Method**

* **ProcessPacket()**

**전송된 데이터의 크기 > 패킷 사이즈**

패킷이 여러 개 온 경우이기 때문에 패킷을 처리한 후 남은 데이터 크기와 다시 비교해본다.

**전송된 데이터의 크기 < 패킷 사이즈**

패킷이 전송이 덜 된 것이기 때문에 버퍼에 남겨두고 그 뒤 버퍼 공간에 다시 recv한다.

**전송된 데이터의 크기 == 패킷 사이즈**

패킷을 처리한다.

**패킷 처리**

패킷에 있는 packetType에 따라 함수를 호출해 준다.

* **LoginOk()**

해당 번호의 플레이어를 활성상태로 바꾼다.

* **ObjectMove()**

오브젝트 ID를 인덱스로 사용하는 배열에 있는 오브젝트의 위치를 바꾼다.

* **PutObject()**

오브젝트 ID를 인덱스로 사용하는 배열에 있는 오브젝트의 활성 상태로 바꾼다.

* **RemoveObject()**

오브젝트 ID를 인덱스로 사용하는 배열에 있는 오브젝트를 비활성 상태로 바꾼다.

* **ChangeScene()**

SceneNum에 해당 되는 씬으로 바꾼다.

* **ChangePlayerState()**

플레이어의 스프라이트를 상태에 맞춰 바꾼다.

* **ChangeHP()**

플레이어의 체력을 변경시킨다.

* **SetItem()**

플레이어의 아이템 개수를 변경시킨다.

* **PlayerChangeWeapon()**

플레이어의 무기 스프라이트를 변경한다.

1. **Server**

**Global Variable**

* **Array<GameObject> GameObjects;**
* **Array<client> clients; // 소켓, 버퍼**

**Method**

**패킷 수신&처리**

* **RecvPacket()**
* **ProcessPacket()**
* **LoginPlayer(), InitPlayer()**

**로그인, 클라이언트 정보 수신, Player 생성**

* **MovePlayer()**

**Move 패킷을 받은 뒤 실행, 좌표 최신화, 이후 충돌 체크 실행**

* **BroadcastOtherState()**

**클라이언트의 State 패킷을 받은 뒤 실행, 받은 상태를 다른 클라이언트에 전달**

* **CreateBullet()**

**총 종류,위치,방향,속도를 토대로 총알 활성화, 이후 오브젝트**

* **UseItem()**

**Item 패킷을 받은 뒤 실행, 사용한 아이템에 따라 Array의 상태를 바꾼다.**

**충돌 체크**

* **CheckCollision\_PnP()**

**플레이어끼리 충돌검사**

* **CheckCollision\_PnO()**

**플레이어와 오브젝트 사이의 충돌검사, 충돌 시 데미지에 따라**

**CheckCollision\_OnO()**

**오브젝트끼리 충돌검사, 총알과 다른 오브젝트 충돌 시 SC\_PACKET\_REMOVE\_OBJ 패킷 송신**

**패킷 송신(Send는 패킷의 종류만큼 많아 지나?)**

* **SendToObjectsData(Object\* objects)**

**충돌체크는 매번 플레이어가 움직일 때, 총알이 움직이고 있는 동안 작동해야 한다.**

**총알이 날라가는 중엔 서버가 매번 총알이 1프레임씩 움직일 때마다 보내나?**

**패킷이 조금 과하게 나눠진 것 같은데**

**Coding Convention**

**Class - 첫번째 문자가 대문자(파스칼 표기법)**

**Method - 첫번째 문자가 대문자(파스칼 표기법), 동사+명사**

**Value & Parameter - 첫번째 문자가 소문자(카멜 표기법)**