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**

코딩 규칙5

프로토콜6

클라이언트6

서버6

**개발 환경**10

**팀원 별 역할분담**10

**High-Level 디자인**10

**애플리케이션 기획**

* **게임 소개**

****

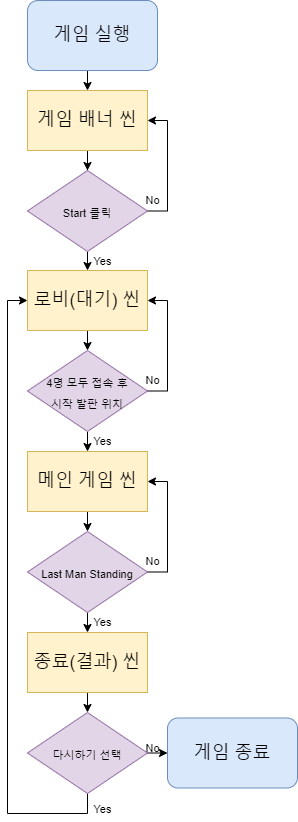
|  |  |
| --- | --- |
| **Agr.io** | |
| **장르** | 서바이벌 슈팅 게임 |
| **최대 인원** | 4명 |
| **플레이 시간** | 5분~8분 |
| **조작** | 키보드, 마우스 |
| **시점** | 탑뷰, 2D |
| **개발언어** | C++, 윈도우API |
| **플랫폼** | 윈도우 |

Agr.io는 PVP 서바이벌 슈팅 게임입니다.

지도이(가) 표시된 사진

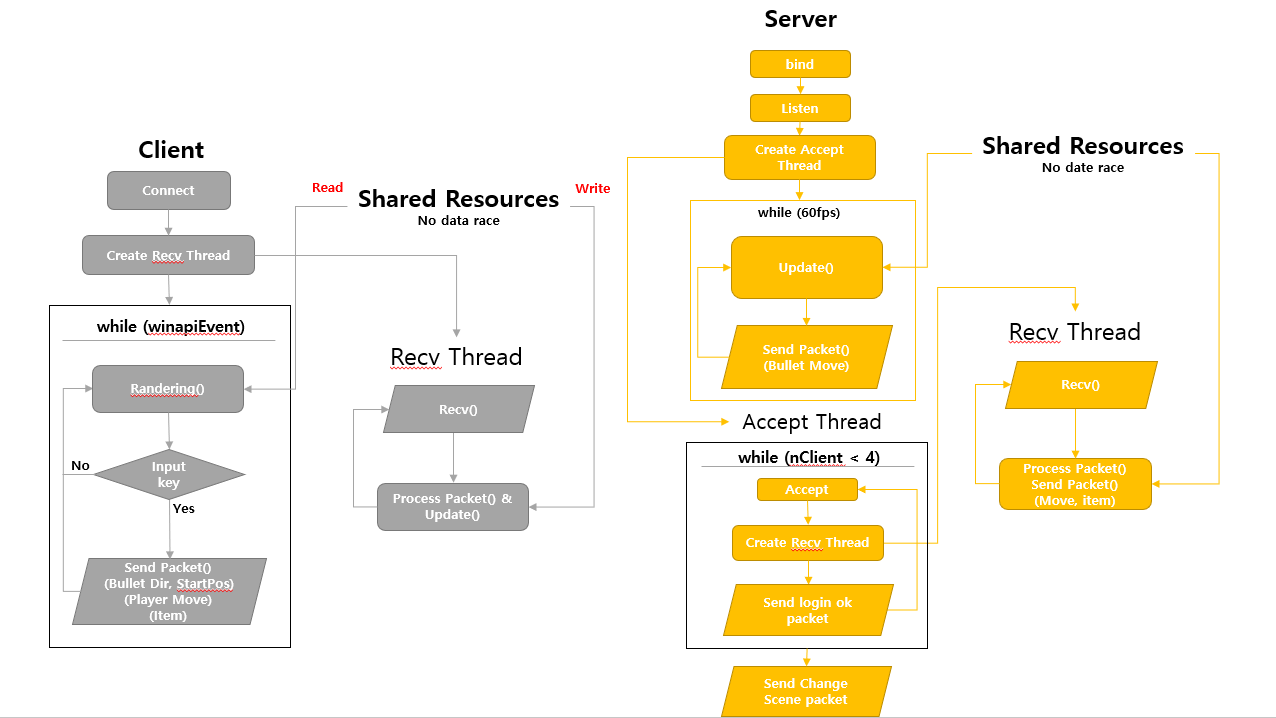
자동 생성된 설명

* **게임 플레이**

****

* **조작 방법**

**High Level Design**

****

**Low Level Design**

**1. 코딩 규칙 (Coding Convention)**

**Class -** 첫번째 문자가 대문자(파스칼 표기법)

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

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

**2. 프로토콜 (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

}

**3. 클라이언트 (Client)**

**Class**

**테이블이(가) 표시된 사진

자동 생성된 설명**

**Variable**

* **vector<GameObject> gameObjects;**
* **vector<CImage> sprites;**

**Method**

* **DWORD WINAPI Recv(LPVOID sock)**

recv를 호출해 packet이 오기를 기다리다 패킷이 오면 앞 부분(2바이트) packetSize, packetType을 확인한 후 packetSize – 2 만큼 다시 recv를 받고 packetType에 맞는 행동을 합니다.

*Ex) packType이 SC\_PAKCET\_PUT\_OBJ이면 ObjectID 위치에 있는 오브젝트에 SetActive(), SetPosition(), SetSprite(), SetSize()를 호출해 렌더링 될 수 있도록 한다.*

* **void Rendering()**

gameObjecs의 render를 호출합니다.

* **Send Packet**

입력된 키에 따라 다른 패킷을 서버로 전송합니다.

* **void GameObject::SetSprite (char spriteNum)**

Sprite의 값을 spriteNum으로 변경합니다.

* **void GameObject::SetSize (char w, char h)**

width, height의 값을 w, h로 변경합니다.

* **void GameObject::SetPosition(const Coordinate& pos)**

Pos의 값을 pos로 변경합니다.

* **void GameObject::SetActive(bool flag)**

오브젝트를 활성|비활성 상태로 변경합니다.

* **Player::SetItemNum(int index, short num)**

인덱스로 items배열에 접근해 배열 값(아이템 개수)을 수정합니다

* **void GameObject::Render()**

해당 오브젝트가 활성 상태라면 sprite,curGun를 인덱스로 리소스 배열에 접근해 해당 리소스를 그립니다.

* **Player::UseItem(int index)**

해당 인덱스의 아이템의 개수를 1개 줄이고 해당 되는 행동을 합니다.

* **Player::ChangeGun(char GunID)**

curGun의 값을 GunID로 변경합니다.

* **Player::ChangeState(char state)// state 왜 있는 거지 sprite로 하면 될 거 같은데**
* **Player::ChangeHp(short hp)**

hp값을 변경합니다.

**4. 서버 (Server)**

**Variable**

* **vector<GameObject> GameObjects;**
* **vector <client> clients;**

**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?)**

* **Send (Object\* objects)**