네트워크 게임 프로그래밍

추진 계획서

Term Project - Agr.io

2017180001 고선민

2017180009 남주영

2017180004 김재원

**Agr.io**

**목차**

**애플리케이션 기획**

1. 게임 소개3

2. 게임 플레이4

3. 조작 방법5

**High-Level 디자인**5

**Low-Level 디자인**

1. 코딩 규칙6

2. 프로토콜6

3. 클라이언트10

4. 서버13

**팀원 별 역할분담**15

**개발 환경**16

**개발 일정**17

**애플리케이션 기획**

**1. 게임 소개**

|  |  |
| --- | --- |
| **Agr.io** | |
| **장르** | 서바이벌 슈팅 게임 |
| **최대 인원** | 3명 |
| **플레이 시간** | 5분~8분 |
| **조작** | 키보드, 마우스 |
| **시점** | 탑뷰, 2D |
| **개발언어** | C++, 윈도우API |
| **플랫폼** | 윈도우 |

Agr.io는 서바이벌 슈팅 멀티플레이 게임입니다.

3명의 플레이어는 맵에서 최후의 1인이 남을 때까지 싸워서 살아남는 게임입니다.

****

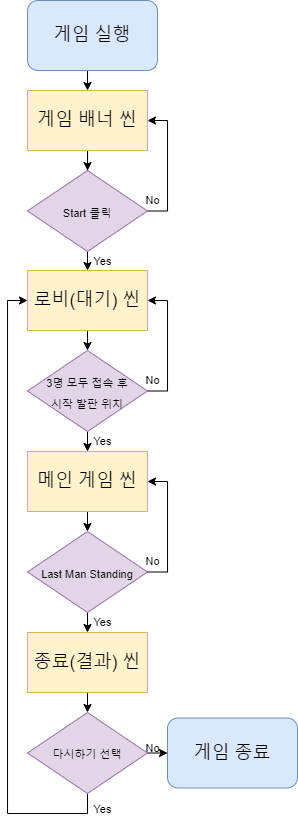
지도이(가) 표시된 사진

자동 생성된 설명

위 스크린샷(좌)은 개발중인 클라이언트로 추후 PVP와 다양한 장애물, 좁아지는 맵 등을 추가할 예정입니다.

게임의 기본적인 구조는 모바일 게임 브롤스타즈(우측 스크린샷)와 유사합니다.

**2. 게임 플레이**

****

- 게임을 실행하고 배너를 거처

Start 버튼을 눌러 로비로

입장합니다.

- 로비에선 자유롭게 돌아다닐 수

있습니다.

- 3명 모두 로비 특정영역에 있는 시작

발판위로 이동한다면 게임이 시작

됩니다.

- 메인 게임 시작 시 정해진 위치에서

시작합니다.

- 플레이어들끼리 총을 쏘아 맞춰 상대방의

HP를 0이하로 만듭니다.

- 플레이어가 한 명씩 탈락할 때 마다

플레이할 수 있는 맵이 작아집니다.

- 게임 중간중간 사용할 수 있는 아이템

(총, 회복 물약, 박스)이 맵 어딘가 스폰

됩니다.

- 최후의 한 명이 남으면 메인 게임을

중단하고 결과창을 보여줍니다.

- 다시하기를 선택하면 로비로 돌아가 대기

합니다.

- 게임종료를 선택하면 그대로 종료합니다.

**3. 조작 방법**

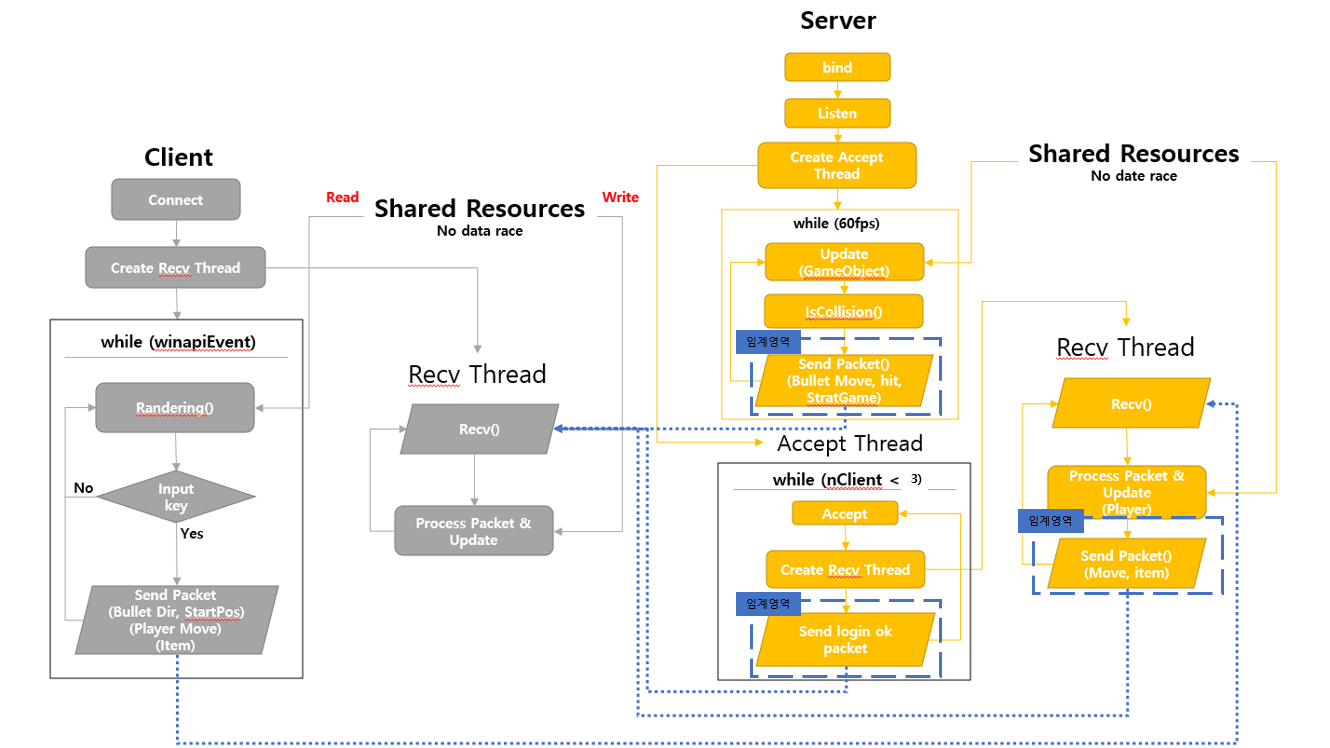
**사격 - 스페이스바**

**이동 - 방향키**

**아이템 사용 - 숫자 1~5**

**클릭 - 마우스**

**High Level 디자인**

****

* 플레이어의 접속을 계속 기다리면서 먼저 접속한 플레이어들끼리 로비에서 움직이며 대기할 수 있도록 Accept 스레드를 사용합니다.
* 첫번째 로그인시 0번 클라이언트가 Create RecvThread 생성이후에 Login패킷을 보내면 서버가 일을 시작하고 이후 접속하는 클라이언트들은 PutObj로 생성됩니다.
* 파란색 점선은 기본적인 데이터의 흐름입니다.

**Low Level 디자인**

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)

**전역변수** - 모든 문자가 대문자

1. **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 = 3;

* **총알 발사 방향**

enum DIR { UP, DOWN, LEFTUP, RIGHTUP, LEFTDOWN, RIGHTDOWN, LEFT, RIGHT }

* **아이템 ID**

enum ITEM { pistol, uzi, shotgun, potion, box }

* **패킷 타입 (Client -> Server)**

const char CS\_PACKET\_LOGIN = 1;

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

const char CS\_PACKET\_PLAYER\_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\_PACKET\_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\_ITEM\_COUNT = 9;

const char SC\_PACKET\_CHAGE\_WEAPON = 10;

* **패킷 정의(Client -> Server)**

struct packet{

unsigned char packetSize;

char packetType;

}

struct cs\_packet\_login : packet{

char playerSkin;

}

struct cs\_packet\_player\_move: packet{

char dir;

}

struct cs\_packet\_player\_state: packet{

char playerState;

}

struct cs\_packet\_shoot\_bullet: packet{

char shootX, shootY;

char dir;

}

struct cs\_packet\_used\_item: packet{

char itemNum;

}

* **패킷 정의(Server -> Client)**

struct sc\_packet\_login\_ok: packet{

char playerID;

char x, y;

}

struct sc\_packet\_change\_scene: packet{

char sceneNum;

}

struct sc\_packet\_obj\_move: packet{

char objectID;

char lookDir;

char x, y;

}

struct sc\_packet\_player\_state: packet{

char objectID;

char playerState;

}

struct sc\_packet\_put\_obj : packet{

char objectID;

char sprite;

char x, y;

unsigned char width, height;

}

struct sc\_packet\_remove\_obj : packet{

char objectID;

}

struct sc\_packet\_change\_hp : packet{

char playerID;

unsigned char hp;

}

struct sc\_packet\_get\_item : packet{

char playerID;

char itemID;

}

struct sc\_packet\_item\_count : packet{

char playerID;

char itemID;

char itemCount;

}

struct sc\_packet\_change\_weapon : packet{

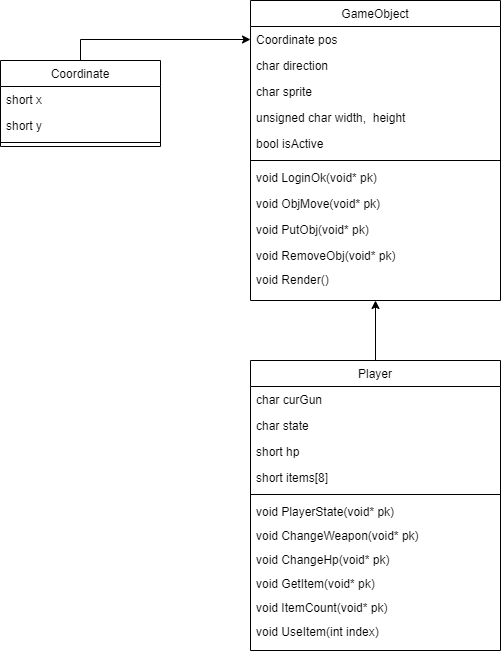
char playerID;

char gunID; // == itemID

}

1. **클라이언트 (Client)**

**Class**

****

**Variable**

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

**Method**

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

recv를 호출해 packet이 오기를 기다리다 패킷이 오면 앞부분(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()**

gameObjects의 render를 호출합니다.

* **Send Packet**

입력된 키에 따라 다른 패킷을 서버로 전송합니다.

* **Void GameObject::LoginOk (void\* pk)**

LOGIN\_OK 패킷이 오면 호출할 함수입니다. 해당 플레이어 오브젝트를 활성 상태로 바꾸고 시작 위치를 정해줍니다.

* **Void GameObject::ObjMove (void\* pk)**

OBJ\_MOVE 패킷이 오면 호출할 함수입니다. 해당 오브젝트의 lookDir, x, y를 수정해줍니다.

* **Void GameObject::PlayerState (void\* pk)**

PLAYER\_STATE 패킷이 오면 호출할 함수입니다. 해당 플레이어의 상태 (이동 중, 공격 중)를 바꿔주어서 적절한 sprite를 사용할 수 있게 해줍니다.

* **Void GameObject::PutObj (void\* pk)**

PUT\_OBJ 패킷이 오면 호출할 함수입니다. 해당 오브젝트를 활성화하고 sprite, width, height, x, y를 정해줍니다.

* **Void GameObject::RemoveObj (void\* pk)**

REMOVE\_OBJ 패킷이 오면 호출할 함수입니다. 해당 오브젝트를 비활성화 상태로 바꾸어서 화면에 출력하지 않도록 합니다.

* **Void GameObject::Render()**

해당 오브젝트를 화면에 출력합니다.

* **Void Player::ChangeHp(void\* pk)**

CHANGE\_HP 패킷이 오면 호출할 함수입니다. 해당 플레이어의 hp값을 수정합니다.

* **Void Player::GetItem(void\* pk)**

GET\_ITEM 패킷이 오면 호출할 함수입니다. 플레이어의 인벤토리에서 아이템을 활성화해서 사용할 수 있도록 합니다.

* **Void Player::ItemCount(void\* pk)**

ITEM\_COUNT 패킷이 오면 호출할 함수입니다. 플레이어가 가지고 있는 인벤토리 내의 아이템 보유 수량을 수정해줍니다.

* **Void Player::ChangeWeapon(void\* pk)**

CHANGE\_WEAPON 패킷이 오면 호출할 함수입니다. 해당 플레이어가 들고 있는 무기를 바꿔줍니다.

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

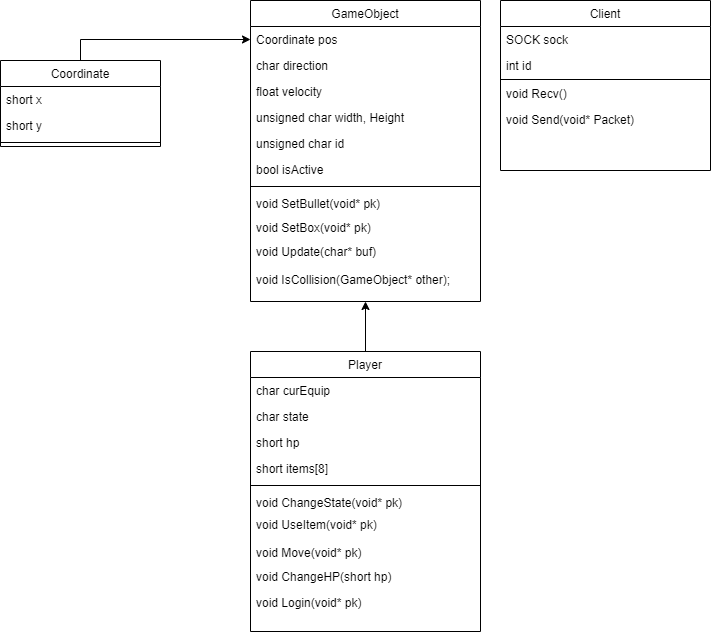
플레이어가 들고 있는 총을 발사 혹은 아이템을 사용합니다. 들고 있는 아이템이 총일 경우 cs\_packet\_shoot\_bullet을, 그 외의 아이템은 cs\_packet\_used\_item 패킷을 구성해서 서버에 Send하여 줍니다.

* **Void ChangeScene(void\* pk)**

CHANGE\_SCENE 패킷이 오면 호출할 함수입니다. 게임을 해당 장면으로 전환합니다.

1. **서버 (Server)**

**Class**

****

**Variable**

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

**Method**

* **void GameObject::SetBullet(void \*pk)**

SHOOT\_BULLET 패킷이 오면 호출할 함수입니다. 비활성 상태인 오브젝트 중 하나를 활성 상태로 바꾸고 시작 위치 속도 방향을 정한 뒤 sc\_packet\_put\_obj 패킷을 만들어 모든 클라이언트에게 Send하여 줍니다. 이때 클라이언트가 사격키를 연타하여도 최대 공격속도를 넘지 않도록 제한하는 역할도 합니다.

* **void GameObject::SetBox(void \*pk)**

USED\_ITEM 패킷이 오면 호출할 함수입니다. 비활성 상태인 오브젝트 중 하나를 활성 상태로 바꾸고 위치 방향을 정한 뒤 sc\_packet\_put\_obj 패킷을 만들어 모든 클라이언트에게 Send하여 줍니다.

* **void GameObject::Update(~~char\* buf~~ > char\* buf, int& bufStart)**

활성 상태이면 자신이 가지고 있는 속도와 방향을 이용해 위치를 갱신한 후 해당 정보를 패킷으로 만들어 패킷을 buf에 복사합니다.

* **void Player::ChangeState(void\* pk)**

PLAYER\_STATE 패킷에 대한 호출 함수입니다. Player.state를 변경한 뒤 sc\_packet\_player\_state 패킷을 만들어 모든 클라이언트에게 Send합니다.

* **void Player::UseItem(void\* pk)**

USED\_ITEM 패킷에 대한 호출 함수입니다. 패킷에서 아이템에 대한 정보를 받아 회복 물약이면 sc\_packet\_change\_hp 패킷을 만들어 모든 클라이언트에게 Send | 박스 아이템이면 sc\_packet\_put\_obj 패킷을 만들어 모든 클라이언트에게 Send | 총이면 총 상태를 바꿔주고 sc\_packet\_change\_weapon 패킷을 만들어 모든 클라이언트에게 Send

그 뒤 sc\_packet\_item\_count 패킷을 만들어 사용한 클라이언트에게 Send합니다.

* **void Player::Move(void\* pk)**

PLAYER\_MOVE 패킷이 오면 방향을 갱신해 주고 최대 이동속도에 맞춰 좌표를 모든 클라이언트에게 Send합니다.

* **void Player::ChangeHP(short hp)**

CHANGE\_HP 패킷이 오면 해당 플레이어의 체력을 바꾸고 sc\_packet\_change\_hp 패킷을 만들어 모든 클라이언트에게 Send합니다.

* **void Player::Login(void\* pk)**

LOGIN 패킷이 오면 0~2번 중 비활성 상태인 플레이어를 활성상태로 만들고 sc\_packet\_login\_ok 패킷을 만들어 해당 클라이언트에게 보냅니다.

sc\_packet\_put\_obj 패킷을 만들어 해당 클라이언트에게 맵의 현재 오브젝트 상태를Send합니다.

sc\_packet\_put\_obj 패킷을 만들어 다른 클라이언트에게 새로운 플레이어가 생성되었다고 알려줍니다.

* **void GameObject::IsCollision (GameObject\* objects)**

플레이어끼리 충돌, 오브젝트끼리 충돌, 총알과 다른 오브젝트 충돌 시 SC\_PACKET\_REMOVE\_OBJ 패킷을 송신합니다.

* **void Client::Send (void\* Packet)**

패킷 포인터를 받아서 char\*로 변환 후 Send 합니다.

**팀원 별 역할분담**

**Client - Login 부분 고선민**

**Client - Rendering 부분 고선민**

**Client - Send 부분 고선민**

**Client - Recv 부분 고선민**

**Client - Input Key 부분 고선민**

**Client - Process Packet 부분 김재원**

**Sever – Send 부분 김재원**

**Sever - Process Packet 부분 김재원**

**Sever - Update 부분 김재원**

**Sever - Lobby 부분 김재원**

**Sever - Login 부분 남주영**

**Sever - Recv Thread 부분 남주영**

**Sever - Accept Thread 부분 남주영**

**Sever - IsCollision 부분 남주영**

**문서 작성 김재원**

**일정 관리 고선민**

**버전 관리 남주영**

**리소스 고선민**

**개발환경**

**운영체제**

* Windows 10

**개발 도구**

* Visual Studio 2019
* 포토샵CS6
* draw.io
* Word

**VCS**

* Github

**커뮤니케이션**

* Discord
* KakaoTalk

**사용 언어, 라이브러리**

* **C++**
* WinAPI

**개발일정**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 월 | 화 | 수 | 목 | 금 | 토 |
|  | 11월 1일 | 2일 | 3일 | 4일 | 5일 | 6일 |
| 선민 | 클라이언트 수정 | | 클라이언트 Bullet 구현 | | 클라이언트 Send(), Recv() 제작 | |
| 주영 | 프로토콜 작성 | 서버 Recv() 스레드 제작 | Updata()구현 | Accept()스레드 구현 |
| 재원 | 서버 Send() 제작 | 서버 ProcessPacket() 구현 | 서버Recv() 스레드 수정 |
|  | 8일 | 9일 | 10일 | 11일 | 12일 | 13일 |
| 선민 | Login() 구현 | ChangeHP()구현 | ObjMove() 구현 | Render() 수정 | | 1주간 미비 사항 추가, 수정 |
| 주영 | PutObj() 구현 | LoginOk() 구현 | Move() 구현 | SetBullet() 구현 | |
| 재원 | 로비 스테이지 구현 | | RemoveObj()구현 | ChangeState() 구현 | Update(), Send(), Recv() 수정 |
|  | 15일 | 16일 | 17일 | 18일 | 19일 | 20일 |
| 선민 | 클라이언트 오브젝트 추가 구현 | 오브젝트 활성화 | UseItem() 구현 | ChangeWeapon() 구현 | 여러 총기 구현 | 1주간 미비 사항 추가, 수정 |
| 주영 | Update(), Send(), Recv() 수정 | | SetBox() 구현 | 아이템 상호작용  디버그 | 아이템 보급 구현 |
| 재원 | SetBullet() 구현 | | 클라-서버 Bullet Timing 조정 | ItemCount() 구현 | Update() 수정 |
|  | 22일 | 23일 | 24일 | 25일 | 26일 | 27일 |
| 선민 | 게임 종료 구현 | 리더보드 구현 | 씬 변환 수정 | 충돌 후 클라이언트 Process 수정 | 클라이언트 최적화 | 기말고사  준비 |
| 주영 | GetItem() 구현 | IsCollision()구현 | | 충돌 후 서버 Process 구현 | 서버 최적화 |
| 재원 | 제한구역 구현 | | Stage 변화 구현 | ProcessPacket() 수정 | 서버 최적화 |
|  | 30일 | 12월 1일 | 2일 | 3일 | 4일 | 5일 |
| 선민 | 게임 씬 최적화 | 추가 기능 구현 | | 밸런스 조정 | 발표 자료 준비, 일정 진척도 종합 | |
| 주영 | 비정상적 네트워크상황 대응 서버 디버그 | | 추가 기능 구현 | | 최종 시연 준비, 깃 허브 버전 관리 | |
| 재원 | 비정상적 네트워크 상황대응 클라이언트 디버그 | | 추가 기능구현 | | 리포트 작성 | |