**네트워크 게임 프로그래밍**

**2D Battle Arena**

Project Progress Report

2016180037 임건호

2016182019 성기홍

**<목차>**

[1. 게임 기획 2](#_Toc58440315)

[A. 게임 컨셉 2](#_Toc58440316)

[B. 게임 플레이 2](#_Toc58440317)

[C. 게임 조작 2](#_Toc58440318)

[D. 게임에 필요한 요소 2](#_Toc58440319)

[2. 하이 레벨 2](#_Toc58440320)

[3. 로우 레벨 2](#_Toc58440321)

[< 클래스 > 2](#_Toc58440322)

[< 클라이언트 > 2](#_Toc58440323)

[< 서버 > 2](#_Toc58440324)

[4. 역할 분담 2](#_Toc58440325)

[5. 개발 환경 2](#_Toc58440326)

[6. 개발 일정 2](#_Toc58440327)

# 

# 

# 게임 기획

## 게임 컨셉

2D로 구현한 2인용 TPS 게임.



<surviv.io 인게임 사진>

## 게임 플레이

ㄱ. 타이틀 화면이 나온다.

ㄴ. 게임 시작을 누르면 게임 내에서 사용할 닉네임/색상을 적으라고 한다.

ㄷ. 확인을 누르면 게임을 하는데 필요한 인원수가 모일 동안 대기한다.

ㄹ. 인원이 모두 모이면 랜덤한 위치에 장애물과 플레이어 캐릭터가 생성된다.

ㅁ. 플레이어는 제한된 공간 안에서 적을 찾아 총을 발사한다.

ㅂ. 총알을 맞으면 HP가 감소하며, 모두 감소 시 패배한다.

ㅂ. 총알을 맞으면 HP가 감소하며, 0이 되면 목숨이 하나 줄어든다.

ㅅ. 목숨이 3개가 다 떨어지면 패배한다.

ㅇ. 적의 목숨을 전부 떨어뜨리면 승리한다.

## 게임 조작

ㄱ. 이동: W, A, S, D

ㄴ. 총 조준: 마우스

ㄷ. 발사: 왼쪽 클릭

ㄹ. 재장전: R

(자동 재장전)

## 게임에 필요한 요소

ㄱ. 타이틀 화면

ㄴ. 닉네임과 서버 주소 입력 창

ㄷ. 모든 인원이 모이면 자동 시작

ㄴ. 닉네임 입력 팝업창

ㄷ. 인원이 모일 동안 대기 인원을 보여주는 대기창

ㄹ. 게임을 플레이할 필드

ㅁ. 게임에 재미를 더할 랜덤 위치 장애물

ㅂ. 플레이어가 조종할 캐릭터

ㅅ. 게임플레이를 도울 UI

ㅇ. 공격을 위한 총알

ㅈ. 피격 시 맞았다는 충격

ㅈ. 피격했다는 이펙트

# 하이 레벨

# 로우 레벨

## < 클래스 >

* class GameObject : 게임에 사용되는 오브젝트들의 좌표, 사이즈, 속도, 종류 관리

{

float m\_PositionX, m\_PositionY, m\_PositionZ;

float m\_SizeX, m\_SizeY;

float m\_VelX, m\_VelY;

float m\_AccX, m\_AccY;

float m\_Mass;

}

* class PlayerCharacter : Public GameObject : 플레이어 캐릭터의 정보 관리

{

char p\_Name[10];

int p\_Color;

int p\_HP;

int p\_CurrentAmmo;

float p\_ReloadDelay;

float p\_ShotDelay;

}

* class Bullet: Public GameObject : 게임 내 총알 관리

{

int b\_Check; // 누가 쏜 총알인지.

bool check; // 사용된 총알인지

}

* class Obstacle: Public GameObject : 장애물 종류

{

int o\_Sort; // 장애물의 종류

}

* class Render : 화면 초기화 및 그림 그리기

{

void DrawSolidRect(float x, float y, float z, float size, float r, float g, float b, float a); // 오브젝트 그리기

unsigned int WindowSizeX, WindowSizeY; // 화면 크기

void Initialize(int windowSizeX, int windowSizeY); // 초기화

}

## < 클라이언트 >

1. 서버와의 연결

{

SOCKET sock = socket(AF\_INET, SOCK\_STREAM, 0);

SOCKADDR\_IN serveraddr;

ZeroMemory(&serveraddr, sizeof(serveraddr));

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_addr.s\_addr = inet\_addr(SERVERIP);

serveraddr.sin\_port = htons(SERVERPORT);

retval = connect(sock, (SOCKADDR\*)&serveraddr, sizeof(serveraddr));

}

1. 타이틀 화면

{

void Start\_Button(float x, float y, float z, float size);

void Exit\_Button(float x, float y, float z, float size);

TitleScreen::Update(float time, NGPInputs\* inputs);

}

1. 사용자 데이터 설정(닉네임, 캐릭터 색깔)/데이터 송신

{

char p\_Name[10];

Hero->SetName(namebuf);

send(sock, (char\*)Hero, sizeof(PlayerCharacter), 0);

int p\_Color;

PlayerCharacter::PlayerCharacter() // 랜덤 위치 캐릭터 생성

}

1. 게임 시작 전 데이터 수신

{

Obstacle Obstacles[MaxObstacles]

PlayerCharacter Players[MaxPlayer]

recvn(sock, (char\*)&ObstaclesParam, sizeof(Obstacle) \* MAX\_Obstacles, 0);

dynamic\_cast<MainGame\*>(CurrentScreen)->SetObstacle(ObstaclesParam);

recvn(sock, (char\*)&Enemy, sizeof(PlayerCharacter), 0);

}

1. 게임 종료

{

void PlayerWin()

void PlayerLose()

CurrentScreen = new ResultScreen(true/false);

}

1. 캐릭터 조작 : 키보드, 마우스 입력 처리

{

void MouseInput(int button, int state, int x, int y)

void KeyDownInput(unsigned char key, int x, int y)

void KeyUpInput(unsigned char key, int x, int y)

void DragInput(int x, int y)

}

1. 게임 배경

{

void BackGround(int windowSizeX, int windowSizeY)

DrawTextureRect()

}

1. 장애물

{

Obstacle Obstacles[MaxObstacles];

}

1. 총알

{

총알을 발사한 위치, 총알의 방향을 토대로 방향벡터 계산

Bullet Bullets[MaxBullet];

}

1. 충돌체크

{

클라이언트에서는 플레이어의 충돌을 처리

void IsPlayerObstacleCollsion(Player player, Obstacle obstacle)

void IsPlayerPlayerCollsion(Player player, Player player)

bool MainGame::AABBCollision(GameObject\* a, GameObject b)

void MainGame::AdjustPosition(GameObject\* a, GameObject b)

}

1. 게임 중 데이터 송/수신

{

캐릭터, 총알의 데이터를 서버로 전송

서버로부터 상대 캐릭터, 총알의 데이터를 수신

PlayerCharacter Players[MaxPlayer];

Bullet Bullets[MaxBullet];

Hero = dynamic\_cast<MainGame\*>(CurrentScreen)->GetHero();

dynamic\_cast<MainGame\*>(CurrentScreen)->GetBullet(HeroBullets);

send(sock, (char\*)Hero, sizeof(PlayerCharacter), 0);

send(sock, (char\*)HeroBullets, sizeof(Bullet) \* Max\_Bullet, 0);

recvn(sock, (char\*)&Enemy, sizeof(PlayerCharacter), 0);

recvn(sock, (char\*)&EnemyBullet, sizeof(Bullet) \* Max\_Bullet, 0);

dynamic\_cast<MainGame\*>(CurrentScreen)->SetEnemy(&Enemy);

dynamic\_cast<MainGame\*>(CurrentScreen)->SetEnemyBullet(EnemyBullet);

}

## < 서버 >

1. 서버 구축

{

SOCKET listen\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

SOCKADDR\_IN serveraddr;

ZeroMemory(&serveraddr, sizeof(serveraddr));

serveraddr.sin\_family = AF\_INET;

serveraddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

serveraddr.sin\_port = htons(SERVERPORT);

retval = bind(listen\_sock, (SOCKADDR \*)&serveraddr, sizeof(serveraddr));

retval = listen(listen\_sock, SOMAXCONN);

}

1. 클라이언트와의 연결

{

g\_Event = CreateEvent(NULL, FALSE, TRUE, NULL);

client\_sock = accept(listen\_sock, (SOCKADDR \*)&clientaddr, &addrlen);

hThread[MaxPlayer] = CreateThread(NULL, 0, ProcessClient, (LPVOID)client\_sock, 0, NULL);

}

1. 사용자 정보 수신, 저장

{

Char PlayerName[MaxPlayer][10]

int PlayerColor[MaxPlayer]

retval = recvn(client\_sock[PlayerNo], (char\*)&(PlayerInfo[PlayerNo]), sizeof(PlayerCharacter), 0);

retval = recvn(client\_sock[PlayerNo], (char\*)&(BulletInfo[PlayerNo]), sizeof(Bullet) \* Max\_Bullet, 0);

send(client\_sock[0], (char\*)&PlayerInfo[1], sizeof(PlayerCharacter), 0);

send(client\_sock[0], (char\*)&(BulletInfo[1]), sizeof(Bullet) \* Max\_Bullet, 0);

send(client\_sock[1], (char\*)&PlayerInfo[0], sizeof(PlayerCharacter), 0);

send(client\_sock[1], (char\*)&(BulletInfo[0]), sizeof(Bullet) \* Max\_Bullet, 0);

}

1. 랜덤 위치에 장애물, 캐릭터 생성

{

Obstacle MakeObstacle(float X, float Y, float SizeX, float SizeY)

Obstacle Obstacles[MaxObstacle]

PlayerCharacter MakePlayer()

PlayerCharacter Players[MaxPlayer]

}

1. 게임 시작 전 동기화

{

다른 접속자 정보, 장애물 위치, 캐릭터 위치 전송

PlayerCharacter Players[MaxPlayer]

Obstacle Obstacles[MaxObstacle]

WaitForMultipleObjects

}

1. 충돌 체크

{

서버에서는 총알의 충돌을 처리

void IsObstacleBulletCollsion(Obstacle obstacle, Bullet bullet)

void IsPlayerBulletCollsion(Player player, Bullet bullet)

}

1. 게임 중 데이터 송/수신

{

각 클라이언트로부터 캐릭터의 정보, 총알의 데이터 수신

각 클라이언트로 동기화 처리된 캐릭터의 정보, 총알의 데이터 전송

PlayerCharacter Players[MaxPlayer];

Bullet Bullets[MaxBullet];

}

1. 게임 종료

{

void PlayerWin()

void PlayerLose()

}

1. 데이터 동기화 처리

{

수신된 데이터와 기존 데이터의 차이, 수신 시간을 토대로 양 클라이언트의 데이터를 동기화함

Void SyncData(float time)

}

# 역할 분담

1. 임건호

* <클래스> Render
* <서버> 서버 구축
* <서버> 클라이언트와의 연결
* <클라이언트> 서버와의 연결
* <서버> 사용자 정보 수신, 저장
* <클라이언트> 게임 배경
* <클라이언트> 게임 시작 전 데이터 수신
* <클라이언트> 캐릭터 조작
* <클라이언트> 총알 데이터 송/수신
* <클라이언트> 게임중 데이터 송/수신
* <클라이언트> 캐릭터와 장애물 충돌체크
* <클라이언트> 캐릭터와 총알 충돌체크
* <클라이언트> 총알과 장애물 충돌체크
* <서버> 게임중 데이터 송/수신
* <서버> 총알 데이터 송/수신
* <서버> 충돌 체크
* <서버> 게임 종료
* <클라이언트> 타이틀 입력
* <클라이언트> 비트맵 입력

1. 성기홍

* <클래스> GameObject
* <서버> 사용자 정보 수신, 저장
* <클라이언트> 기본 게임 루프
* <클라이언트> 입력 함수
* <클라이언트> 게임 배경
* <클라이언트> 타이틀 화면
* <클라이언트> 결과 화면
* <클라이언트> 사용자 데이터 설정/송수신
* <서버> 랜덤 위치에 장애물, 캐릭터 생성
* <서버,클라이언트> 게임 시작 전 데이터 송/수신
* <클라이언트> 게임중 데이터 송/수신
* <서버> 게임중 데이터 송/수신
* <클라이언트> 비트맵 String, 텍스처 매핑
* <서버> 데이터 동기화 처리
* <클라이언트> 총알
* <클라이언트> 충돌 체크
* <클라이언트> 게임 종료

# 개발 환경

1. 구글 문서를 활용하여 기획서 작성
2. C++ 프로그래밍
3. GitLab을 활용한 프로그래밍 공유

<https://gitlab.com/hgbb3210/ngp_termproject>

1. OpenGL API 사용

# 개발 일정

1. **기존 일정**

**< 임건호 >**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 1 | 2 | 3 | 4 | 5 | 6  <서버>  bind()  listen()  <클라>  connect() | 7 |
| 8  <클래스>  Render | 9 | 10 | 11 | 12 | 13  <클라>  SetCharacterSetting()  bitmap mapping | 14 |
| 15  <서버>  send(Character,  Obstacles )  <클라>  recv(), WaitingPlayer() | 16 | 17 | 18 | 19 | 20  <클라>  IsPlayerObstacleCollsion()  CharacterMove() | 21 |
| 22  <서버>  recv()  (character, bullet)  SyncData() | 23 | 24 | 25 | 26 | 27  <클라>  recv()  (character, bullet)  DrawUI() | 28 |
| 29  <클라>  recv()  WinLose  PlayerWin()  PlayerLose() | 30 |  |  |  |  |  |

**< 성기홍 >**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7  <클래스>  GameObject |
| 8 | 9  <클라>  GameScreen Class  Bitmap  RenderScene  Update  입력 함수 | 10 | 11 | 12 | 13 | 14  <서버>  recv()  character  MakeObstacles  MakeCharacter |
| 15 | 16  <클라>  RenderScene()  PlayScreen | 17 | 18 | 19 | 20 | 21  <클라>  ShotBullet()  Send()  Bullet |
| 22 | 23  <서버>  BulletCollision()  Send()  (Character,  Bullet) | 24 | 25 | 26 | 27 | 28  <서버>  승패 체크  Send()  WinLose |
| 29 | 30  마무리 |  |  |  |  |  |

**2) 수정된 일정**

**< 임건호 >**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 1 | 2 | 3 | 4 | 5 | 6  <서버>  bind()  listen()  <클라>  connect() | 7 |
| 8  <클래스>  Render,  shader | 9 | 10 | 11 | 12 | 13  <클라>  character move, resource, screen | 14 |
| 15  <서버클라>  server connect()  send, recv(player) | 16 | 17 | 18 | 19 | 20  <클라>  object create(),  character move() | 21 |
| 22  <서버>  modify  // bullet, aabb, move, color | 23 | 24 | 25 | 26 | 27  <서버클라>  send, recv(bullet)  field setting  aabb // 총알 장애물 충돌 | 28 |
| 29  <클라>  playerCharacter()  // 랜덤위치  aabb()  //캐릭터 총알 충돌  <서버클라>  send, recv(playerHp) | 30 |  |  |  |  |  |

**< 성기홍 >**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7  <클래스>  GameObject |
| 8 | 9  <클라>  GameScreen Class  Bitmap  RenderScene  Update  입력 함수 | 10 | 11 | 12 | 13 | 14  <클라>  ShotBullet |
| 15 | 16  <클라/서버>  send/recv  PlayerCharacter | 17 | 18 | 19 | 20 | 21  <클라/서버>  MakeObstacle  send/recv  Obstacles  reload |
| 22 | 23  <클라/서버>  send/recv  EnemyData | 24 | 25 | 26 | 27 | 28  <클라>  Bitmap String  Setting Player name |
| 29 | 30  png texture mapping  result screen |  |  |  |  |  |