네트워크 게임 프로그래밍
텀프로젝트

2018182035 정재훈
2018180001 강민우
2018182033 이한빛

목 차

I. 애플리케이션 기획

1. 기획 단계

가. 게임 선택

(1) 게임 제목

slash and shoot

(2) 게임 제공자

제공 인원: 이한빛

재작년도: 2019년

사용 API: 윈도우 프로그래밍

나. 게임 설명

(3) 원본 게임 설명

2D 슈팅 게임으로 스테이지별로 나오는 몹들을 처리하여 생존하는 게임이다. 몹들은 플레이어에게 몰려들고 플레이어는 칼 또는 총을 쏴 적들을 물리친다. 스테이지별로 나오는 몹들의 수나 종류가 달라진다.

(4) 추가될 내용

서버를 이용하여 플레이어 간의 전투를 할 수 있도록 수정할 것이고 이제는 몹들이 단순히 플레이어를 공격하는 것이 아니라 플레이어 진형 소속이 되어 상대 플레이어를 공격하게 된다. 플레이어가 몹들을 잡으면 해당 몹들은 플레이어를 잡은 진형 소속으로 다시 생성되게 되며 생성되는 타이밍은 스테이지가 시작되었을 때이다.

다. 서버 설계

(5) 사용할 프로토콜

TCP/IP

(6) 서버 구현 내용

1. 실시간 리더보드(진형별 몹들의 수, 플레이어 점수)

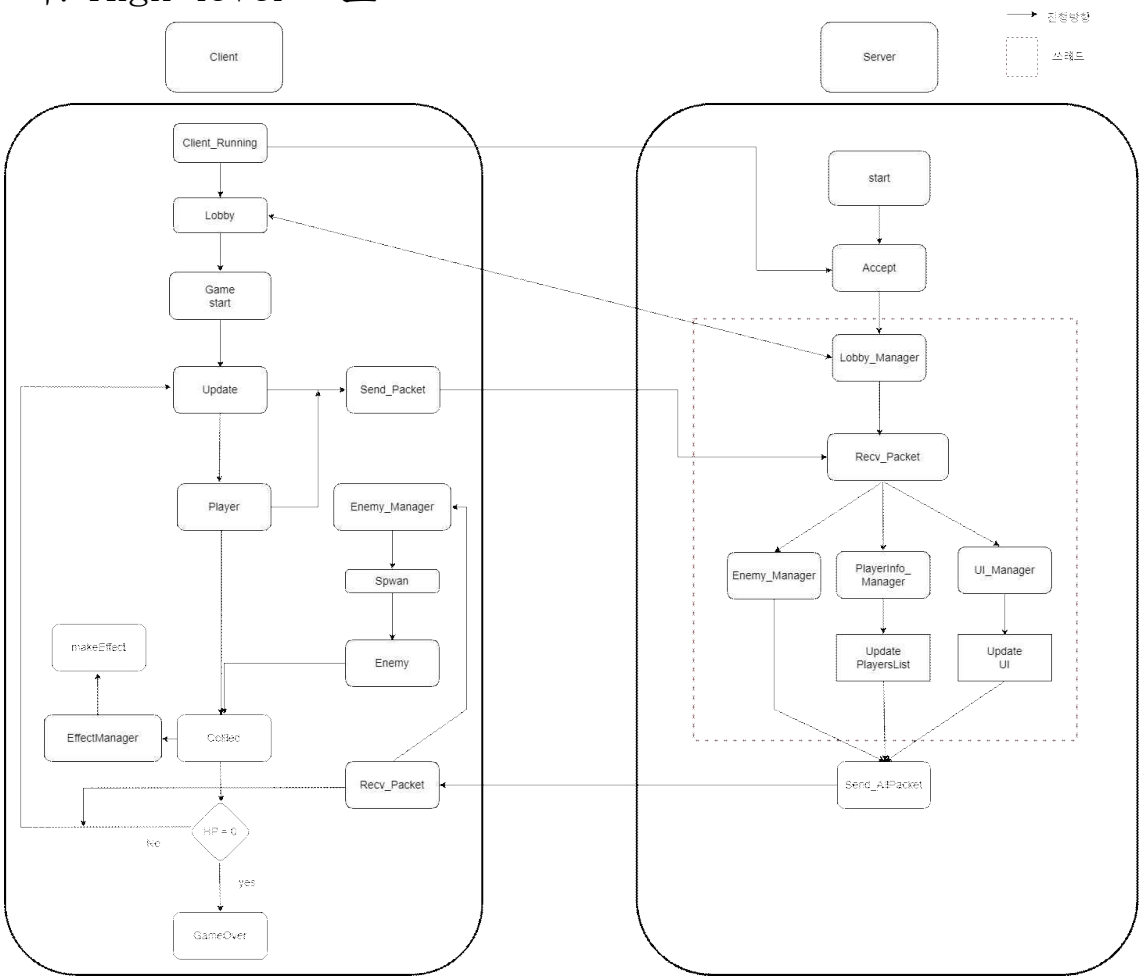
2. 상태 메시지(서버가 클라이언트들에게 스테이지 시작, 보스 출현 경고 메시지 등을 보냄)

3. 서버 동기화(서로 다른 클라이언트들의 정보를 출력하기 위해 클라이언트들의 준 시간을 통일하여 응답성을 높이기 위함)

II. Level 디자인

2. High-level 디자인

가. High-level 도표



나. 게임 흐름도(서버)

1. 클라이언트와 서버 구동 및 연결 - < Game Start, Accept >
2. 로비에서 플레이어들이 준비되면 게임 시작 - < Lobby, LobbyManager >
3. 서버에서 클라이언트가 보내는 패킷(Player_Info, Collision_Info)을 받아 처리 - < Recv_Packet >
4. Recv_Packet에서 몹 관리, 플레이어 정보 관리, UI 정보 관리(리더보드, 상태 메시지) 클래스에 필요한 정보를 넘김 - < PlayerInfo_Manager, Enemy_Manager, UI_Manager >
5. Send_AllPacket은 보낼 데이터를 묶어 클라이언트들에게 패킷 전송

Low-level 디자인

사용 패킷

struct PlayerInfo{
POINT pos
SWORD sword
int id
}
	struct ENEMYS{
	ENEMY EnemyList[500];
	}

struct UI{
int PlayerNum
int Stage
int MSG_TYPE
}
	struct LOBBYPACKET{
	int PlayerCount
	bool Ready, start
	int id
	}

struct ALL_PACKET{
PlayerInfo P_info[5]
ENEMY EnemyList;
UI Ui
}
	struct CollideInfo{
	int Enemy_id
	POINT pos
	}

(1) Client

▶ Class Clinet

▷ void init()

- 소켓 생성 및 bind 및 connect를 담당하고 RecvThread를 생성한다.

▷ void Send_Packet()

- 서버에 패킷을 보내는 역할을 담당한다.

▷ void Recv_Packet()

- 서버에서 오는 패킷을 분리(Player,Enemy,Ui)하여 각각 데이터를 처리하는 함수에 보내준다.

▷ DWORD WINAPI RecvThread(LPVOID arg)

- 서버에서 오는 패킷을 받는다. 서버에서 받은 정보(플레이어 정보, 충돌 정보,Enemy) 및 Lobby에 대한 패킷을 처리하는 클래스에 분담함.

▷ void UpdatePlayers(Player) - Player패킷 정보를 갱신한다.

▷ void UpdateEnemys(Enemy) - Enemy정보를 갱신한다

▷ void UpdateUi(UI) - Ui정보를 갱신한다.

▶ Class Lobby - 서버의 Lobby_Manager와의 통신을 담당

▷ void UpdateLobbyState() - 서버에게 게임 준비가 완료되었다는 정보를 보낸다.

▷ DWORD WINAPI RecvLobby(LPVOID arg) - LOBBYPACKET을 받아서 Lobby의 상태를 갱신한다.

▷ void initLobby() - 로비를 초기화하는 함수이며 서버에서 게임이 시작하는지에 대한 정보를 받기 위한 Recv 스레드를 생성한다.

▷ void DrawLobby() - 로비 화면을 그리는 함수

(2) Server

▶ Class Server

▷ void Init()

- 소켓 생성 및 bind()와 listen()을 담당한다.

▷ void Update()

- accept()를 담당하고 클라이언트별로 대응되는 쓰레드를 생성한다.

▷ DWORD WINAPI ProcessClinet(LPVOID arg)

- 클라이언트별로 요청을 처리하는 쓰레드

▷ void Send_AllPacket(PlayerInfo,UI,ENEMYS)

- 각각의 클래스에서 받은 정보를 갖고 데이터를 하나의 패킷으로 묶어 클라이언트들에게 보내줌.

▷ void Recv_Packet()

- 클라이언트에서 받은 정보(플레이어 정보, 충돌 정보)를 각 클래스에 분담함.
- 플레이어 정보는 PlayerInfo_Manager 클래스에 분담.
- 플레이어 정보 중 UI에 사용될 일부 정보를 UI_Manager 클래스에 분담.
- 충돌 정보를 Enemy_Manager 클래스에 분담.
- 정보를 나눠준 클래스들은 개별 스레드를 생성하여 작업.

▶ Class Lobby_Manager

- 로비에 접속한 클라이언트를 관리하는 클래스로 클라이언트의 준비가 완료되면 게임을 시작함.

▷ void WaitPlayer() - 플레이어를 기다리는 함수로 recv()를 통해서 플레이어가 준비된 상태인지 확인하고 모든 플레이어의 준비가 완료되면 GameStart()를 실행한다.

▷ void GameStart() - 게임 시작을 위해 각 클라이언트에게 시작을 알리는 정보를 보내고 로비 상태에서 클라이언트의 정보(PlayerInfo, CollideInfo)를 받는 상태로 전환한다.

▶ Class Enemy_Manager

- Enemy에 대한 정보는 Enemy_Manager 클래스가 일괄적으로 관리하며 충돌 정보를 이용하여 서버에서 Enemy의 개체수를 조절한다.

▷ void init() - EnemyList를 생성한다.

▷ void Recv(Collide) - 충돌 정보(Collide)패킷을 받는다.

▷ void EnemyState(Enemy) - Enemy의 정보를 받고 해당 Enemy의 정보를 갱신한다.

▷ Enemy* HandOverInfo() - EnemyList를 클라이언트에게 데이터를 넘겨주는 Send_AllPacket에 넘겨준다.

▶ Class PlayerInfo_Manager

- 서버에 접속한 클라이언트의 플레이어 정보들을 일괄처리 하는 클래스.

▷ void InitPlayer() - 최대 플레이어 수와 디폴트 플레이어 정보를 담는다.

▷ void RecvPlayer() - 플레이어 정보를 받고 플레이어 정보를 업데이트함.

▷ PlayerInfo* HandOverInfo() - 플레이어 정보를 종합하여 클라이언트에게 데이터를 넘겨주는 Send_AllPacket에 넘겨준다.

▷ void ErrorInfoCheck() - 현재 플레이어 정보로 넘어온 데이터들이 누락 또는 비정상적인지 확인하는 함수.

▶ class UI_Manager

- 리더보드를 갱신하고 상태 메시지(플레이어 화면에 보낼 메시지 - 스테이지 시작 전 또는 시작을 알리는 메시지)를 띄우기 위한 클래스.

▷ void init() - 리더보드를 초기화한다.

▷ Ui* HandOverInfo() - UI정보를 Send_AllPacket에 넘겨준다.

▷ void Recv_Ui(UI) - Recv_Packet()에서 넘어온 UI패킷을 받고 리더보드 데이터를 갱신한다.

▷ void Send_Msg() - 클라이언트에게 상태 메시지를 보낸다.

III. 개발환경 및 개발 일정

1. 팀원별 역할 분담

가. 팀원 I 역할(이한빛)

EnemyManager,UI_InfoManager

나. 팀원 II 역할(강민우)

Class Client, Class Server

다. 팀원 III 역할(정재훈)

Lobby, LobbyManager, Player_InfoManager,

2. 개발환경

Window 64비트 운영 체제, x64 기반 프로세서, WinApi

개발 툴 : Visual studio 2022

언어 : C++

3. 개발 일정

가. 일별/개인별 계획 수립

11월

이한빛

일	월	화	수	목	금	토
				3	4	5
6	7	8 Enemy_Manager void init()	9	10	11 Enemy_Manager void Recv(Collide)	12
13	14 중간점검	15 Enemy_Manager void EnemyState(Enemy)	16	17	18 Enemy* HandOverInfo()	19
20	21	22 UI_Manager void init()	23	24	25 UI_Manager Ui* HandOverInfo()	26 UI_Manager void Recv_Ui(UI)
27 UI_Manager void Send_Msg()	28 중간점검	29	30			

강민우

일	월	화	수	목	금	토
				3	4	5
6	7 Class Client init()	8	9 Class Server init() update()	10	11 Class Client RecvThread()	12
13	14 Class Server ProcessClient() 중간점검	15	16 Class Client Recv_Packet() Send_Packet()	17	18 Class Server Send_AllPacket Recv_Packet	19
20	21 Class Clinet UpdateEnemys()	22	23	24 Class Clinet UpdatePlayers()	25	26
27	28 중간점검	29 Class Clinet UpdateUi()	30			

정재훈

일	월	화	수	목	금	토
				3	4	5 Class LobbyManager WaitPlayers() 플레이어 접속 대기 구현
6 Class Lobby InitLobby() 클라이언트 로비를 생성.	7	8	9	10	11 Class Lobby RecvRobby() 로비 정보를 받는 스레드 함수 생성.	12 Class Lobby UpdateLobbyState() 게임 준비상태 갱신함수 생성.
13 Class Lobby DrawLobby() 로비 화면을 그리는 함수 생성.	14 중간점검	15	16	17 Class LobbyManager GameStart() 게임시작 함수 생성	18	19 Class PlayerInfo_Mangaer InitPlayer() 관리할 플레이어 정보 디폴트 생성
20 Class PlayerInfo_Manager RecvPlayer() 플레이어 갱신 함수 생성.	21	22	23	24	25	26 Class PlayerInfo_Manager HandOverInfo() 데이터를 종합하여 건네는 함수 구현
27 Class PlayerInfo_Manager ErrorCheckInfo() 넘어온 데이터의 오류를 검출하는 함수 생성	28 중간점검	29	30			

12월

이한빛

일	월	화	수	목	금	토
				1 점검	2 점검	3
4	5	6	7	8		

강민우

일	월	화	수	목	금	토
				1 점검	2 점검	3
4	5	6	7	8		

정재훈

일	월	화	수	목	금	토
				1 점검	2 점검	3
4	5	6	7	8		