**네트워크 게임 프로그래밍**

**Term Project**

**2015182002 고동현**

**2012182018 신우진**

**1. 게임기획**

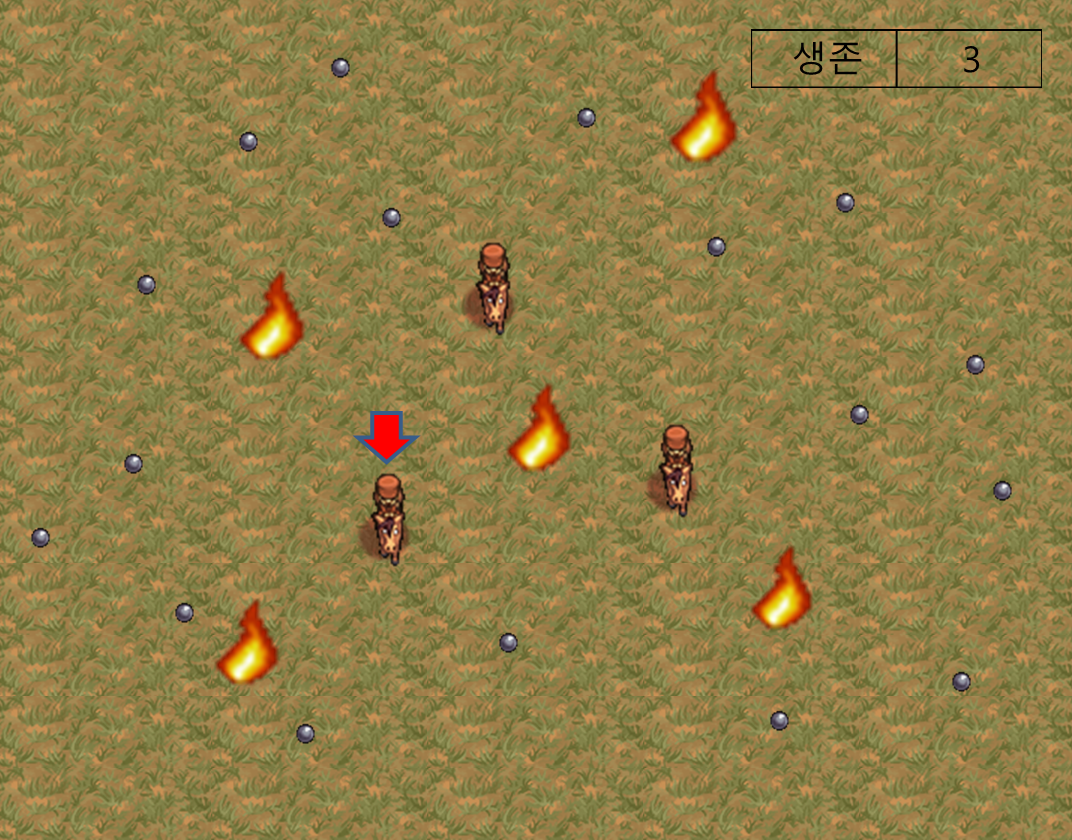
**<게임 설명>**

WIN\_API시간에 만들었던 미니게임을 토대로 TCP/IP 프로토콜을 사용해서 네트워크 통신을 통한 2인 총알 피하기 게임을 만들어보려 한다. 플레이어는 화면상에 보이는 오브젝트를 피해야 한다. 총알은 물론, 고정 장애물을 피해야 하며 랜덤으로 생성되는 아이템을 이용해 마지막까지 살아남으면 승리한다.

**<게임 규칙>**

* 2명의 플레이어가 모두 준비가 완료되면 게임을 시작한다.
* 사방에서 총알이 날라온다.
* 캐릭터는 상하좌우로 움직인다.
* 게임 시작할 때 고정 장애물이 생성된다.
* 고정 장애물은 끝날 때까지 한 자리에서 지속한다.
* 플레이어는 총알이나 장애물에 충돌하면 죽는다.
* 일정 시간이 지나면 랜덤 위치에 보호 아이템이 만들어진다.
* 보호 아이템을 먹으면 총알이나 장애물 충돌을 1회 무시해준다.
* 보호 아이템은 중복이 불가능하다..
* 자신 캐릭터 위에는 화살표 표시가 있다.
* 한 명의 플레이어가 남을 때까지 게임은 지속된다.

**<게임 스크린샷>**



**<서버와 클라이언트 구현>**

**1) Server**

- 접속한 클라이언트마다 ID를 부여한다.

- 접속한 클라이언트가 모두 준비가 완료되면 게임 시작 타임을 클라이언트에게 알려준다.

- 플레이어의 최초 위치 좌표를 알려준다.

- 고정 장애물의 최초 위치 좌표를 알려준다.

- 총알의 위치좌표를 갱신한다.

- 클라이언트의 키 입력 정보를 받는다.

- 플레이어의 위치좌표를 갱신한다.

- 총알 위치를 갱신한다.

- 보호 아이템 정보를 갱신한다.

- 플레이어와 오브젝트간 충돌처리를 한다.

- 1명의 플레이어가 살아남게 되면 게임종료

**2) Client**

- 서버에 접속을 시도하고 다른 클라이언트가 접속할 때 까지 ‘대기’라는 단어가 화면에 나온다.

- 모든 플레이어가 준비 되면 서버로부터 게임시작 타임을 받는다.

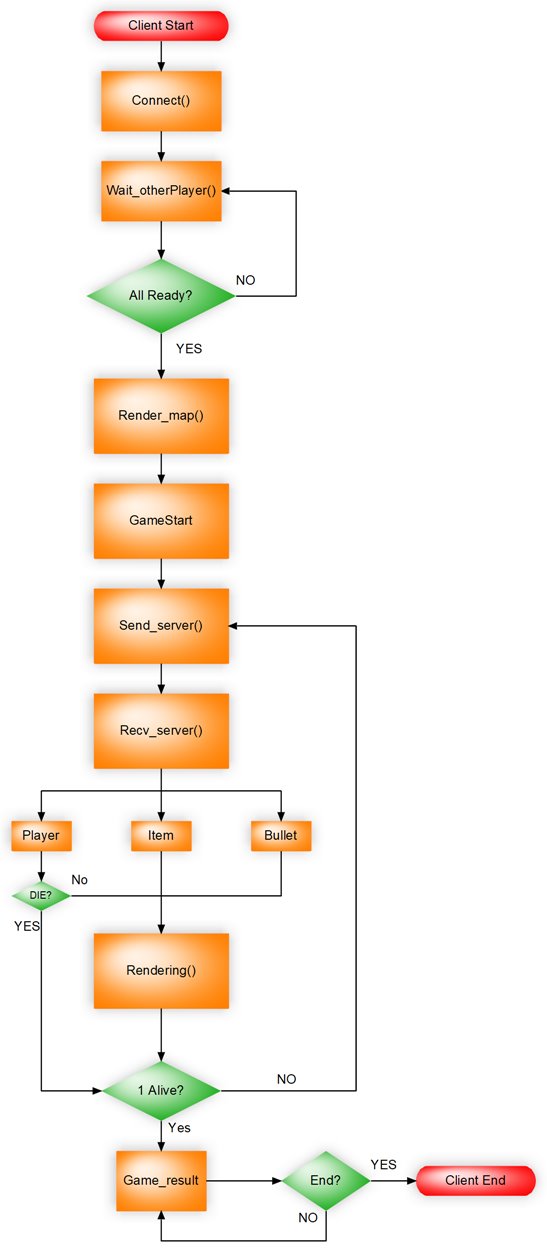
- 맵, 살아있는 플레이어, 아이템, 고정 장애물, 총알을 그린다.

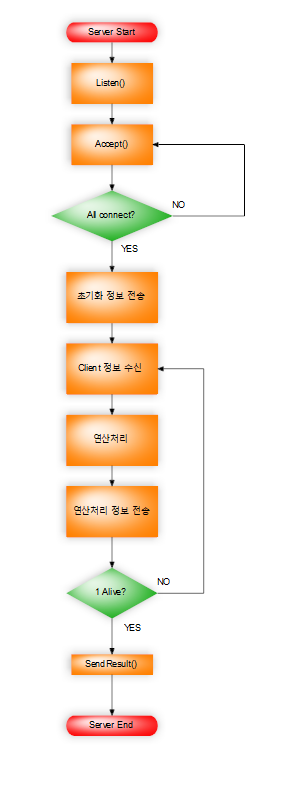
- 키 입력 값을 서버에게 보낸다.

**2. High Level Design**

**1) Server**

* bind()를 수행한 후, 클라이언트의 connect() 요청을 수용하기 전까지 accept()상태에서 머물러 있는다.
* 두 명의 클라이언트가 모두 접속하면 클라이언트에게 플레이어 인원 값을 보내준다.
* 접속한 클라이언트에게 초기화된 게임정보를 전송해준다.
* 1초에 30번 연산된 데이터를 클라이언트에 전송해준다.
* 접속한 클라이언트에서 캐릭터의 정보를 받아서 총알, 장애물, 아이템과의 연산처리를 진행한다.
* 생존 플레이어 수가 한 명일 경우 게임 결과를 보내준다.

  
 <서버 플로우 차트> <클라이언트 플로우 차트>



**2) Client**

* 메뉴화면에서 “게임준비” 버튼을 누르면, 서버에 connect 함수로 연결을 시도한다.  
  (게임준비 버튼은 대기중으로 바뀐다.)
* 서버로부터 인원 값을 받았으면 대기중 버튼이 게임시작으로 바뀌고 3초의 카운터를 띄운다.
* 플레이어 캐릭터와 다른 플레이어 캐릭터, 고정 장애물의 위치 등 초기값을 서버에서 받는다.
* 사용자 키보드 입력 값을 서버로 보내준다.
* 서버에서 받은 데이터로 각 플레이어 위치, 아이템, 총알 위치를 업데이트 후 화면에 보여준다.
* 서버로부터 게임 결과를 받으면 게임은 중지하고 승리/패배를 화면에 보여준다.

**3. Low Level Design**

**1) 공용프로토콜**

**플레이어가 누른 키를 알려준다.**

* **#define RIGHT 0, #define LEFT 1**
* **#define DOWN 2, #define UP 3**

**플레이어의 상태를 알려준다.**

* **enum Result {WIN, LOSE}**

**캐릭터 패킷**

* **struct Object {   
  BYTE x, BYTE y,   
  int key, BYTE id,   
  bool protect, Result result\_state; }**

**총알 패킷**

* **struct Bullet {BYTE x, BYTE y}**

**아이템 패킷**

* **struct Item {BYTE x, BYTE y }**

**장애물 패킷**

**- struct Obstacle{BYTE x, BYTE y, bool exist }**

**2) Server**

**총알과 플레이어 위치정보를 초기화하는 함수**

* **void Initialize()**

**키보드 입력을 처리해주는 함수**

* **void Check\_Kayboard()**

**클라이언트에 플레이어 패킷을 보내주는 함수**

* **void SendObject(Object client, int client\_id)**

**클라이언트에 총알패킷을 보내주는 함수**

* **void SendBullet(Bullet bullet, int client\_id)**

**클라이언트에 게임 결과를 알려주는 함수  
 - void SendResult( Result result , int client\_id )**

**통신을 위한 스레드를 만들어준다.**

* **DWORD WINAPI MainServer(LPVOID arg)**

**각 클라이언트에서 넘어온 정보를 통해서 연산처리하는 스레드를 만들어준다.**

* **DWORD WINAPI ProcessPacket(SOCKET)**

**동기화 방법 중 이벤트로 제어**

**- CreateEvent(), WaitFor\*()**

**3) Client**

**오브젝트를 그려준다**

**- void Object\_Render();**

**UI를 그려준다**

* **Void UI\_Render();**

**서버에 데이터를 송신하는 함수로 키 입력 값을 보낸다  
 - void Send\_server();**

**서버로부터 데이터를 수신하는 함수로 게임 상태를 전달 받는다  
 - void Recv\_server();**

**4. 팀원 별 역할분담**

**신우진**

* **Project Manager**
* **클라 Send, Recv 구현**
* **서버 충돌체크 구현**
* **동기화**

**고동현**

* **서버 recv,send 구현**
* **서버 충돌체크 구현**
* **클라이언트와 서버 통신 구현**
* **MainServer () 구현**
* **동기화**

**5. 개발환경**

- Windows 7

- Visual Studio 2017

- C / C++

- Window API

**6. 개발일정**

|  |  |  |
| --- | --- | --- |
| **날짜** | **신우진** | **고동현** |
| **11/21** | 기획서 수정 및 클라이언트 수정 | |
| **22** |
| **23** | 클라이언트 수정 | |
| **24** | 클라이언트 Initialize()구현 | 기본 TCP/IP 프로토콜 세팅 |
| **25** |
| **26** | 클라 플레이어 Key값 send 해주기 | 서버 Initialize() 구현 |
| **27** |
| **28** | 서버에서 총알 정보 받기Recv\_server(); | 서버 플레이어 Key값 recv 해주기 |
| **29** |
| **30** | Object\_Render() 구현 | 클라에 총알 위치 보내기 SendBullet() |
| **12/1** |
| **2** | 1인 플레이어로 테스트 | 1인 플레이어로 테스트 |
| **3** | 정기회의(진행 및 수정사항 체크) | |
| **4** | 서버에서 플레이어와 총알 충돌체크 ProcessPacket() | 2인 플레이어 작업 MainServer() |
| **5** |
| **6** | 서버에서 플레이어와 맵 충돌체크 ProcessPacket() | 2인 플레이어 작업 MainServer() |
| **7** | 2인 플레이어로 테스트 |
| **8** | 서버에서 아이템 정보 받기Recv\_server(); | 아이템 위치 생성 ProcessPacket() |
| **9** |
| **10** | 정기회의(진행 및 수정사항 체크) | |
| **11** | 서버에서 플레이어와 아이템 충돌체크 ProcessPacket() | 서버에서 플레이어와 장애물 충돌체크 ProcessPacket() |
| **12** |
| **13** | 그래픽 추가 작업 | 총알 관리  ProcessPacket() |
| **14** |
| **15** | 동기화 작업 | 동기화 작업 |
| **16** | 최종 테스트 및 보안 | |
| **17** | 최종 테스트 및 보안 | |
| **18** | 최종검수 | |

|  |  |  |
| --- | --- | --- |
| 2017.11.24(금) | | |
|  | 우진 | 동현 |
| 개발 일정 | 클라이언트 Initialize()구현 | 기본 TCP/IP 프로토콜 세팅 |
| 작업 내용 | 클라이언트 TCP/IP 통신 기본 세팅 | 서버 TCP/IP 통신 기본 세팅 |

|  |  |  |
| --- | --- | --- |
| 2017.11.25(토) | | |
|  | 우진 | 동현 |
| 개발 일정 | 클라이언트 Initialize()구현 | 기본 TCP/IP 프로토콜 세팅 |
| 작업 내용 | connect()후 client id 받기 | 클라이언트에 id값 주기 |

|  |  |  |
| --- | --- | --- |
| 2017.11.26(일) | | |
|  | 우진 | 동현 |
| 개발 일정 | 클라 플레이어 Key값 send 해주기 | 서버 Initialize() 구현 |
| 작업 내용 | 1. 클라이언트에서 초기 위치 값 받아 캐릭터 생성 2. 클라 keyboard값 send | 클라이언트에 캐릭터 초기 위치 값 보내주기 |

|  |  |  |
| --- | --- | --- |
| 2017.11.27(월) | | |
|  | 우진 | 동현 |
| 개발 일정 | 클라 플레이어 Key값 send 해주기 | 서버 Initialize() 구현 |
| 작업 내용 | 1. 클라 keyboard값 send 2. 클라 총알정보 받기 3. 플레이어 그래픽 추가 | 1. 클라에서 키보드값 받아서 연산처리 2. 클라에 총알 위치 연산 값 보내주기 |

|  |  |  |
| --- | --- | --- |
| 2017.11.28(화) | | |
|  | 우진 | 동현 |
| 개발 일정 | 서버에서 총알 정보 받기Recv\_server(); | 서버 플레이어 Key값 recv 해주기 |
| 작업 내용 | 1. 진행 사항 정리 2. 1인 플레이 테스트 | 1. 총알과 플레이 충돌체크 연산 추가 2. 총알 그래픽 CImage를 이용해 추가 3. 2인 플레이 추가 |