**네트워크 게임 프로그래밍**

**텀프로젝트 추진 계획서**

2013182055 최치송

2018180048 박의인

**애플리케이션 기획**

|  |  |
| --- | --- |
| 800x 800(UI 200 x 800) | **<갤러가>** 리소스를 이용한 1:1 슈팅게임  상대플레이어를 추가하여 비행기를 파괴하거나 화면 너머로 드론을 보내서 체력을 0으로 만들면 승리하는 게임  사용자는 1P 위치에서 플레이하며  상대가 2P역할을 수행한다 |

|  |  |
| --- | --- |
|  | TITLE: 타이틀 씬  1p,2p 중 하나를 선택하여 게임준비  ROOM: 대기방  -1p: 서버 플레이어  -2p: 클라이언트 플레이어  양 플레이어가 준비되면 PLAY로 이동  PLAY: 게임 씬  - 둘 중 한명의 HP가 0이 되면 Print Score로 이동  Print Score: 점수 출력  - 통신을 종료하고 각 플레이어가 얻은 점수 출력 후, 아무 키나 누르면 타이틀로 이동 |

좌우 방향키: 비행기가 좌, 우로 움직인다.

Z키: 탄환 발사

X키: 드론 출격

비행선이 파괴되면 HP-1

드론이 화면뒤로 넘어가면 HP-2

**캐릭터**

|  |  |  |  |
| --- | --- | --- | --- |
| 75x80 | 55x60 | 75x80 | 55x60 |
| 사용자 비행선 | 사용자 드론 | 상대 비행선 | 상대 드론 |

**하이레벨 디자인 -\*코드 구조는 추후에 추가**

|  |  |
| --- | --- |
|  | 모든 Manager 클래스를 싱글톤 패턴으로 작성하여 모든 GameNode에서 Manager초기화 시켜 GameNode를 상속받은 모든 클래스에서 Manager를 사용할 수 있게 작성함 |

모든 Manager 클래스를 싱글톤 패턴으로 작성하여 모든 GameNode에서 Manager초기화 시켜 GameNode를 상속받은 모든 클래스에서 Manager를 사용할 수 있게 작성함

**대기방**

\*\*recv()는 WSA Manager는 server와 client가 연결된 뒤, 상시에 thread를 통해 recv() 호출

1 ➞ Server플레이어가 WSA Manager를 통하여 서버를 생성 후, Accept() 호출

2 ➞ Client플레이어가 WSA Manager를 통하여 생성된 서버에 Connect()

3 ➞ Room 씬에서 버튼이 클릭될 때마다 send()를 통하여 버튼 상태값 전송

**인게임 - 상대 플레이어와의 동기화**

1 ➞ 플레이어1 - thread를 통하여 WSA Manager로 recv()를 수행

2 ➞ 플레이어2 - KeyButtonDown()과 KeyButtonUp() 시 send() 함수를 통해서 키값 전송

3 ➞ 플레이어1 - 수신받은 값를 enemy\_controler클래스의 InputData(char data)를 통하여 값에 따라 플레이어2의 상태를 변경하거나 오브젝트(총알, 드론) 생성

**로우레벨 디자인**

설명의 위해 필요한 정보

|  |  |
| --- | --- |
| struct Point  {  float x, y;  }  대상의 x, y 위치로 사용 | enum ENEMYSTATE  {  ES\_IDLE,  ES\_LEFT,  ES\_RIGHT  }  대전 상대의 캐릭터의 상태값에 사용 |

class WSAManager를 싱글톤으로 작성하여 통신을 주고받도록 설계하고 해당 클래스의 멤버함수 중 아래의 것들을 사용하여 통신함

|  |
| --- |
| 1. void Accept\_Socket()    1. 서버가 룸생성시 accept()를 호출한 뒤 recv()를 게임종료시까지 호출하는함수    2. thread를 통하여 실행되는 함수    3. recv()로 받아온 함수는 클래스 내의 버퍼에 저장됨 2. void Server\_Send(const char buff)    1. 서버가 버퍼의 데이터를 전송하기 위한 함수 3. void Connect\_Socket()    1. 클라이언트가 룸입장시 connect()를 호출한 뒤 recv()를 게임종료시까지 호출하는함수    2. thread를 통하여 실행되는 함수    3. recv()로 받아온 함수는 클래스 내의 버퍼에 저장됨 4. void Client\_Send(const char buff)    1. 클라이언트가 버퍼의 데이터를 전송하기 위한 함수 5. char getServerBuffData()    1. 클래스 멤버변수로 선언된 버퍼에서 데이터를 가져오는 함수 6. void resetBuff()    1. 클래스 멤버변수로 선언된 버퍼의 값에 NULL을 대입하는 함수 |

위에서 받아오는 정보는 class EnemyControler를 작성하여 대전 상대가 행동값을 입력할때마다 send()를 호출하여 데이터를 전송하여 게임 내에 적의 모습에 반영함

|  |
| --- |
| 1. void ActCodeInput(char CODE)    1. 버퍼의 값을 입력하여 작동하는 함수    2. 인자값에 따라서 하단에 선언된 함수들을 호출하도록 작성 2. void SpawnDrone(Point p)    1. 인자로 받아온 위치값에 EnemyDrone을 생성 3. void ShootBullet((Point p)    1. 인자로 받아온 위치값에 EnemyBullet을 생성 4. void ChangeShipState(State state)    1. 인자값을 EnemyPlayer의 State 값에 대입하여 캐릭터 상태변경 |

이후 둘중 하나의 HP가 0이되면 0이 된 플레이어가 게임오버와 관련된 Define값을 전송하고 이긴 플레이어의 프로그램 내의 승패관련 정보(1pHP, 2pHP)가 일치하는지 확인하고 일치하는 경우 게임을 종료시키고 점수판Scene으로 넘어간다.

※만약 승패관련 정보가 불일치시 메세지를 출력하고 강제종료 시킨다.

상황에 따라 통신을 위해서 사용하는 데이터의 Define값은 아래와 같으며 char를 넘어가지 않게한다.

**대기방**

|  |  |  |
| --- | --- | --- |
| Define Name | Mean | value |
| NOT\_READY | 플레이어 준비안됨 | 0x01 |
| BE\_READY | 플레이어 준비완료 | 0x02 |
| ENEMY\_NOT\_READY | 상대 플레이어 준비안됨 | 0x04 |
| ENEMY\_READY | 상대 플레이어 준비완료 | 0x08 |
| GAME\_PLAY | PLAY 씬으로 씬 변경 | 0x10 |

**인게임**

|  |  |  |
| --- | --- | --- |
| Define Name | Mean | value |
| ENEMY\_NONE | 적 플레이어 정지 | 0x01 |
| ENEMY\_LEFT | 적 플레이어 왼쪽으로 이동 | 0x02 |
| ENEMY\_RIGHT | 적 플레이어 오른쪽으로 이동 | 0x04 |
| ENEMY\_SHOT | 적 플레이어 총알 발사 | 0x08 |
| ENEMY\_DRONE | 적 플레이어 드론 발사 | 0x10 |
| GAME\_OVER | 플레이어 중 한 명의 HP가 0이 되었다 | 0x20 |

**TODO 리스트**

씬작업(4)

: 타이틀, 룸, 인게임, 점수창

윈도우(6+3)

: 플레이어, 드론, 총알 (아군/적)

* 드론, 총알의 개체수는 vector를 사용하여 관리

EnemyControler

* 플레이어, 드론, 총알의 작업이 완료된 뒤에 작업될 것

CollisionManager

- 충돌이 필요한 객체들의 주소값을 멤버변수에 대입하여 충돌계산

WinSock(2)

* WSAManager 클래스에 논블로킹 적용, WSAManager(Client Manager/Sever Manager)

**팀원 별 역할분담**

|  |  |  |
| --- | --- | --- |
|  | 최치송 | 박의인 |
| 역할 | Sever Manager, WSAManager 클래스에 논블로킹 적용,  EnemyControler,  씬작업 | Client Manager,  플레이어 클래스,  드론 클래스,  총알 클래스,, CollisionManager |

**개발환경**

|  |
| --- |
| 운영체제: 윈도우  개발환경: 비주얼 스튜디오 2019  API: Window API  버전관리: github |

**개발일정 (일별/개인별)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|  |  |  | 11/4 | 11/5 | 11/6 | 11/7 |
|  |  |  | Client Manager  소켓 생성, 소멸 | Player class 작성  초기화 관련 멤버함수와 private 멤버변수 접근을 위한 GetSet함수 작성 | | |
| 11/8 | 11/9 | 11/10 | 11/11 | 11/12 | 11/13 | 11/14 |
| Player class  키 입력에 따른 상태 변화 추가 | | Client Manager  nonbloking buff test | 중간점검 | | | |

**박의인**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 11/15 | 11/16 | 11/17 | 11/18 | 11/19 | 11/20 | 11/21 |
| 중간점검 | | | Drone class 작성  초기화 관련 멤버함수와 private 멤버변수 접근을 위한 GetSet함수 작성 | | |  |
| 11/22 | 11/23 | 11/24 | 11/25 | 11/26 | 11/27 | 11/28 |
| Bullet class 작성  초기화 관련 멤버함수와 private 멤버변수 접근을 위한 GetSet함수 작성 | | CollisionManager  플레이어와 드론, 총알 충돌 | | | CollisionManager  총알, 드론 충돌 | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 11/29 | 11/30 | 12/1 | 12/2 | 12/3 | 12/4 | 12/5 |
| 최종점검 | | | | | | |
| 12/6 | 12/7 | 12/8 | 12/9 | 12/10 | 12/11 | 12/12 |
|  |  |  |  |  |  |  |

**최치송**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|  |  |  | 11/4 | 11/5 | 11/6 | 11/7 |
|  |  |  | ServerManager class 작성  소켓생성부터 소켓종료까지 구현 | ServerManager class의 논블로킹  buff 로직 적용 | | |
| 11/8 | 11/9 | 11/10 | 11/11 | 11/12 | 11/13 | 11/14 |
| ServerManager class의 논블로킹  buff 로직 적용 | | ServerManager, ClientManager  논블로킹 버퍼 통신 테스트 | 중간점검 | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 11/15 | 11/16 | 11/17 | 11/18 | 11/19 | 11/20 | 11/21 |
| 중간점검 | | | EnemyControler  버퍼 데이터 순차처리를  위한 로직 작성 | | | |
| 11/22 | 11/23 | 11/24 | 11/25 | 11/26 | 11/27 | 11/28 |
| EnemyControler  총알 생성기능 추가 | EnemyControler  드론 생성기능 추가 | Room Scene class 작성  통신 관련 Manager 클래스를 호출하여 소켓초기화 작업 | | InGame Scene class 작성  오브젝트 및 UI 배치 | | TITLE/Score Scene class 작성  Manager클래스의 함수를 사용하여 소켓 통신 종료 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 11/29 | 11/30 | 12/1 | 12/2 | 12/3 | 12/4 | 12/5 |
| 최종점검 | | | | | | |
| 12/6 | 12/7 | 12/8 | 12/9 | 12/10 | 12/11 | 12/12 |
|  |  |  |  |  |  |  |