|  |
| --- |
| **네트워크 게임 프로그래밍** |
| **“Crash Castle"** |

2013184043 양태윤

2015182046 강래민

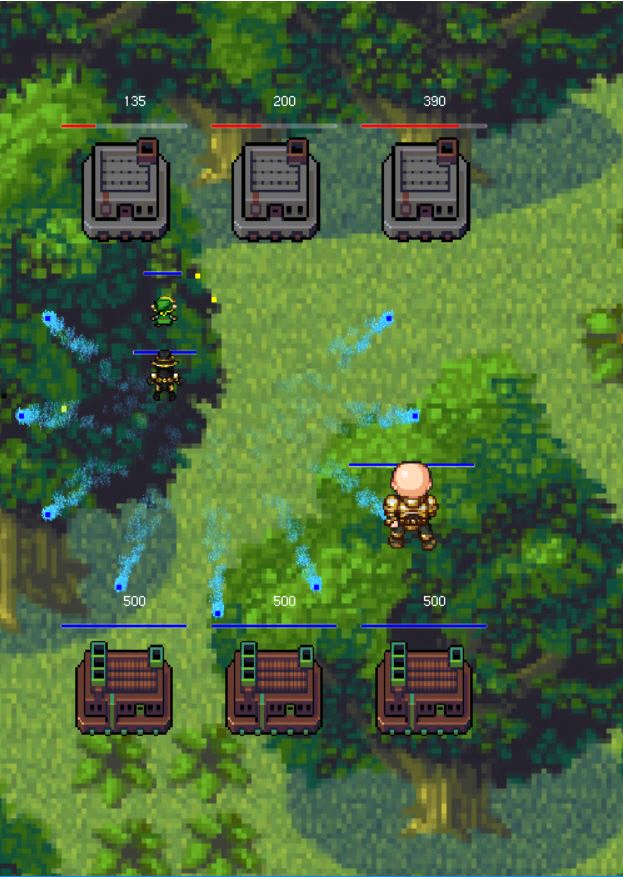
2015182053 이상호

|  |
| --- |
| **목차**  **1. Application 기획** |
| * **게임 소개** |
| * **게임 오브젝트** |
| * **게임 설명** |
| * **게임 조작** |
| * **게임 진행** |
| **2. High Level 디자인** |
| * **Server Design** |
| * **Client Design** |
| * **오류 처리** |
| * **소켓 타입** |
| **3. Low Level 디자인** |
| * **Server Design** |
| * **Client Design** |
| * **오류 처리** |
| * **동기화** |
| **4. 역할 분담** |
| **5. 개발 환경** |
| **6. 개발 일정** |

**1.Application 기획**

* 게임소개

Open GL과 2D 스프라이트 리소스를 사용하여 만든 클래시 로얄 방식의 게임이다. 게임은 2명의 플레이어의 건물, 투사체(미사일,총알), 유닛(궁수,전사,마법사) 등으로 구현되며 . 플레이어가 화면영역에 마우스 좌 클릭을 하면 해당지점에 랜덤으로 캐릭터가 생성되어 정해진 행동패턴대로 움직이며 공격을 시작한다. 2명의 플레이어가 1:1로 대결하며 플레이어는 이동키를 통해 성을 움직여 유닛과 총알을 피할 수 있다.



* 게임 오브젝트



: 궁수 유닛 -> 빠르게 사방으로 움직이며 랜덤 한 방향으로 총알을 발사(HP : 100)



: 마법사 유닛 -> 천천히 사방으로 움직이며 12방향으로 미사일을 발사(HP : 100)



: 전사 유닛 -> 오직 앞으로만 나아가며 근접공격(HP : 250)



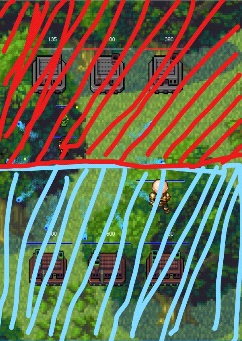
: 1P의 건물 (HP : 500), 키보드로 이동

: 2P의 건물 (HP : 500), 키보드로 이동

 : 궁수 유닛이 발사하는 총알 (데미지 : 10)



: 마법사 유닛이 발사하는 미사일 (데미지 : 15)

* 게임 설명
* 이 게임은 동시에 2명이 함께 플레이하는 PVP방식이다.
* 마우스 클릭으로 지정된 영역에 유닛을 배치할 수 있다.
* 
* 각 플레이어의 지정된 영역은 화면의 1/2의 영역이다.

(각각 블루 플레이어와 레드 플레이어의 배치 가능 구역)

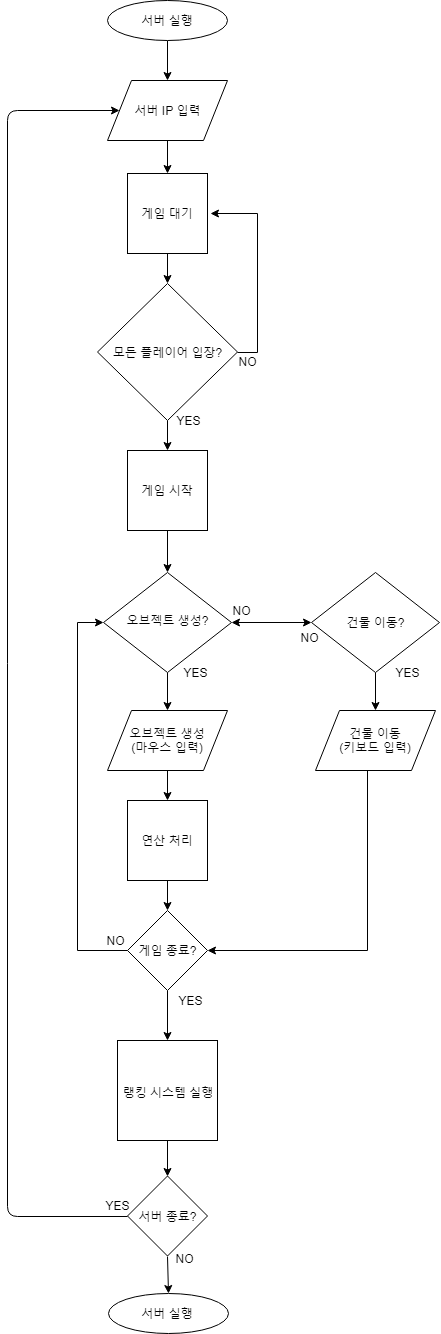
* 오브젝트가 유닛이나 투사체와 충돌하게 되면 해당 오브젝트의 HP가 줄어들게 되고. HP 가 0인 오브젝트는 사라진다.
* 아군오브젝트나 투사체 에는 충돌을 해도 데미지를 입거나 소멸하지 않고 그냥 통과한다
* 유닛들은 화면 가장자리에 부딪힐 시 반대방향으로 뒹기며 투사체 들은 그냥 소멸한다.
* 먼저 상대방의 건물을 모두 제거하는 플레이어가 승리한다.
* 게임조작
* 유닛 배치

**마우스 좌클릭**으로 원하는 지점에 유닛을 배치할 수 있으며 배치 후 1초의 시간이 흘러야 새로운 유닛을 좌클릭으로 배치할 수 있다.

* 키보드 조작

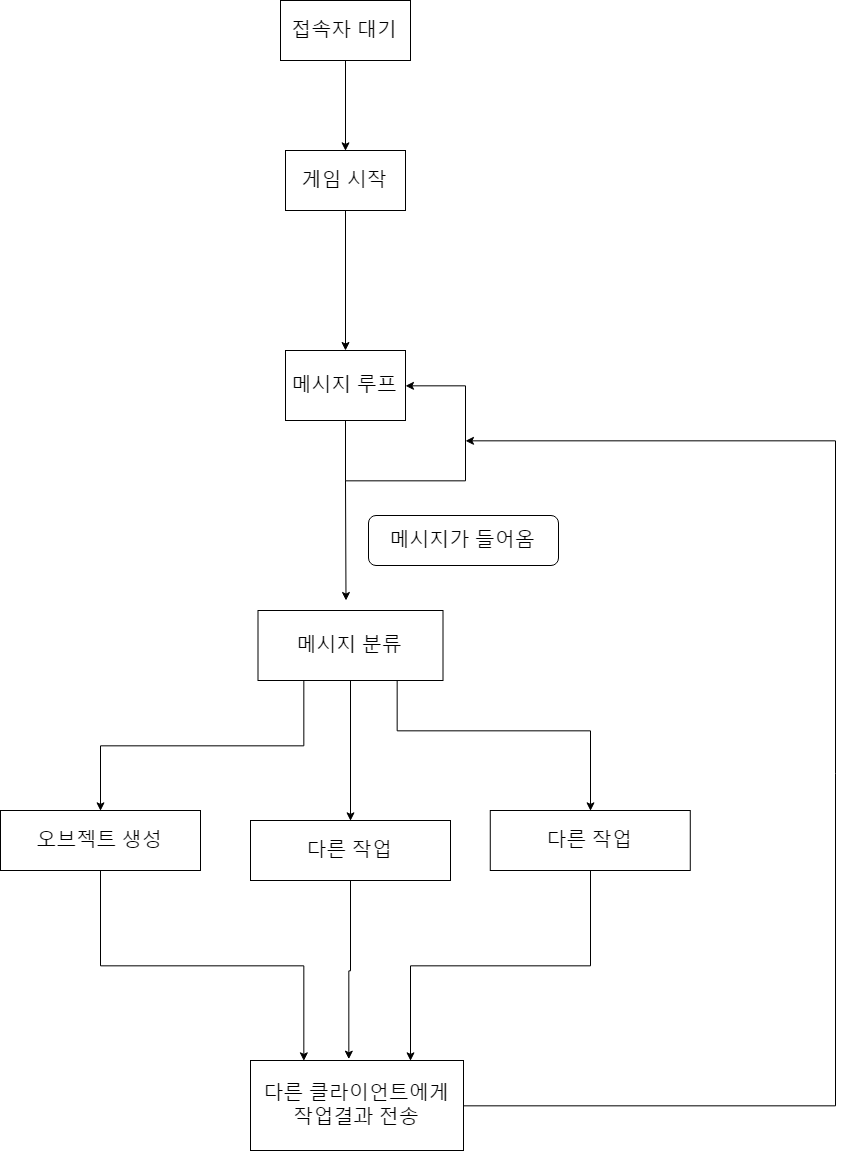
키보드 방향키로 건물을 움직일 수 있다.

**2. 게임 진행 순서도**



**3. High Level 디자인**

* **개발 환경**
* 윈속 버전 : 2.2
* 주소체계 : IPv4
* 사용 프로토콜 : TCP/IP
* 사용 언어 : C/C++
* IDE : Visual Studio 2017
* **서버**



* **순서도 설명**
* 접속자 대기

모든 접속자가 들어올 때까지 게임을 대기합니다.

* 메시지 루프

클라이언트로부터 메시지가 들어올 때까지 기다립니다.

* 메시지 분류/작업

메시지가 들어오면 메시지 분류하고 그 메시지에 맞는 작업을 합니다.

* 다른 클라이언트에게 작업결과 전송

작업한 결과를 다른 클라이언트들에게 작업 결과를 알려줍니다. 그리고 다시 메시지 루프로 들어갑니다.

* **메시지 종류**
* 오브젝트 생성 메시지

오브젝트를 생성합니다.

* 게임 종료 메시지

게임을 종료합니다.

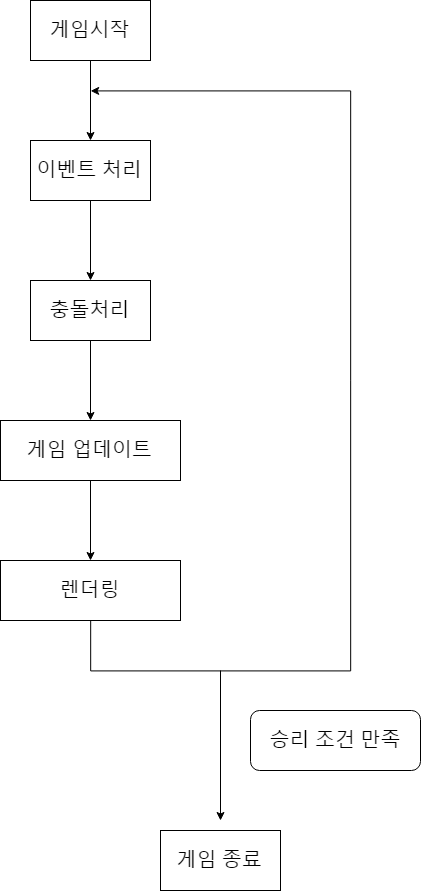
* 게임 시작 메시지

모든 접속자가 들어와서 게임을 시작합니다.

* 키보드 입력 메시지

사용자가 키보드 입력을 통해 건물을 움직입니다.

* **클라이언트**



* **순서도 설명**
* 이벤트 처리

마우스, 키보드, 기타 이벤트에 대한 처리를 합니다.

서버에 보낼 이벤트면 서버로 보내고 서버로부터 다른 이벤트들도 입력받습니다.

* 충돌 처리

게임 내 총알과 오브젝트에 대해 충돌처리를 합니다.

* 게임 업데이트

이벤트, 충돌에 대한 오브젝트들의 상태를 업데이트 합니다.

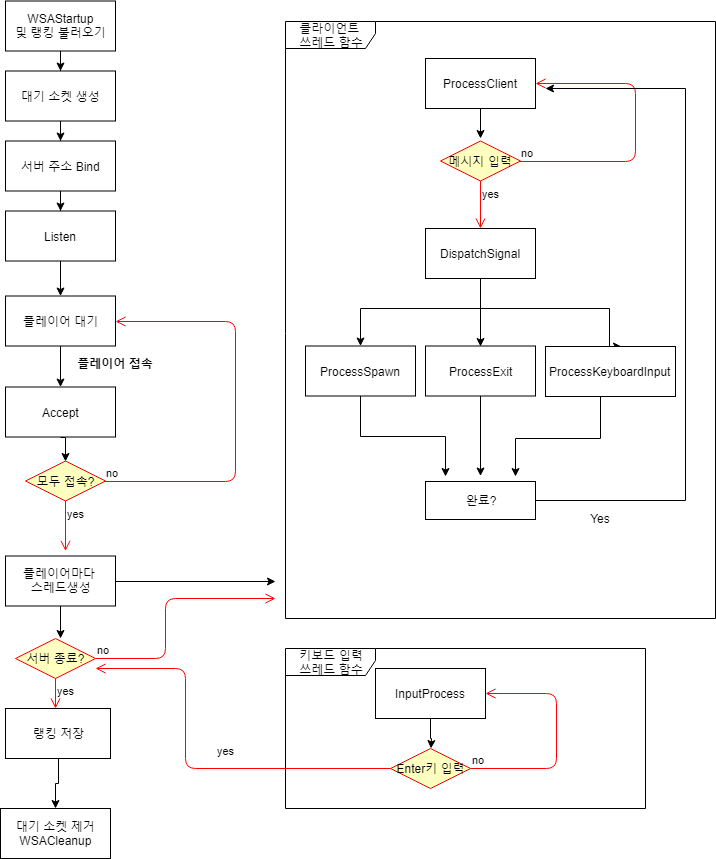
* 렌더링

업데이트 된 상태로 화면을 그립니다.

* 게임 종료
* 승리조건이 만족하면 게임을 종료하고 아니면 다시 처음으로 갑니다.
* **이벤트 종류**
* 마우스 입력을 받아 유닛 생성
* 키보드 입력을 받아 유닛 이동
* 승리조건에 맞아 게임 승리
* 패배조건에 맞아 게임 패배
* 키를 눌러 게임 종료
* 기타 다른 기능들

**4. Low Level 디자인**

* **서버**

****

* **순서도 설명**
* 초기화 부분

WSAStartup – 윈도우 소켓을 사용하기 위한 준비를 합니다.

대기소켓을 생성합니다.

Bind로 서버 주소 구조체를 연결한 후 Listen함수로 Listening 상태로 만듭니다.

* 플레이어 대기

~~Accept로 플레이어를 수용하고 소켓과 ID를 배열에 넣어둡니다.~~

Accept로 플레이어를 수용하고 소켓과 ID를 unordered\_map에 넣어둡니다.

모든 플레이어가 접속되었다면 다음으로 진행합니다.

대기 소켓은 넌블록킹 소켓으로 만들어 Accept에서 블록 되지 않고 넘어가도록 만듭니다. 키입력이 됬을 때 서버 루프를 나와 정상 종료합니다.

* 플레이어 모두 접속

플레이어가 모두 접속했다면 연결된 소켓별로 스레드를 생성합니다.

플레이어에게 게임 시작 메시지를 보냅니다.

* 랭킹

게임이 종료되면 랭킹을 합산하여 서버로 랭킹정보를 보내 서버의 랭킹파일에 저장을 합니다

* 플레이어 모두 종료

플레이어 접속이 모두 종료되었다면 대기소켓을 제거하고

윈도우 소켓도 종료합니다.

* 쓰레드 함수
* ProcessClient

반복문을 돌면서 SIGNAL로 정의된 상수를 기다립니다.

SIGNAL이 들어오면 DispatchSignal함수로 넘겨줍니다.

* InputProcess

Enter키가 눌렸는지 확인하고 눌렸다면 서버에 종료신호를 보냅니다.

* DispatchSignal 함수

각 상수에 맞는 함수를 호출합니다.

* ProcessSpawn 함수

클라이언트에서 유닛이 생성될 때 SIGNAL\_SPAWN을 보내는데 그에 대응하는 함수입니다. 클라이언트로부터 새로운 유닛이 생성된 위치, 타입, 소속, 초기 이동방향을 전송받아 연결된 다른 클라이언트들에게 전달합니다. 클라이언트는 받은 데이터로 그 위치에 해당하는 유닛을 만듭니다.

* ProcessExit 함수

클라이언트가 종료 시 SIGNAL\_EXIT을 보내는데 그에 대응하는 함수입니다. 받은 신호를 연결된 다른 클라이언트들에게 전달합니다. 클라이언트는 나간 플레이어의 모든 유닛을 제거합니다.

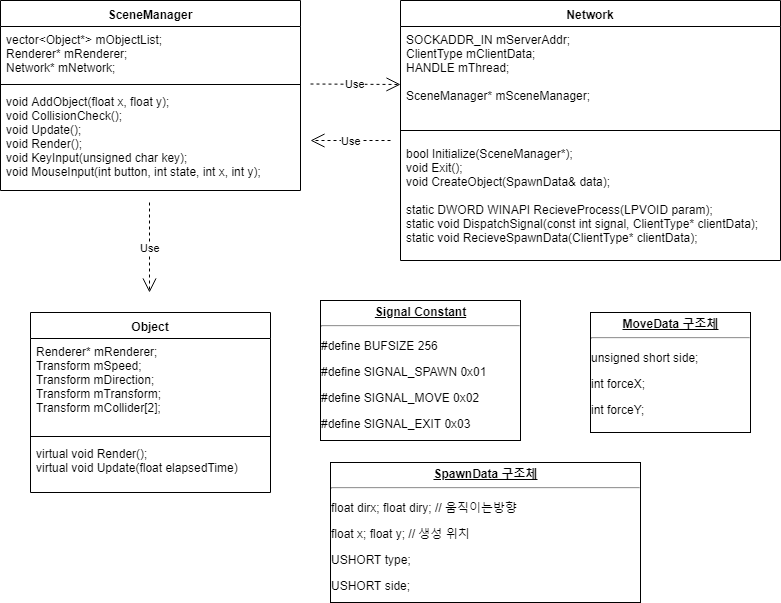
* ProcessKeyboardInput함수

클라이언트에서 키보드 입력으로 플레이어가 움직이면 SIGNAL\_MOVE를 받는데 이동방향, 거리, 소속을 받아 상대편 클라이언트에게 알려줍니다.

* 랭킹 시스템

게임이 끝나면 해당 플레이어의 이름과 점수를 받아 서버에 파일로 저장해둡니다. 다시 게임을 시작하면 랭킹을 확인할 수 있고 클라이언트에서SIGNAL\_EXIT이 보내질 때 ProcessExit함수 내에서 추가로 자신의 점수와 이름을 입력받아 랭킹에 입력할 수 있습니다.

* **클라이언트**



* UML 설명
* SceneManager 클래스

화면에 그려지는 오브젝트들을 관리합니다.

텍스쳐, 소리 등 리소스등을 생성될 때 입력받고 관리합니다.

화면을 그리는 Renderer클래스를 초기화하고 관리합니다.

Update함수에서 갖고 있는 오브젝트들의 Update함수를 호출합니다.

Render함수에서 갖고 있는 오브젝트들의 Render함수를 호출합니다.

오브젝트들의 충돌처리, 수명을 관리하고 지워야한다면 지웁니다.

키보드, 마우스 입력에 따른 이벤트를 처리하고 서버에 보내야 한다면 Network객체에 알려줍니다.

* Object 클래스

속도, 이동방향, 위치를 기억해두고 Update시 흐른 시간만큼 위치를 갱신합니다.

자신이 어떤 텍스쳐인지 기억해두고 Render시 해당 텍스처로 그립니다.

자신의 공격력, 체력을 기억해두고 체력이 다하면 죽었다는 걸 SceneManager에게 알려줍니다.

* Network 클래스

Initialize함수에서 윈도우 소켓을 초기화하고 서버에 Connect함수로 접속합니다. 접속에 성공했다면 모든 플레이어가 접속할 때까지 대기합니다.

스레드 함수인 ReceiveProcess는 서버로부터 SIGNAL 메시지를 기다립니다. SIGNAL 메시지가 들어왔다면 DispatchSignal로 보내줍니다.

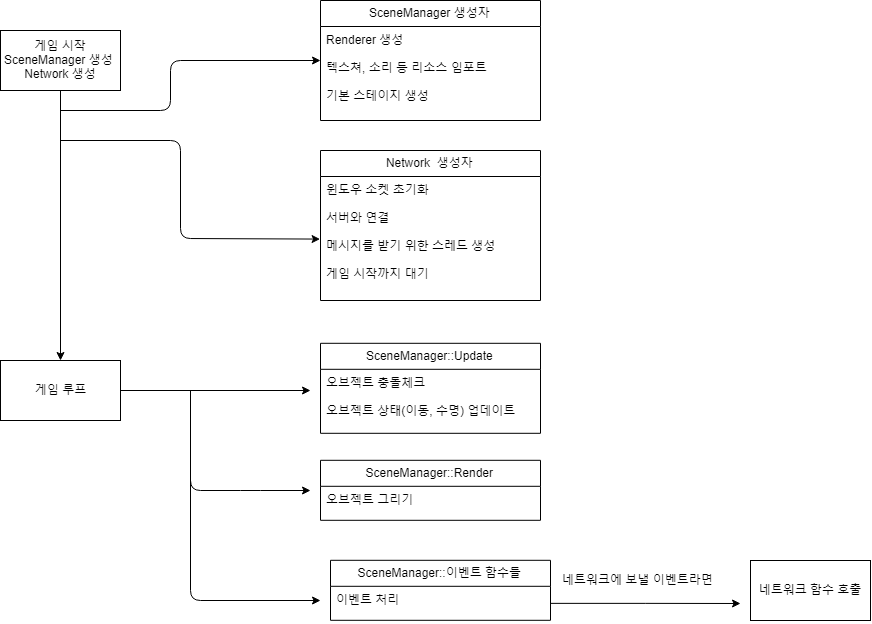
DIspatchSignal 함수는 SIGNAL상수별로 알맞은 함수를 호출합니다.

ReceiveSpawnData 함수는 서버로부터 유닛생성에 관한 메시지를 받아서 SceneMan ger에게 해당 위치에 유닛이 생성 되게합니다.

CreateObject 함수는 SceneManager에서 유닛이 생성될 때 호출됩니다. 현재 생성된 유닛의 정보를 서버로 SIGNAL\_SPAWN과 함께 보냅니다.

KeyInput 함수는 키보드 이벤트가 발생해 플레이어가 움직일 때 호출됩니다. 움직인 정보를 서버로 SIGNAL\_MOVE와 함께 보냅니다.

ReceiveMoveData 함수는 건물의 움직임을 서버로부터 받을 때 호출됩니다. SceneManager에서 건물을 움직입니다.

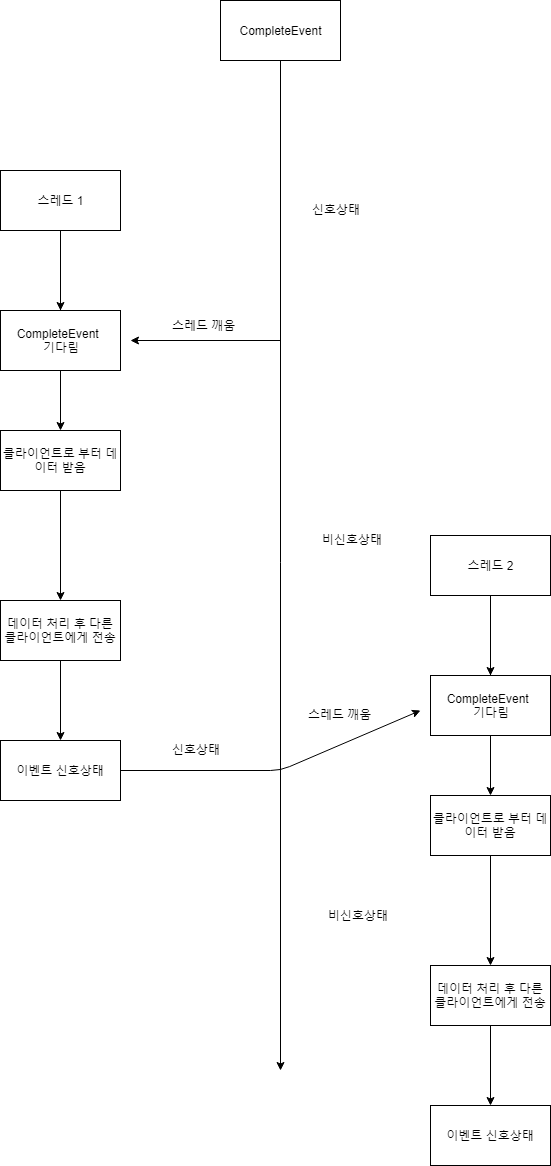


* 동기화 부분
* ~~연결된 클라이언트 소켓과 id가 들어있는 배열~~
* 연결된 클라이언트 소켓과 id가 들어있는 unordered\_map

모든 클라이언트가 접속할 때까지 스레드를 정지시켰다가 다 접속하면 실행합니다.

* ~~스레드간 기아상태 해결~~

~~CompleteEvent를 둬서 한 클라이언트 스레드가 자료를 받아서 전송까지 완료하면 이벤트를 신호 상태로 바꿔 다른 클라이언트 스레드가 깨어나게 할 계획입니다.~~



**5. 역할분담**

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 내용 |  | 담당 |
| 서버 | 서버 프레임워크 제작 | Main에 구현 | 양태윤 |
| 멀티 스레드 구현 | ClientProcess | 이상호 |
| 동기화 작업 | 이벤트 사용 | 강래민 |
| 메시지 분류 | DispatchSignal | 강래민 |
| 오브젝트 생성 처리 | ProcessSpawn | 양태윤 |
| 접속 종료 처리 | ProcessExit | 이상호 |
| 랭킹 시스템 |  | 강래민 |
| 키보드 입력 | ProcessKeyboardInput | 양태윤 |
| 소켓옵션 |  | 강래민 |
| 플레이어 접속 | ProcessStart | 이상호 |

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 내용 |  | 담당 |
| 클라이언트 | Network클래스 만들기 | Network | 양태윤 |
| 클라이언트 프레임워크 | SceneManager, Object | 양태윤 |
| 메시지 분류 | DispatchSignal | 강래민 |
| 오브젝트 생성 | CreateObject | 양태윤 |
| 랭킹 시스템 |  | 강래민 |
| 키보드 입력 | ProcessKeyboardInput | 양태윤 |
| 클라이언트 수신 함수 | ReceiveProcess | 양태윤 |
| 충돌체크 | CollisionCheck | 강래민 |
| 시작화면 제작 |  | 이상호 |

**6. 개발일정**

**1주차**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 월  10/24 | 화  10/23 | 수  10/24 | 목  10/25 | 금  10/26 | 토  10/27 | 일  10/28 |
| 양태윤 | 계획서  초안 논의 | | | 계획서  회의 | 계획서  회의/검수 |  |  |
| 강래민 |  |  |
| 이상호 |  |  |

**2주차**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 월  10/29 | 화  10/30 | 수  10/31 | 목  11/01 | 금  11/02 | 토  11/03 | 일  11/04 |
| 양태윤 | 계획서  제출 | 서버 프레임워크 구현 | | 진행 상황 보고 및 보완 회의 | 클라이언트 프레임워크 제작(SceneManager) | |  |
| 강래민 | 클라이언트  리소스 수정 및  프레임워크 구현 | | 클라이언트  충돌처리 구현  CollisionCheck() | |  |
| 이상호 |  | |  |

**3주차**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 월  11/05 | 화  11/06 | 수  11/07 | 목  11/08 | 금  11/09 | 토  11/10 | 일  11/11 |
| 양태윤 | 진행 상황 보고 및 보완 회의 | 클라이언트 프레임워크 제작(Object) | | 진행 상황 보고 및 보완 회의 | 클라이언트 수신 구현  ReceiveProcess() | |  |
| 강래민 | 클라 랭킹시스템 구현 | | 서버 랭킹시스템 구현 | |  |
| 이상호 | 서버 스레드 구현  ClientProcess() | | 플레이어 접속 구현 | |  |

**4주차**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 월  11/12 | 화  11/13 | 수  11/14 | 목  11/15 | 금  11/16 | 토  11/17 | 일  11/18 |
| 양태윤 | 진행 상황 보고 및 보완 회의 | 클라 오브젝트 생성  CreateObject() | | 진행 상황 보고 및 보완 회의 | 서버 오브젝트생성  ProcessSpawn() | |  |
| 강래민 | 클라 메시지 분류 구현  DispatchSignal() | | 서버 메시지 분류 구현  DispatchSignal() | |  |
| 이상호 | 게임 시작 화면 제작 | | 게임 종료 처리 구현  ProcessExit() | |  |

**5주차**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 월  11/19 | 화  11/20 | 수  11/21 | 목  11/22 | 금  11/23 | 토  11/24 | 일  11/25 |
| 양태윤 | 진행 상황 보고 및 보완 회의 | 클라 키보드 입력 구현 | | 진행 상황 보고 및 보완 회의 | 서버 키보드 입력 구현 | |  |
| 강래민 | 서버 동기화 처리  (이벤트를 활용한 동기화) | | 서버 동기화 처리  (스레드간 기아현상이 안 발생하는지 확인) | |  |
| 이상호 | 소켓 옵션 검토  (사용하면 좋은 소켓 옵션이 있는지 확인) | | 클라 동기화 처리  (유닛 생성, 충돌체크에서 차이가 없는지 확인) | |  |

**6주차**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 월  11/26 | 화  11/27 | 수  11/28 | 목  11/29 | 금  11/30 | 토  12/01 | 일  12/02 |
| 양태윤 | 진행 상황 보고 및 보완 회의 | 테스트 및 디버깅 | | 진행 상황 보고 및 보완 회의 |  | 테스트 및  미비점 보완 | |
| 강래민 | 테스트 및 디버깅 | |  |
| 이상호 | 테스트 및 디버깅 | |  |

**7주차**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 월  12/03 | 화  12/04 | 수  12/05 | 목  12/06 | 금  12/07 | 토  12/08 | 일  12/09 |
| 양태윤 | 사전 검사 | 사전 검사 내용을 바탕으로 버그 및 오류 수정 | | | | | |
| 강래민 |
| 이상호 |

**8주차**

|  |  |
| --- | --- |
|  | 월  12/10 |
| 양태윤 | 최종제출 |
| 강래민 |
| 이상호 |

**작업일지**

|  |  |
| --- | --- |
| **2주차** | |
| **역할** | **양태윤** |
| * 서버 프레임워크 구현 * 클라이언트 프레임워크 구현 | |
| **진행상황** | |
| * 프레임워크 제작 완료 * 서버의 경우 간단하므로 클래스없이 함수로만 작성 * 클라이언트에 SceneManager , Network 클래스 구현 | |
| **보완점 및 계획** | |
| * 간단한 메시지를 주고받는 에코서버를 만들어서 값이 정상적으로 교환되는지 확인 예정. | |
| **역할** | **강래민** |
| * 클라이언트 리소스 수정 및 프레임워크 구현 * 클라이언트 충돌처리 구현 | |
| **진행상황** | |
| * AABB 충돌체크를 이용해 오브젝트 간 충돌검사 * 사운드, 이미지 리소스 찾기 | |
| **보완점 및 계획** | |
| * 서버, 클라이언트 랭킹 시스템 구현 예정. | |
| **역할** | **이상호** |
| * 클라이언트 리소스 수정 및 프레임워크 구현 | |
| **진행상황** | |
| * 사운드, 스프라이트 이미지 등의 리소스를 찾고 적용 * 이미지의 투명도 처리 | |
| **보완점 및 계획** | |
| * 서버 프레임워크가 마무리 되었으므로 서버에 스레드함수를 구현하고 테스트 해볼 예정. | |

|  |  |
| --- | --- |
| **3주차** | |
| **역할** | **양태윤** |
| * 클라이언트 메시지 스레드 구현 * 클라이언트 프레임워크 구현 | |
| **진행상황** | |
| * 클라이언트에 Object, Bullet, Player 클래스 구현 * 데이터를 받는 스레드를 생성해서 게임 업데이트와는 별개로 동작하도록 구현 | |
| **보완점 및 계획** | |
| * 간단한 메시지가 아닌 실제 유닛 생성 데이터를 보내고 이를 받아서 생성할 예정. | |
| **역할** | **강래민** |
| * 서버, 클라이언트 랭킹 시스템 구현 | |
| **진행상황** | |
| * ProcessExit(), InitRankData(), RecordRankData() 구현. * 게임 종료 시 텍스트 파일에 클라이언트 접속 IP와 함께 점수 저장 확인. * 기존 점수와 추가된 점수를 비교해 상위 3개의 점수만 저장. | |
| **보완점 및 계획** | |
| * 서버, 클라이언트 메시지(Signal) 분류 구현 예정 | |
| **역할** | **이상호** |
| * 서버 스레드 구현 및 클라이언트 접속처리 | |
| **진행상황** | |
| * 서버 스레드 구현하여 클라이언트 수동 접속 가능(팀 수동입력) * 플레이어 접속을 자동(팀 수동입력x)으로 처리하려 했으나 노트북 고장(전치 1주일)로 인한 작업 일시중지상태 | |
| **보완점 및 계획** | |
| * 노트북 수리 후 자동접속과 시작화면 전환을 동시에 시행할 예정 * 자동 접속기능은 스레드가 생성 되면 팀을 순서대로 서버가 지정하는 방식을 이용할 것이기에 오랜 시간이 걸리지않을 것으로 예상 | |

|  |  |
| --- | --- |
| **4주차** | |
| **역할** | **양태윤** |
| * 클라이언트 오브젝트 생성 * 서버 오브젝트 생성 | |
| **진행상황** | |
| * 클라이언트에서 생성시 좌표와 방향, 종류, 팀 정보를 보냄. * 서버에서 그 데이터를 받아 연결된 클라에게 다시 전송 | |
| **보완점 및 계획** | |
| * 오브젝트 생성 및 동기화 확인 * 키보드 동기화 구현 예정 | |
| **역할** | **강래민** |
| * 서버, 클라이언트 메시지(Signal) 분류 구현 | |
| **진행상황** | |
| * 몬스터 생성 메시지(Signal)를 클라이언트에서 서버로 전송 시 생성 * 게임 종료 메시지(Signal)를 클라이언트에서 서버로 전송 시 종료 및 점수 저장 | |
| **보완점 및 계획** | |
| * 서버 동기화 처리 구현 예정(이벤트 기법 활용 : 스레드 기아상태 방지) | |
| **역할** | **이상호** |
| * 게임 시작화면 제작 * ProcessExit() 제작 | |
| **진행상황** | |
| * 미뤄졌던 자동접속 제작 * 게임yoon의 대기화면은 아직 미정 * 원활한 결과처리와 종료프로세스 제작 | |
| **보완점 및 계획** | |
| * 노트북 고장으로 인해 진행제작이 한 단계씩 미뤄지고 있음 빠른 제작 요구 | |

|  |  |
| --- | --- |
| **5주차** | |
| **역할** | **양태윤** |
| * 클라이언트 키보드 입력 구현 * 서버 키보드 입력 구현 | |
| **진행상황** | |
| * 클라이언트에서 키보드 입력 시 데이터를 서버에게 보냄. * 서버에서 그 데이터를 받아 연결된 클라이언트에게 다시 전송 | |
| **보완점 및 계획** | |
| * 키보드 이동 및 동기화 확인 * 클라이언트 버그 확인 및 수정 예정 | |
| **역할** | **강래민** |
| * 서버 동기화 처리 구현 예정(이벤트 기법 활용 : 스레드 기아상태 방지) | |
| **진행상황** | |
| * 스레드의 기아 상태가 발견되지 않음. | |
| **보완점 및 계획** | |
| * 클라이언트 버그 확인 및 수정 예정 * 서버 버그 확인 및 수정 예정 | |
| **역할** | **이상호** |
| * 게임 시작화면 제작 * 소켓옵션 검토 | |
| **진행상황** | |
| * 게임 시작화면 제작 * Ioctlsocket(논블로킹) | |
| **보완점 및 계획** | |
| * 게임 종료화면 제작 필요 | |

|  |  |
| --- | --- |
| **6주차** | |
| **역할** | **양태윤** |
| * 버그 수정 및 테스트 | |
| **진행상황** | |
| * 클라이언트 텍스쳐 보관 방식을 변경 * 서버가 enter키를 누르면 정상 종료하도록 변경 * 서버에 키 입력을 위한 스레드 생성 * 서버가 플레이어가 여러명 들어와도 수용하도록 변경 | |
| **보완점 및 계획** | |
| * 클라이언트 버그 확인 및 수정 예정 * 서버 버그 확인 및 수정 예정 | |
| **역할** | **강래민** |
| * 클라이언트 버그 확인 및 수정 예정 * 서버 버그 확인 및 수정 예정 | |
| **진행상황** | |
| * 클라이언트 및 서버에서 닉네임 송수신 버그 수정 | |
| **보완점 및 계획** | |
| * 프로젝트 테스트 및 디버깅 | |
| **역할** | **이상호** |
| * 클라이언트 동기화 확인작업 | |
| **진행상황** | |
| * 게임시작시간이 다를 시 먼저 생성된 유닛이 다른 플레이어에게 안보이는 현상 발견 * 건물움직임의 싱크가 맞지 않음을 발견 | |
| **보완점 및 계획** | |
| * 게임시작시간을 양플레이어 동시에 하도록 변경 요망 * 건물움직임 싱크 맞춰야 함 | |

|  |  |
| --- | --- |
| **7주차** | |
| **역할** | **양태윤** |
| * 버그 수정 및 테스트 | |
| **진행상황** | |
| * 키 입력 시 누르지 않아도 움직이는 버그 수정 * 랭킹에서 이름이 제대로 들어가지 않는 버그 수정 * 서버에서 소켓을 보관하는 방식을 unordered\_map 방식으로 변경 | |
| **보완점 및 계획** | |
| * 최종 제출 전 확인 | |
| **역할** | **강래민** |
| * 프로젝트 테스트 및 디버깅 | |
| **진행상황** | |
| * 게임 시작 순간을 동시에 시작 하도록 수정 | |
| **보완점 및 계획** | |
| * 최종 제출 전 확인 | |
| **역할** | **이상호** |
| * 버그 수정 및 미비한점 보완 | |
| **진행상황** | |
| * 게임 종료화면 제작 * 건물이 벽을 뚫고 나가는 점을 보완 | |
| **보완점 및 계획** | |
| * 최종 검사까지 테스트 | |