종합설계기획

프로젝트 제안서

지도교수님 : 이형구 교수님

2013184019 유재용

**목차**

*1. 게임설명*

*a. 애플리케이션 소개*

*b. 개발환경*

*2. 개발 계획(서버)*

a. 프로토콜

b. 앱 전반적인 서버 통신

c. 통신 외 서버의 역할

*3. 개발 계획(서버 이외의 요소)*

a. 캐릭터(원시 종족)

b. 사운드

*4. 일정 및 역할분담*

a. 개발순서

b. 도표

1. 게임설명

a. 애플리케이션 소개



<우리 게임의 모티브가 될 오버워치 인게임 화면>

**출처 : http://www.sporbiz.co.kr/news/articleView.html?idxno=33944**

게임은 **1인칭 FPS** 장르로, 게임 진행은 ‘블리자드’사의 ‘**오버워치’를 모티브**로 하고 있다. **세 개의 우주 종족의 싸움**이라는 세계관으로 시작해 우주의 한 행성이 연상되는 맵에서 각 종족의 특성을 살려 싸우게 된다. 세 종족은 우주인, 원시인, 인간(아직은 가제인 상황)이고 주변의 엄폐물을 사용하거나 스킬 사용으로 상대에게 먼저 잡히지 않고 적을 처치해야한다. 게임 승리는 총 킬 수를 계산해서 결정한다.

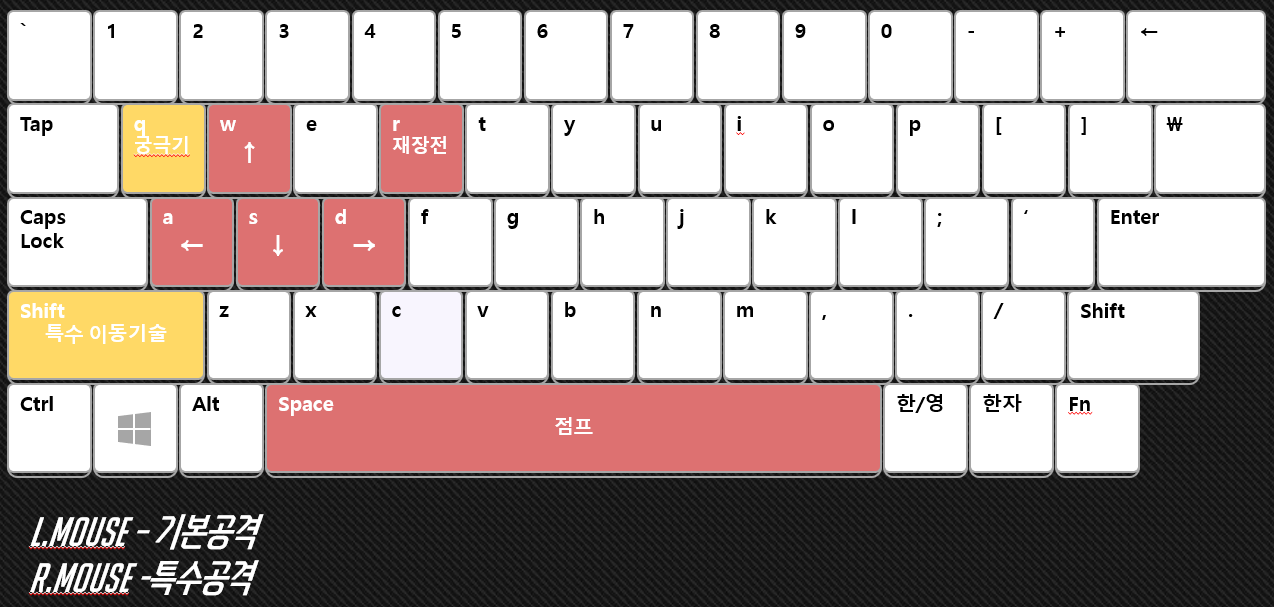


<자기장에 의해 맵이 제한됐는데 적응하지 못하고 상대방이 자멸했다.>

**출처: http://blog.naver.com/wjs02057/221075379381**

**게임 일정 시간마다 맵의 환경이 변화**하게 되는데 이 부분은 ‘블루홀’사의 ‘**배틀그라운드**’에서 힌트를 얻었다. 배틀그라운드는 오픈월드 수준의 방대한 넓이의 맵에서, 유저들의 게임 진행을 강요하기 위해 살아있는 유저수가 줄어듦에 따라 유저가 생존할 수 있는 맵의 넓이를 점차 줄여간다. 그 예로 ‘자기장 지역’이 있다. 자기장 지역이 점차 넓어지고, 그러므로 유저는 맵을 좁게써야 하고, 결국에는 상대방을 맞닥뜨려 최후의 1인이 남게 된다.

우리의 게임은 그다지 넓지 않은 맵을 제공하진 않을 것이지만 **게임 시간에 따라 다른 전술**을 사용해야 하고 **변화하는 환경에 따라 강세를 보이는 종족 또한 변화**하도록 환경변수를 구현하고자 한다.

조작법

C:\Users\woolo_so5omoy\AppData\Local\Microsoft\Windows\INetCache\Content.Word\이동.png: W,A,S,D

스킬 및 공격 : Shift, L.Mouse, R.Mouse, Q

점프 : SPACE BAR

재장전 : R

b. 개발환경



마이크로소프트 비주얼 스튜디오 2017

DirectX 12

그래픽 툴 미정

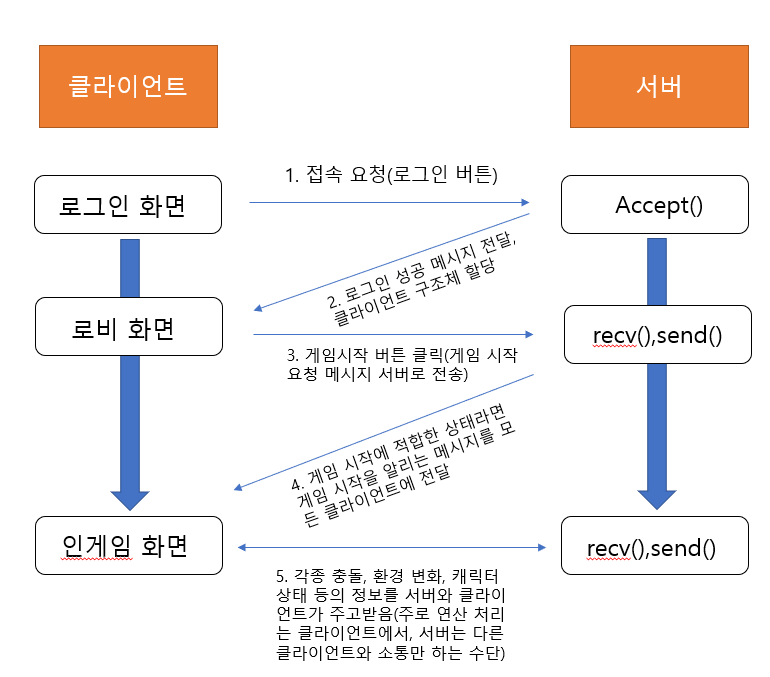
플랫폼 : PC

2. 개발 계획

a. 프로토콜

IOCP를 사용한 다중 클라이언트를 지원하는 서버 프로그램을 구현할 것이다. 특히 fps장르의 게임은 피격이나 이동에 매우 민감하기 때문에 신뢰성이 보장되는 프로토콜을 채택할 것이다.

b. 앱 전반적인 서버 통신



1. **접속 요청**

클라이언트가 서버 아이피와 본인이 게임 내에서 사용할 아이디를 입력하면 일단 **소켓 연결**을 하고, 서버에서는 지금 접속 요청을 한 클라이언트와 이미 접속해 있는 클라이언트들과 **아이디 중복체크**를 한다.

1. **로그인 성공 메시지 전달, 클라이언트 구조체 생성**

문제가 없다면 **다음 씬(로비 씬)으로 이동할 수 있는 메시지를 전달**한다. 그리고 서버에는 모든 클라이언트의 정보를 저장해 둘 것인데, 서버에 새로 접속한 **클라이언트의 정보를 위한 구조체를 하나 더 추가**한다. 이 구조체는 게임 내내 사용하게 된다.

1. **게임시작버튼 클릭**

어느정도 게임 진행이 가능한 인원이 모이면 게임 시작 버튼을 누르게 될 것이다. 방장 클라이언트가 게임시작버튼을 누르면 서버로 게임시작요청 메시지를 전달한다. 이 때 방장 클라이언트를 식별하는 방법은 위에서 생성한 구조체 안의 bool형 타입 변수를 사용할 것이다.(ex, bool isGameMaster = true;)

1. **적합한 상태일 때 게임 시작을 알리는 메시지 모든 클라이언트에 전달**

이 때 적합한 상태란, 어느정도 게임 진행이 가능한 유저 수의 존재 여부, 모든 클라이언트가 레디 상태인지 등이 있겠다. 이 모든 정보도 구조체에 저장이 되게 되며 검사 후 문제가 없으면 모든 **클라이언트에 게임 시작을 알려**준다.

1. **게임 플레이 중의 통신들**

게임이 시작되면 객체 간 충돌, 캐릭터와 탄(무기)의 충돌, 캐릭터들의 상태 변화 등 다양한 메시지를 서버와 클라이언트가 통신하게 될 것이다. **서버는 충돌이나 상태들을 ‘감지’**만 해주고 감지결과를 클라이언트들에게 뿌려주면 **그 결과에 대한 처리(ex,탄을 맞으면 경직처리를 해준다)들은 모두 클라이언트에서** 한다.

c. 통신 외 서버의 역할

서버는 게임 시작과 동시에 **게임 플레이 시간을 계산**하여야 한다. 그 시간을 아이템 재생성, 게임 제한시간 계산이나 우리 게임의 특징인 환경변화에 사용할 것이다. 이러한 것들은 클라이언트마다 **조금이라도 상이하면 안되는 정보들이기 때문**에 서버에서 계산하고 클라이언트들에 뿌려준다. 클라이언트상의 계획대로라면 중력의 변화는 1분마다 일어나는데, 그 것이 조금이라도 어긋나면 엄청난 문제를 유발할 것이다.

3. 개발 계획(서버 이외의 요소)

a. 캐릭터(원시 종족)

우리 팀은 팀원 마다 각자 맡은 캐릭터 하나 씩을 구상하기로 했다. 우리 게임에 등장하는 인간, 외계인, 원시 종족 중 나는 원시 종족을 맡는다. 

<원시 종족의 모티브가 된 스타크래프트의 ZERG종족>

**출처: http://starcraft.wikia.com/wiki/File:Hydralisk\_SC2\_Cncpt4.jpg**

**캐릭터 : 원시종족**

이 캐릭터는 가장 발전이 늦다는 컨셉에 의해 무기가 아닌 자기가 직접 공격하는 모션을 부여한다.

**크기 : 0.7M X 2M (가로 X 세로)**

**체력 : 200**

**기본 공격 : 근접공격**

자기가 직접 타격하기 때문에 다른 캐릭터와 다르게 근접 공격을 구사한다. 근접 공격인 만큼 다른 종족보다 강한 공격력을 가진다.(**공격력 : 25**)

**특수 공격 : 근접공격**

조금 더 강하고 느린 근접 공격이다. 투명화 상태일 때는 공격을 성공시키기 쉽기 때문에 충분히 느리게 설정한다(**공격력 : 50**)

**이동기 : 돌진**

원시종족의 투박한 느낌을 살리기 위해 무작정 돌진하는 느낌을 구현할 것이다. **(이동 거리 : 10m)**

**궁극기 : 투명화**

일정 시간(약 **10초** 예상) 투명화가 되어 상대에게 접근이 쉬운 상태가 된다. 투명상태지만 상대의 공 격에는 그대로 피격당한다. 또한 캐릭터가 **첫 공격 모션을 취하면 즉시 투명상태가 풀린다**.



<원시종족의 궁극기인 투명화의 모티브가 된 LOL챔피언 ‘카직스’의 은신>

**출처: https://m.blog.naver.com/PostView.nhn?blogId=qutngu4080&logNo=70155166608&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F**

위의 캐릭터는 클라이언트의 진행에 따라 병렬로 구현을 시작할 것이다. Character를 담는 슈퍼클래스의 구현이 클라이언트 쪽에서 완성이 되면 그것을 상속받아 각자의 캐릭터를 구현한다.

b. 사운드

사운드 같은 리소스들은 공동 작업을 전제로 한다. 주로 애셋스토어나 무료 리소스를 사용할 예정이고 리소스 선별과 소스에 적용하는 과정은 모두가 함께 한다.

3. 개발일정

a. 개발순서

1. **패킷 디자인**

서버와 클라이언트들 사이에서 **통신에 사용할 메시지들, 구조체들에 관한 정의**를 한다.이는 클라이언트 개발 전에 클라이언트 내에서 서버통신에 사용할 정보들을 미리 파악 해 놓아야 하므로 가장 먼저 해야 할 일로 꼽았다.

1. **서버 구조(프레임워크) 제작**

서버는 클라이언트에서 넘어오는 각종 메시지와 구조체에 따른 적절한 처리를 해야 하기 때문에, **메시지만 읽어도 적절한 함수를 수행할 수 있도록 디자인** 할 것이다. 또 우리 게임은 서버 시간에 따른 처리가 필요하기 때문에 **연산용 스레드를 따로 생성** 해 그 것을 전담시킨다.

1. **클라이언트와 서버 연동**

위에 제작한 서버와 어느정도 개발이 진행된 **클라이언트를 연동**시킨다. 클라이언트 개발단계에서 패킷에 사용할 정보들을 어느정도 정리 해놨기 때문에 서버 프레임워크가 잘 작동한다면 다양한 메시지로의 응용은 그리 어렵지 않을 것으로 예상된다.

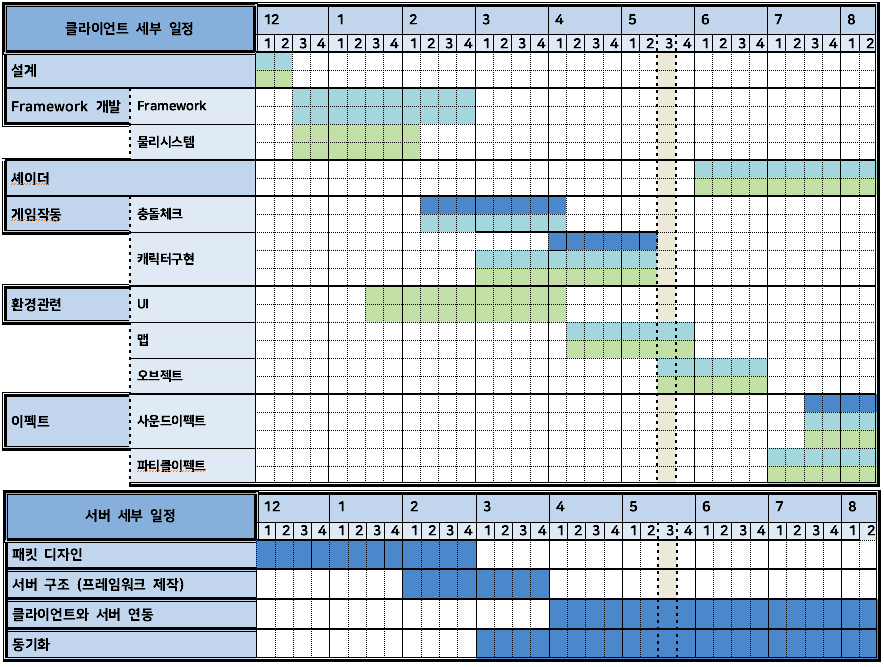
1. **충돌체크**

**객체 간의 충돌을 감지하는 소스를 구현**한다. 서버는 객체의 좌표 값을 일정 시간마다 전달받고, **가상의 게임을 서버에서도 구동**한다는 느낌으로 서버에서 충돌 감지를 할 것이다. 충돌이 감지되면 **모든 클라이언트에 충돌 후속처리에 관한 적절한 메시지**를 전달한다.

1. **동기화**

많은 클라이언트 스레드가 접속하다 보면, 모두가 접근할 수 있는 데이터에 한 번에 접근해서 의도치 않은 결과가 발생할 수 있다. 그것에 대한 **임계 처리를 구현**한다. 동기화는 모든 제작 과정에서 처리가 안 되어 있으면 **문제가 발생한다고 예상될 때마다 구현**할 것이다.

b. 구현 일정 도표



위와 같은 계획으로 구현을 시작할 것이고 기획이 수정될 때마다 수정을 해야하는 사유와 수정 내용을 기록한다.