네트워크 게임 프로그래밍

Term Project 추진 계획서

게임공학과

(PM)2012180022 양기석

2012181042 황성섭

2016-12-06 (화)

\ 목 차

1. 게임 소개……………………………………………………………………………………………………………………………………….

게임 제목

게임 장르

게임 설명

게임 조작법

게임 흐름

게임 스크린 샷 / 예상도

1. 요구 사항 분석………………………………………………………………………………………………………………………………

기능 요구사항

1. High-level 디자인…………………………………………………………………………………………………………………………..

클라이언트 흐름도

서버 흐름도

스레드 동기화 방식

1. Low-level 디자인……………………. …………………………………………………………………………………………………….

클라이언트 구조체 / 함수

서버 구조체 / 함수

1. 팀원 역할분담……………………….. . …………………………………………………………………………………………………….

양기석

황성섭

1. 개발환경…………………………………. ……………………………………………………………………………………………………..
2. 개발일정…………………………………. ……………………………………………………………………………………………………..
3. **게임 소개**

**\*게임 제목** - Try Again

**\*게임 장르** – RPG, 로그라이크

**\*게임 설명**

이 게임은 MMORPG 게임들의 주요 컨텐츠이기도 한 보스 레이드만을 다룬다.

소수의 인원이 여러 패턴이 존재하는 하나의 보스를 서로 부족한 점을 보안하며

협동하여 물리치는 게임이다.

**최소 1인 ~ 최대 4인**까지 플레이가 가능하고, 각각의 플레이어는 2가지 직업( **궁수**와 **사제** )을

선택하여 플레이 할 수 있다.

각 플레이어는 총 3가지의 스킬을 사용 할 수 있고, 스킬은 모두 논-타겟팅 발사 스킬이다.

각각의 직업은 서로 다른 3가지 스킬을 가지며 스킬별로 고유한 쿨타임을 가진다.

첫 번째 스킬은 마우스를 방향으로 날아가는 약한 공격or회복 스킬이며, 1초의 쿨타임을 가진다.

~~두 번째 스킬은 짧은시간 이동속도가 증가하는 스킬이며 15초의 쿨타임을 가진다.~~

~~세 번째 스킬은 마우스를 클릭한 방향으로 간단한 패턴을 그리며 날아가는 강한 공격or회복 스킬이며, 20초의 쿨타임을 가진다.~~

**궁수**는 공격을 전담하며 보스에게 스킬을 이용해 데미지를 줄 수 있다.

**사제**는 아군에게 스킬을 맞춤으로써, 아군의 HP를 올릴 수 있다.

**보스**와 플레이어는 적당한 체력을 보유하고 있으며, 체력이 0이하가 되면 사망한다.

**~~보스~~**~~는 총 2마리가 존재하고, 각각 여러 패턴의 탄막을 뿌리며 플레이어를 공격한다.~~

기존 모티브 게임이 탑 뷰인데 반해, 사이드 뷰 시점을 가지며, 2D로 제작하기로 한다.

또한, 맵 크기에 따라 ~~종/~~횡 스크롤이 가능하게 한다.

**\*게임 조작법**

WSAD 키로 상하좌우 점프를 할 수 있으며, 마우스 클릭으로 스킬을 사용 할 수 있다.

~~스킬은 1~3번으로 선택할 수 있다.~~

**\*게임 흐름**

서버의 IP, 포트 번호 입력

**대기룸**

게임 시작

**게임진행**

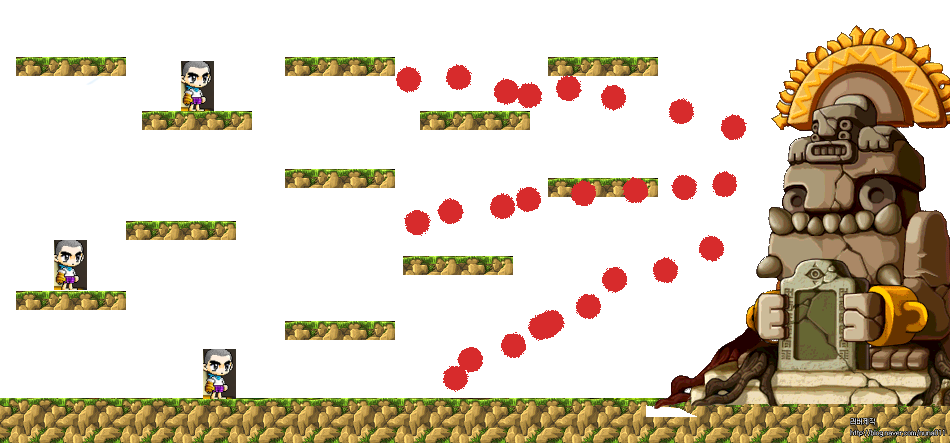
패배

승리

**게임종료**

**\*게임 스크린 샷**

보스

****

탄막

유저1

유저2

유저3

그림1. 게임 예상도

(출처: http://blog.daum.net/\_blog/ArticleCateList.do?blogid=0aMmK&CATEGORYID=37&dispkind=B2201)

1. **요구 사항 분석**

**\*기능 요구 사항**

-네트워크 방식은 TCP로 한다.

-게임 시작을 누르면 서버의 IP, 포트번호를 입력 할 수 있다.

-서버는 최대 4개의 클라이언트를 수용할 수 있다.

-서버에 접속을 성공한 클라이언트는 대기 씬으로 이동한다.

-대기 씬에서는 캐릭터 이동과 점프가 가능하다.

-대기 씬에서 서버에 접속한 모든 플레이어의 행동이 서로에게 실시간으로 전달된다.

-대기 씬에서 처음 접속한 클라이언트가 방장이 되며, 방장이 게임을 시작할 수 있다.

-방장이 게임을 시작하면 서버에 접속한 클라이언트들은 같은 게임 씬으로 이동한다.

-각 클라이언트는 맵 스크롤시 자신의 캐릭터를 중심으로 보는 (카메라)뷰를 가지며,

~~고정된 충돌체인 맵 과의 충돌처리는 맵 데이터를 서버에서 한번만 로딩한 뒤,~~

~~클라이언트에서 처리한다.~~

맵 정보 또한 클라이언트가 가진다.

-클라이언트에 공통적으로 사용 될 정보(보스 탄막, 보스, 맵 등)은 서버에서 관리하며,

변경 될 여지가 있는 정보의 충돌처리(캐릭터 - 탄막 등)는 서버에서 처리하고 각 클라이언트에게 알려준다.

- 멀티스레드를 이용한 이벤트 처리로 동기화 과정을 거친다.

- 스레드 동기화시의 문제점을 해결해야 한다.

클라이언트 오류로 데이터를 보내지 못하면 무한 대기 상태에 빠질 수 있다.

따라서, 한 스레드가 일정 시간 동안 데이터를 받아오지 못하면, 다음 스레드로

넘어가는 방법을 생각 할 수 있다.

- 서버의 갱신 속도가 일정하지 않으면 문제점이 생길 수 있다.

Ex)네트워크로 연결 된 타 클라이언트 캐릭터의 움직임이 부자연스러운 움직임을 보인다.

-> 데이터를 처리 할 때 최대 처리의 시간을 생각하여 처리할 데이터가

적을 때에도 평균 처리 시간만큼을 지연시킨다면 조금 더 자연스러운

연출을 기대할 수 있다.

1. **High Level 디자인**

**\*클라이언트 흐름도**

StartScene

DialogBox 입력처리

False

getSceneManeger()

True

WaitScene

False

방장의 ‘R’키 입력

방

True

sendToServer()

GameScene

False

recvFromServer()

Server

IsGameOver()

True

엔딩 씬

**\*서버 흐름도**

시작

접속 대기

접속한 클라이언트 수만큼 g\_Client가

증가 한다.

totalServerInfo.nowClient=g\_Client로

유저 번호를 할당한다.

쓰레드 할당

g\_Client==1 방장

sendToClient()

WaitScene

Client

방장의 ‘R’키 입력

Start 체크

방

recvFromClient()

sendToClient()

GameScene

True

Client

False

recvFromClient()

게임오버

엔딩 씬

**\* 스레드 동기화 방식**

**- g\_nTreated ( 처리한 스레드 수 ) = 0;**

**-Thread Function-**

공유 자원

이벤트 처리

recvFromClient**(** ,

**clientInfo[g\_nTreated]**  ,…**)**

**~~g\_nTreated++;~~**

**//서버에서의 데이터 처리**

**TreatedByServerInfo**

**( clientInfo )**

쓰레드2

쓰레드1

서버의 deltaTime 계산

~~TimeOut 시간 = 최대 4~ ms~~

~~총 최대 16~ ms 이하~~

- ~~클라이언트의 60프레임을 기준으로 60/ 1000 / g\_Client(int) sec 의 최대 스레드 대기시간을 가지며, TIMEOUT시 바로 다음 스레드가 실행 되게 한다.~~

- 데이터를 주고 받는 쓰레드2 에서 공유자원을 처리하는 부분~send() 까지 이벤트 처리를 한다.

- 서버 쓰레드1에서 약 30프레임의 델타 타임 값을 클라이언트에 전달해준다.

1. **Low Level 디자인**

**\*** **클라이언트 구조체 / 함수**

- **클라이언트 구조체**

~~서버로부터 처음에 받는 맵의 정보를 담는 구조체.~~

**~~Struct mapInfo {~~**

**~~Int \*\*first\_mapInfo;~~**

**~~Int \*\*second\_mapInfo;~~**

**~~}~~**

~~서버에서 처리한 데이터의 정보를 담는 구조체.~~

**~~Struct treatedByServerInfo {~~**

**~~Player \*player; //다른 플레이어의 정보~~**

**~~Boss \*boss; //보스의 정보~~**

**~~queue<Projectiles> projectiles; //스킬들의 정보~~**

**~~queue<Effect> effect; //이펙트의 정보~~**

**~~}~~**

서버에 보내고 받는 데이터의 정보를 담는 구조체.

**struct ClientInfo {**

**PlayerInfo player[MAX\_CLIENT]; //플레이어의 정보**

**int playerSize; // 플레이어 번호**

**ArcherSkillOneInfo arrow[MAX\_CLIENT][MAX\_ARROW\_SIZE]; // 아쳐스킬 구조체**

**int arrowSize[MAX\_CLIENT]; // 클라이언트당 화살 발사 수**

**BossInfo boss; // 보스 구조체**

**PattenInfo patten[MAX\_PATTEN][MAX\_BULLET]; // BULLET 패턴 구조체**

**int pattenSize[MAX\_PATTEN];**

**int myClientValue; //자신이 몇번째 클라이언트인가**

**int nowClient; // 현재 클라이언트의 번호**

**int maxClient; // 클라이언트 접속의 최대치**

**int scene; //무슨 씬인가**

**};**

플레이어 정보를 담는 구조체

**struct PlayerInfo {**

**Vector2D pos; // 플레이어 좌표 정보**

**Vector2D bitmapSize; // 이미지 크기**

**ArmInfo arm; // 팔(활)의 구조체**

**CollInfo collider; // 충돌 구조체**

**int hp; // 플레이어 체력**

**};**

팔(활)의 구조체

**struct ArmInfo{**

**Vector2D pos; // 팔의 좌표 정보**

**float angle; // 팔의 각도 정보**

**};**

플레이어 스킬의 구조체

**struct ArcherSkillOneInfo {**

**Vector2D pos; // 스킬 좌표 정보**

**Vector2D bitmapSize; 이미지 크기**

**float angle; //스킬의 각도 정보**

**CollInfo collider; // 충돌 구조체**

**int demage; // 데미지 정보**

**bool isColl; // 충돌 판단 변수**

**};**

보스의 구조체

**struct BossInfo {**

**Vector2D pos; // 보스 좌표 정보**

**Vector2D bitmapSize; // 이미지 크기**

**int hp; // 보스 체력**

**CollInfo collider; // 충돌 구조체**

**};**

BULLET 패턴 구조체

**struct PattenInfo {**

**Vector2D pos; // 패턴 좌표 정보**

**Vector2D bitmapSize; // 이미지 크기**

**CollInfo collider; // 충돌 구조체**

**int state; 패턴의 상태 정보**

**int demage; // 데미지 정보**

**bool isColl; // 충돌 판단 변수**

**};**

충돌 구조체

**struct CollInfo{**

**int type; // 충돌 처리 형태 정보 (BOX)**

**int obejctType; // 충돌의 대상 정보**

**RECT rt; // 충돌 사각형 콜라이더**

**Vector2D center; // 중점 좌표 정보**

**int radius; // 반지름 정보**

**};**

- **클라이언트 함수**

~~void~~ **~~IPPortInput~~**~~() - 게임시작시 IP/포트번호 입력을 한다.~~

~~bool~~ **~~ChangeScene( int SceneNum )~~**

~~return value : 씬 변경 성공여부~~

~~- 씬을 변경하는 함수~~

~~void~~ **~~sendCharacterChange(SOCKET s, BOOL IsChangedCharactor, int len, int flags)~~**

~~- 캐릭터 변경시 서버에 데이터를 보낸다.~~

~~void~~ **~~recvClientCharacterInfo(SOCKET s, treatedByServerInfo tbServerInfo~~**

**~~, int len, int flags)~~**

~~- 캐릭터들의 정보가 바뀌면 서버로부터 캐릭터들의 정보를 얻어온다.~~

~~bool~~ **~~IsChange() -~~** ~~클라이언트의 바뀐 정보가 있는지 여부 확인~~

~~bool~~ **~~IsGameOver() -~~** ~~게임 종료 조건 체크~~

~~bool~~ **~~StartGameMessageToServer( int sceneNum)~~**

~~자신이 방장 속성(bool IsChief==true)일 때 게임시작 버튼을 누를 경우~~

~~해당 정보를 변경.~~

void **recvFromServer(SOCKET s, int flags)**

return value : 에러 처리용 값

-서버로부터 데이터를 받는다.

bool **sendToServer(SOCKET s, ClientInfo clientInfo, int flag);**

return value : 에러 처리용 값

-고정-가변 길이의 데이터 전송방식으로 데이터를 보낸다.

bool **init\_sock();**

return value : 에러 처리용 값

-소켓 생성 후 서버와 연결을 한다.

**\*** **서버 구조체 / 함수**

**- 서버 구조체**

~~클라이언트에 처음에 보낼 맵의 정보를 담는 구조체.~~

**~~Struct mapInfo {~~**

**~~Int \*\*first\_mapInfo;~~**

**~~Int \*\*second\_mapInfo;~~**

**~~}~~**

~~서버에서 처리한 데이터의 정보를 담는 구조체.~~

**~~Struct TreatedByServerInfo~~**

서버로부터 클라이언트에게 데이터의 송수신을 위한 정보를 담은 구조체

**struct ClientInfo {**

**PlayerInfo player[MAX\_CLIENT]; //플레이어의 정보**

**int playerSize; // 플레이어 번호**

**ArcherSkillOneInfo arrow[MAX\_CLIENT][MAX\_ARROW\_SIZE]; // 아쳐스킬 구조체**

**int arrowSize[MAX\_CLIENT]; // 클라이언트당 화살 발사 수**

**BossInfo boss; // 보스 구조체**

**PattenInfo patten[MAX\_PATTEN][MAX\_BULLET]; // BULLET 패턴 구조체**

**int pattenSize[MAX\_PATTEN];**

**int myClientValue; //자신이 몇번째 클라이언트인가**

**int nowClient; // 현재 클라이언트의 번호**

**int maxClient; // 클라이언트 접속의 최대치**

**int scene; //무슨 씬인가**

* **내부 구조체의 내용은 collider함수를 제외하면 같다 –**

**(Collider() – 충돌박스 정보를 리턴)**

}

- 서버 함수

DWORD WINAPI **ProcessClient(LPVOID arg)**

-클라이언트로부터 데이터를 받아 정보 구조체에 처리 후 담는다.

-플레이어와 보스, 패턴의 충돌 작업이 진행 되며 데이터는 정보 구조체에 담는다.

-고정-가변 길이의 데이터 전송방식으로 클라이언트에게 데이터를 보낸다.

DWORD WINAPI **ProcessTime(LPVOID arg)**

-BULLET 패턴의 Update의 속도를 조절한다.

1. **팀원 역할분담**

\*양기석(PM) - 클라이언트 제작 ( 충돌 처리와 캐릭터 관련 개발 )

- 서버에서의 충돌 처리

- 주로 클라이언트의 네트워크 기능 구현.

\*황성섭 - 클라이언트 제작 ( 스킬과 보스 관련 개발 )

- 캐릭터와 보스의 탄막 패턴 정보 데이터 관리 및 보내기

- 주로 서버에서의 네트워크 기능 구현.

1. **개발환경**

Visual Studio 2013

Github (프로젝트 관리)

1. **개발 일정 (수정)**

**-엑셀 파일 첨부**