네트워크 게임 프로그래밍

최종 보고서

2013년 12월 12일

2010180020 박성경

2011180007 김대현

**애플리케이션 기획**

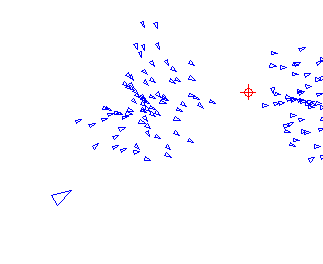
Mini MMORPG

: 쿼터뷰 방식의 RPG를 간단하게 구현해 본다

간단한 MMORPG 게임 예시

* Realm of the Mad god - 실제 플레이 구상 화면

(인공지능 수업 예제 프로그램)



**전체 기능**

**-- Client --**

기본적으로 플레이어가 몬스터를 잡고 레벨업을 하는 것

키보드 이동을 기본으로 한다.

몬스터 대신 플레이어를 공격 가능 (PvP)

- 키보드 -

w,a,s,d 키보드로 이동한다

- 마우스 -

왼클릭 : 클릭한 방향으로 공격한다

**-- Server –**

- Monster -

일정 시간이 지나면 Monster가 생성된다.

몬스터의 수는 총 20마리 이하가 되도록 자동적으로 생성된다

몬스터의 시야 사정거리 안쪽으로 왔을 경우에 몬스터가 플레이어를 따라간다

몬스터와 충돌 시 플레이어의 HP가 낮아진다

몬스터의 체력이 절반이 되면 도망간다

- Item –

일정 시간에 따라 10개 이하로 생성된다

**네트워크 기능**

-- 클라이언트 --

서버에서 전송 받은 데이터를 토대로 타 플레이어와 몬스터를 그려준다.

공격 시 서버에 공격했다는 정보를 전송해준다

피격 시 서버에서 데이터를 전송 받아 플레이어의 hp를 깎거나 죽는다.

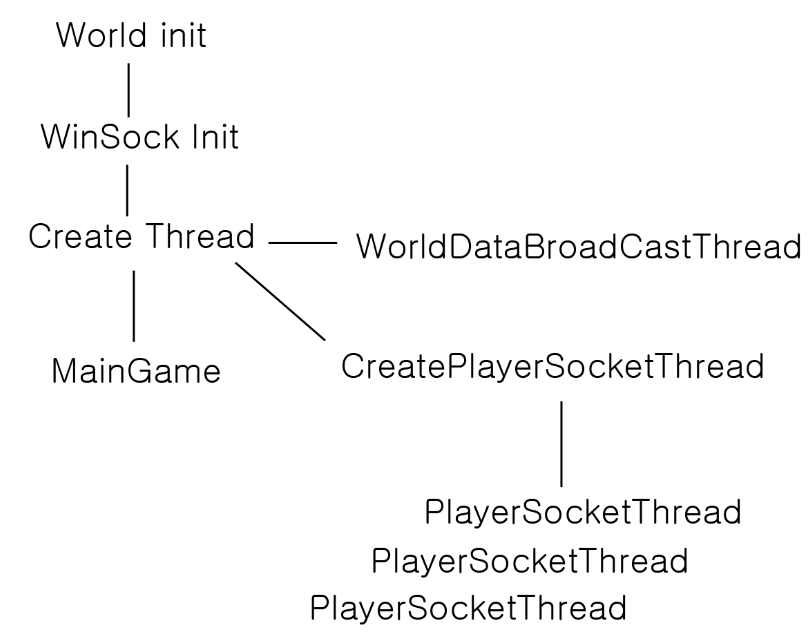
-- 서버 --

플레이어들의 상태를 관리한다

충돌체크가 된 플레이어들에게 공격 당했다는 메시지를 보낸다

클라이언트에게 업데이트된 몬스터와 플레이어와 아이템 데이터를 전송한다.

**High Level**



World Init : 게임 세계 초기화

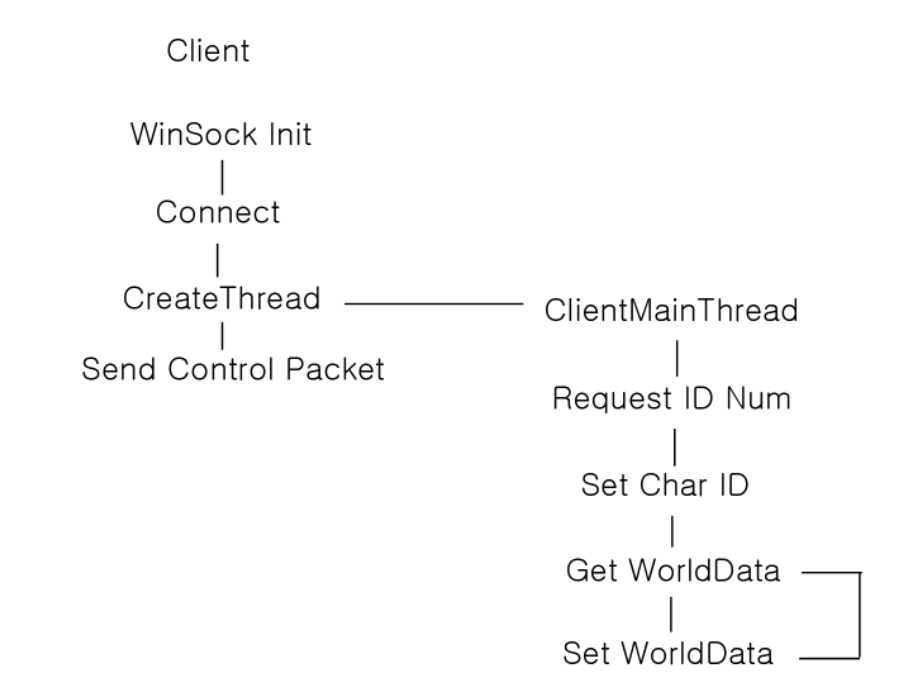
WinSock Init : 윈소켓 초기화

Thread Create : 서버에 필요한 쓰레드들 생성

WorldDataBroadCastThread : 현재 접속한 유저에게 WorldData를 전송 시킨다.

CreatePlayerSocketThread : Listening 상태로 접속 요청이 오면 PlayerSocket을 생성한다.

PlayerSoketThread : ClientSoket을 받아 처리하고 Player의 전송 패킷을 처리한다.



WinSock Init : 윈소켓 초기화

Connect : 서버와 연결

CreateThread : 서버패킷 처리 스레드 생성

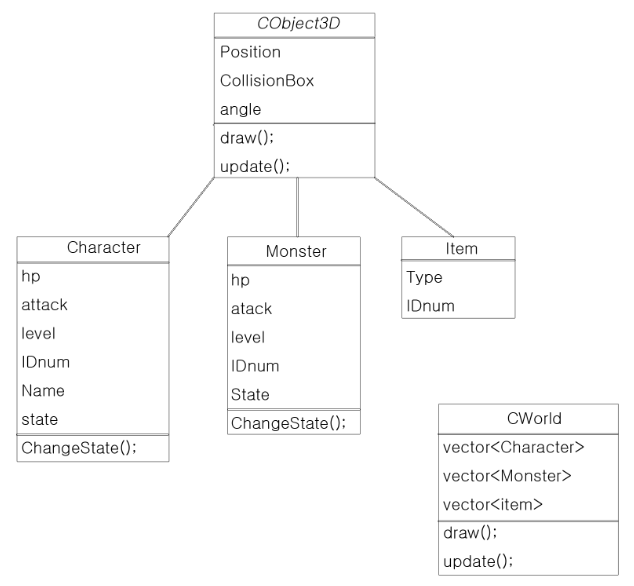
ClientMainThread : 서버로 패킷 전송 역할 스레드 생성

Set Char ID : 내 캐릭터를 식별할 수 있는 ID를 기억

Get WorldData : 서버로부터 WorldData를 받아온다.

Set WorldData : 받아온 WorldData를 클라이언트에 적용 시킨다.

**OBJECT**



CObject3D : 다른 객체들이 상속받을 추상화 클래스

CWorld : 객체 데이터들을 관리해줄 매니지먼트 클래스

***Low Level (Define, Structs, Funtions)***

**-- Item TypeNum --**

enum ItemType

HEAL 1000

~~BOMB 1001~~

~~SPDUP 1002~~

- 아이템 타입별 아이디

**-- Message Num --**

~~enum MSGtype~~

~~DESTROY 2000~~

~~RESPAWN 2001~~

~~TELEPORT 2002~~

~~- 플레이어에게 전송할 강제 상태바꿈 메세지~~

서버에서 처리해 줌으로써 필요 없어짐

**-- STRUCT --**

struct POINT

float x, y, z

- 게임내에 쓰일 점좌표

struct CollisionBox

struct POINT Min

struct POINT Max

- 충돌체크에 쓰일 구조체 데이터

struct WorldData {

int PlayerNum;

PlayerData playerInfo[10];

int MonsterNum;

MonsterData monsterInfo[20];

int ObjectNum;

WorldObjectData objectInfo[10];

int ItemNum;

ItemData itemInfo[10];

}

- 월드 데이터 구조체

- 몬스터, 플레이어의 레벨, 위치좌표와 아이템 위치좌표가 들어간다.

**-- 추가 STRUCT --**

Struct PlayerData {

POINT UserPosition;

POINT UserVelocity;

Int attackDir;

Int moveDir;

Int userHP;

Int userID;

Bool atkState;

};

Struct MonsterData {

POINT MobPosition;

Int MonsterHP;

};

Struct WorldObjectData {

Float angle;

};

Struct ItemData {

POSITION ItemPosition;

Int type;

};

**-- Functions --**

**-- Common --**

bool CollisionCheck(Struct CollisionBox, struct CollisionBox);

- 충돌체크

- return true or false

**-- Client --**

~~void ChangeState(state \*changeState);~~

~~- 캐릭터의 상태를 변경~~

-> 서버에서 캐릭터 상태를 변경해 준다.

void Attack(Struct CollisionBox);

~~- 현재 지점에서 캐릭터에게 공격 명령을 했을 시에 CollisionBox 데이터를 서버로 보낸다.~~

-> Attack함수는 있으나 서버로 공격 명령 패킷 보내는 함수로 바뀜

~~void MoveChar(float \_x, float \_y, float \_z);~~

~~- 캐릭터를 해당 좌표만큼 이동시키고 현재 실위치 데이터를 서버에 전송한다.~~

-> 서버에서 계산한다.

void recvWorldData(); -> SetWorldData(struct WorldData);

- 지속적으로 월드 데이터를 받아 업데이트 한다.

**-- Server --**

~~bool SetForcedPosition(int CharID, float \_x, float \_y, float \_z);~~

~~- 플레이어를 찾아 성공적으로 이동시켰을 시 true 아이디가 없으면 false~~

~~-- ex) 플레이어가 월드맵 바깥으로 갔을때 강제 소환.~~

-> 서버내에서 위치 좌표만 바꿔주면 되므로 필요 없는 함수가 되었다.

~~bool SendDestroy(int CharID);~~

~~- 플레이어를 찾아 성공적으로 이동시켰을 시 true 아이디가 없으면 false~~

~~- 플레이어의 HP가 0이하 일 때 파괴시키는 명령어~~

-> 서버 내에서 모든 계산이 이루어 지고 클라이언트는 데이터만 받는다.

-> 필요 없는 함수가 되었다.

void SendWorldData(int CharID);

- 플레이어의 위치에 따라 시야 범위내의 데이터만 전송

-> 모든 플레이어를 렌더링

void CreateMonster(int MonsterLevel);

- 몬스터를 만드는 함수

void CreateItem(int ItemType);

- 아이템을 만드는 함수

개발 환경

1. 운영체제

- Windows7 32bit( 김대현 )

- Windows7 64bit( 박성경 )

2. 사용언어

- C++

- OpenGL

3. 개발도구

Visual Studio 2010

Visual Studio 2012

**역할 분담**

- 김대현

객체 프레임워크

클라이언트 데이터 전송 부분

클라이언트 월드 데이터 처리 부분

- 박성경

전송될 데이터 구조체 세부 디자인

서버 월드 데이터 전송 부분

몬스터 AI 구현

**바뀐 구현 내용**

|  |  |
| --- | --- |
| **구현 하려고 했던 내용** | |
| **Server** | **Client** |
| 멀티 스레드 서버 | 캐릭터 현재 위치 전송 |
| 공격 시 충돌체크 | Attack 기능 구현 |
| 모든 오브젝트 위치 전송 기능 | 다른 플레이어 렌더링 |
| Monster 위치 전송 | 몬스터 렌더링 |
| Create Monster | 충돌체크할 범위를 서버로 전송 |
| Create Item | 상태 변화 메시지 처리 |
| 아이템 효과 전송 | ItemType에 따른 메시지 처리 구현 |
| 플레이어 파괴 메시지 전송 |  |

**바뀐 내용**

|  |  |
| --- | --- |
| Client | |
| 변경 전 | 변경 후 |
| 캐릭터 현재 위치 전송 | 이동 방향 명령 패킷 전송 |
| Attack 기능 구현 | 공격 방향 명령 패킷 전송 |
| 다른 플레이어 렌더링 | 나를 포함 모든 플레이어 위치 동기화 |
| 몬스터 렌더링 | 몬스터 위치 동기화 |
| 충돌체크할 범위를 서버로 전송 | 공격 방향만 전송 |
| 상태 변화 메시지 처리 | 서버 내에서 처리 |
| ItemType에 따른 메시지 처리 구현 | 서버 내에서 처리 |

|  |  |
| --- | --- |
| Server | |
| 변경 전 | 변경 후 |
| 멀티 스레드 서버 | 변경 없음 |
| 공격 시 충돌체크 | 변경 없음 |
| 모든 오브젝트 위치 전송 기능 | 변경 없음 |
| Monster 위치 전송 | 변경 없음 |
| Create Monster | 변경 없음 |
| Create Item | 변경 없음 |
| 아이템 효과 전송 | 아이템 효과가 적용된 플레이어 데이터 전송 |
| 플레이어 파괴 메시지 전송 | 현재 상태를 지속적으로 전송 |
|  |  |

**바뀐 서버 설계의 변화**

Client가 보낸 패킷은 TCP/IP 프로토콜로 처음부터 최종까지 처리했다.

**UDP BroadCast를 통한 통신 서버**

**문제점 :**

1. 클라이언트의 주소를 알고 있어야 한다.
2. LAN이 아닌 외부 통신은 막힐 가능성이 크다.

**문제 해결법**

클라이언트의 주소를 알아야 해서 클라이언트에서 쓸모 없는 패킷을 보내 주소를 알아냈다.

* 쓸모 없는 패킷을 주고 받는 더 큰 문제가 생겼다.

**UDP MultiCast를 통한 통신 서버**

**문제점 :**

1. 내부 LAN을 통한 통신은 매우 좋았다.
2. 강의실에서 하니 같은 그룹을 쓰는 다른 팀이 있던 것 같다.
3. 이상한 패킷이 들어와 버린다.

**문제 해결법**

모든 통신을 TCP/IP로 하기로 결정 하였다.

**바뀐 클라이언트 설계의 변화**

통신 방법은 서버와 함께 변화 했으므로 동일하다.

**클라이언트에서 계산 후 서버로 전송 후 모든 플레이어에게 계산 결과를 전송**

**문제점**

1. 클라이언트간 동기화가 매우 불안정 하다.

**문제 해결법**

클라이언트는 내 캐릭터가 무엇을 할 지 명령 패킷만 보낸다.

이후 서버에서 모든 계산을 다해서 모든 플레이어에게 갱신 데이터를 보낸다.

**서버에서 계산 후 데이터를 받아 내캐릭터 포함 데이터를 갱신한다.**

**문제점**

1. 캐릭터들의 움직임이 부드럽지 못하다.

**문제 해결법**

각 캐릭터마다 가속도를 주고 데이터를 받기 전까지 가속도 유지를 하여

위치 보간을 해준다.