네트워크 게임 프로그래밍

추진 계획서

2013년 11월 5일

2010180020 박성경

2011180007 김대현

**애플리케이션 기획**

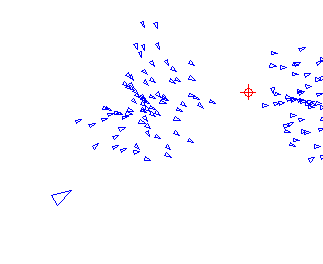
Mini MMORPG

: 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.

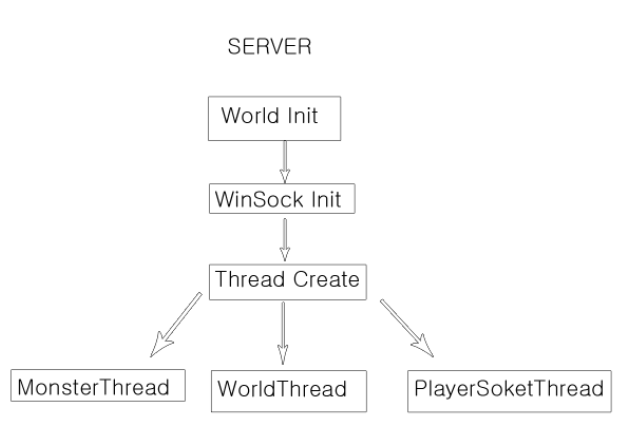
-- 서버 --

플레이어들의 상태를 관리한다

충돌체크가 된 플레이어들에게 공격 당했다는 메시지를 보낸다

클라이언트에게 업데이트된 몬스터와 플레이어와 아이템 데이터를 전송한다.

**High Level**



World Init : 게임 세계 초기화

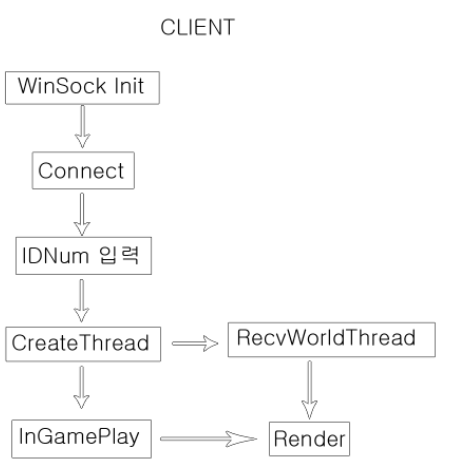
WinSock Init : 윈소켓 초기화

Thread Create : 서버에 필요한 쓰레드들 생성

MonsterThread : Monster의 AI와 같은 행동관련 쓰레드 생성

Worldthread : 전체 월드에 대한 관리 및 몬스터와 아이템 스폰 관리 쓰레드

PlayerSoketThread : ClientSoket을 처리해주는 쓰레드 (다중 쓰레드)



WinSock Init : 윈소켓 초기화

Connect : 서버와 연결

IDNum 입력 : 기존에 플레이 하던 유저인지 신규유저인지 서버에서 판별

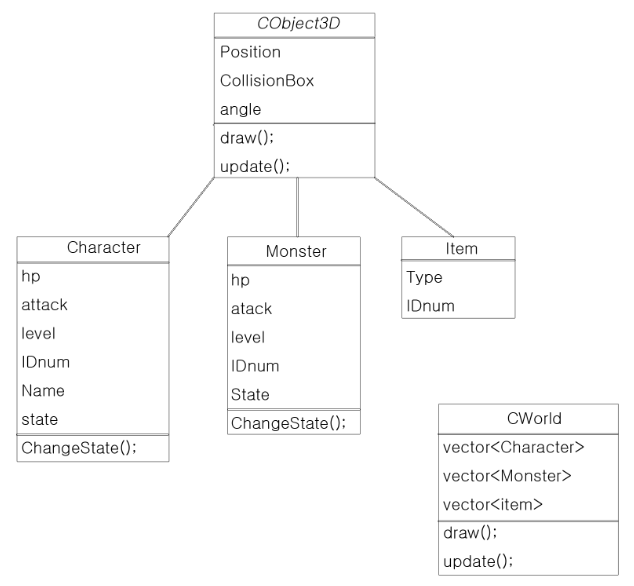
CreateThread : 서버패킷 처리 스레드 생성

InGamePlay : 플레이어 캐릭터의 제어 및 실제 게임 플레이

RecvWorldThread : 서버에서 준 데이터 월드를 토대로 렌더링 해줄 객체 관리

Render : 게임화면 렌더링 부분

**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

**-- STRUCT --**

struct POINT

float x, y, z

- 게임내에 쓰일 점좌표

struct CollisionBox

struct POINT Min

struct POINT Max

- 충돌체크에 쓰일 구조체 데이터

struct WorldData{}

- 월드 데이터 구조체

- 몬스터, 플레이어의 레벨, 위치좌표와 아이템 위치좌표가 들어간다.

**-- Functions --**

**-- Common --**

bool CollisionCheck(Struct CollisionBox, struct CollisionBox);

- 충돌체크

- return true or false

**-- Client --**

void ChangeState(state \*changeState);

- 캐릭터의 상태를 변경

void Attack(Struct CollisionBox);

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

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

- 캐릭터를 해당 좌표만큼 이동시키고 현재 실위치 데이터를 서버에 전송한다.

void recvWorldData();

- 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.

bool SendDestroy(int CharID);

- 플레이어를 찾아 성공적으로 이동시켰을 시 true 아이디가 없으면 false

- 플레이어의 HP가 0이하 일 때 파괴시키는 명령어

void SendWorldData(int CharID);

- 플레이어의 위치에 따라 시야 범위내의 데이터만 전송

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

**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

**개발일정** (각 일요일은 휴식타임)

1주 - 기본 뼈대

Server - 월드 구성과 위치 데이터 받기

Client - 기본적인 뼈대 객체, 객체 움직임, 현재 위치 전송

|  |  |  |
| --- | --- | --- |
| 일 | 이름 | 박 성경 | 김 대현 |
| 11/5(화) | 추진 계획서 최종 커펌 | 추진 계획서 최종 커펌 |
| 11/6(수) | 멀티 스레드 서버 구성하기 | 캐릭터, 몬스터 클래스 |
| 11/7(목) | 멀티 스레드 서버 구성하기 | 캐릭터, 몬스터 클래스 |
| 11/8(금) | 월드 구현 | 캐릭터 움직임 |
| 11/9(토) | 월드 구현 | 위치 데이터 전송 |

2주 - 위치 데이터 통신과 플레이어 다중 접속 기능

Server - 위치 데이터 갱신하여 플레이어 위치 전송

Client - 월드데이터를 받아 타 플레이어를 렌더

|  |  |  |
| --- | --- | --- |
| 일 | 이름 | 박 성경 | 김 대현 |
| 11/11 | 위치 데이터 구조체 처리 | 위치 데이터 전송 |
| 11/12 | 다중 접속 시도 | 몬스터 구현 |
| 11/13 | 다중 접속 시도 | 몬스터 구현 |
| 11/14 | 한 플레이어게 다른 플레이어 위치 전송 | 위치 데이터 받아 처리하기 |
| 11/15 | 위치 전송 | 위치 데이터 받아 처리하기 |
| 11/16 | 기타 최적화 작업 | 위치 데이터 받아  타 플레이어 렌더링 |

3주 - 몬스터 및 아이템 구현

Server - 플레이어 위치를 토대로 몬스터 AI 구현 및 아이템과 몬스터 위치 전송

Client - 월드 데이터에 포함된 몬스터 정보를 받아 렌더링

|  |  |  |
| --- | --- | --- |
| 일 | 이름 | 박 성경 | 김 대현 |
| 11/18 | 몬스터 클래스 받아 월드 구성 | 몬스터와 캐릭터 동시 렌더링 |
| 11/19 | 월드 데이터 구조체 구현 | 월드 데이터 구조체 구현 |
| 11/20 | 몬스터의 위치 좌표 전송 | 몬스터의 위치 좌표 처리 |
| 11/21 | 몬스터의 위치 좌표 전송 | 몬스터의 위치 좌표 처리 |
| 11/22 | 플레이어 위치데이터로  몬스터 AI 구현 | 몬스터와 캐릭터 동시 렌더링 |
| 11/23 | 몬스터 AI 구현 | 몬스터와 캐릭터 동시 렌더링 |

4주 캐릭터 공격 명령어, 아이템 구현 및 동기화

Server - 모든 플레이어가 같은 화면을 볼 수 있도록 동기화, 캐릭터 공격 명령어 받기, 아이템 구현

Client - 디버그 및 데이터 전송 최종 점검, 아이템 메세지 처리

|  |  |  |
| --- | --- | --- |
| 일 | 이름 | 박 성경 | 김 대현 |
| 11/25 | CollisionBox 데이터 처리 | CollisionBox 데이터 전송 |
| 11/26 | 받은 데이터로 충돌체크 | 공격명령함수 만들기 |
| 11/27 | 받은 데이터로 충돌체크 | 공격명령함수 만들기 |
| 11/28 | SendDestroy() 구현 | 받은 데이터로 ChangeState()함수 처리 |
| 11/29 | SendDestroy() 구현 | ChangeState()함수 처리 |
| 11/30 | 아이템 구현 | 아이템 메세지 처리 준비 |

5주 최종 플레이 및 시연 준비

Server - 시연될 장소의 네트워크 환경에 따른 준비

Client - 디버그 및 데이터 전송 최종점검

|  |  |  |
| --- | --- | --- |
| 일 | 이름 | 박 성경 | 김 대현 |
| 12/2 | 아이템 메세지 전송 | 아이템 메세지 처리 |
| 12/3 | 강의실 네트워크 조사 | 강의실 클라이언트 환경 조사 |
| 12/4 | 네트워크 환경에 따른 환경 맞춤 작업 | 디버그 |
| 12/5 | 강의실에서 다시 테스트 | 강의실에서 다시 테스트 |
| 12/6 | 또 다른 네트워크에서 테스트 | 또 다른 네트워크에서 테스트 |
| 12/7 | 최종 테스트 | 최종 테스트 |

6주 최종 평가

Server - 최종 테스트

Client - 최종 테스트

|  |  |  |
| --- | --- | --- |
| 일 | 이름 | 박 성경 | 김 대현 |
| 12/9 | 최종 테스트 | 최종 테스트 |
| 12/10 | 최종 강의실 테스트 | 최종 강의실 테스트 |
| 12/11 | 최종 테스트 | 최종 테스트 |