REPORT

2025 네트워크 게임 프로그래밍 텀프로젝트 추진계획서

한국공학대학교
TECH UNIVERSITY OF KOREA

학과	게임공학과
과목명	네트워크 게임 프로그래밍(03)
학생	2023182041 이용재 2020184029 이현석 2020180045 황태규
제출일	2025.10.25

목차

Ⅰ. 애플리케이션 기획

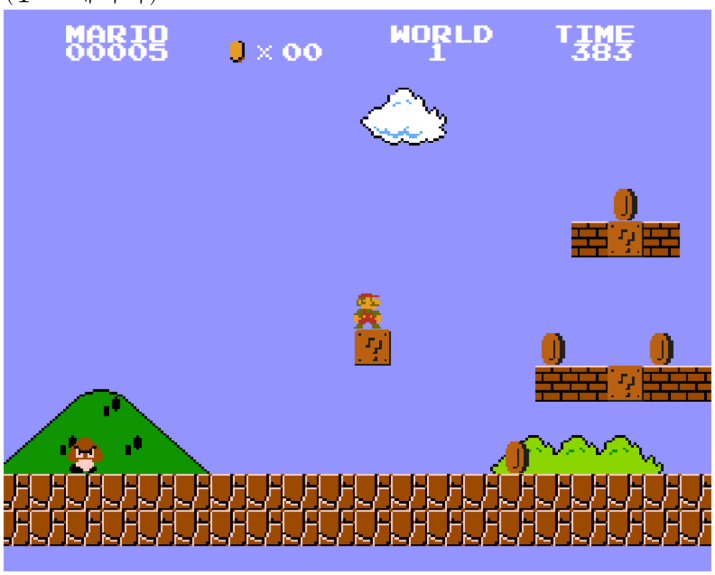
1. 게임 설명

2025－1 윈도우 프로그래밍 서 WINAPI 를 통하여 개발 | 이용재

본 게임은 'SUPER MARIO BROS'의 모작으로, 2D 사이드 스크롤 플랫폼 액션 게임이며 점프, 이동, 몬스터 공격 등을 통하여 각 스테이지를 클리어하며, 최종 구간까지 도달하는 것이 목표입니다.

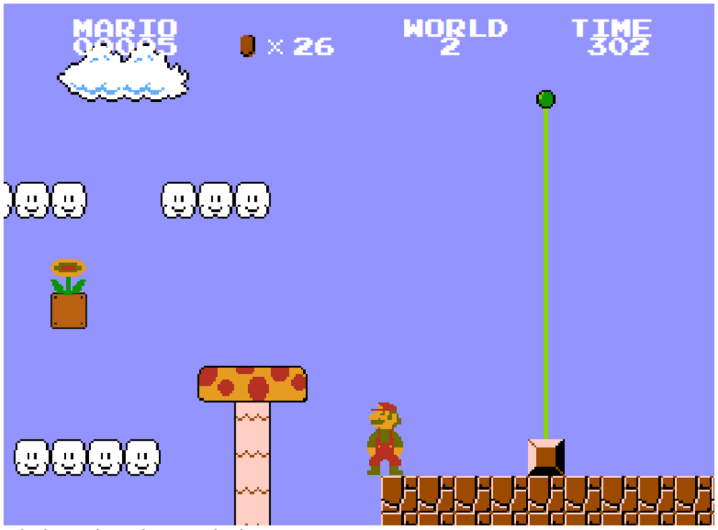
2. 월드

(1 스테이지)



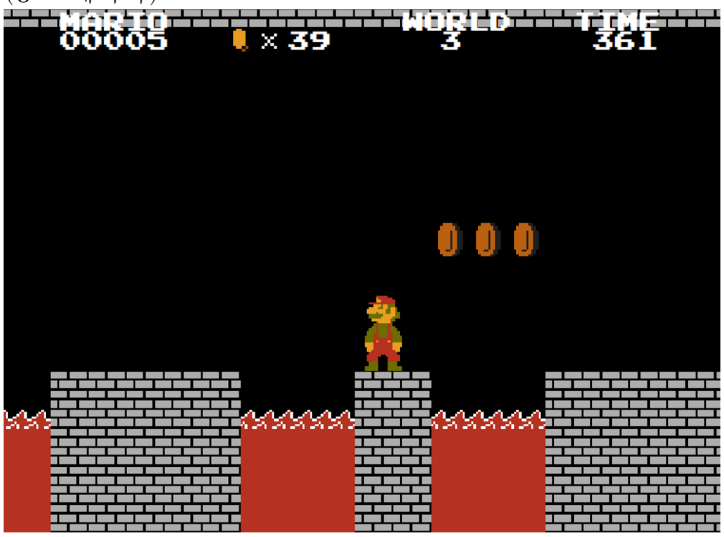
테마: 낮, 숲

(2 스테이지)



테마: 낮, 하늘, 버섯

(3 스테이지)



테마: 쿠파 성

3. 플레이어

1P : 마리오
2P : 루이지
3P : 와리오
플레이어의 상태:
- Normal mario: 게임의 디폴트 마리오 상태
- Star Mario: 10 초간 무적이 되며, 플레이어와 몬스터 충돌 시 몬스터가 한 방에 사망
- Special Mario: Normal mario 보다 상단으로 한칸 더 크며, 데미지를 받을 시 Normal mario 상태로 변함
    - Big Mario:
    - Flower Mario: SPACE 입력 시 불꽃이 플레이어가 바라보는 방향으로 떨어짐
        - 불꽃: 중력의 영향을 받으며, 바닥에 충돌 시 위로 뛰어오름, 몬스터 및 블록에 충돌 시 사라짐
    - Tino Mario: SPACE 입력 시 불꽃이 플레이어가 바라보는 방향으로 일직선으로 날아감, Z 입력 시 플레이어가 바라보는 방향으로 물기
        - 티노불꽃: 중력의 영향을 받지 않으며, 몬스터 및 블록을 관통함
            쿨타임: 5 초
        - 물기: 잠시동안 무적이 되며 일정 부분에 몬스터가 있다면 데미지를 입힘
            쿨타임: 15 초

4. 아이템

버섯:

－등장 시 우측으로 이동하며, 블록을 만날 시 방향을 전환함

－효과: 플레이어가 상태가 Small mario상태라면 Big mario로 전환

초록버섯:

－효과: 플레이어의 목숨 + 1

꽃:

－해당 아이템은 움직이지 않음

－효과: Flower mario 상태로 전환

티노:

－해당 아이템은 움직이지 않음

－효과: Tino mario 상태로 전환

스타:

－등장 시 위로 점프를 하며, 블록과 만날 시 방향을 전환하고, 바닥에 충돌할 시 튀어오름

－효과: Star mario 상태로 전환

5. 몬스터

갈색 굼바:

–플레이어에게 밟힐 시 찌그러져 죽음

–좌우로 이동하며, 벽을 만날 시 방향을 전환

초록 굼바, 회색 굼바:

–플레이어에게 밟힐 시 찌그러져 죽음

–좌우로 이동하며, 벽을 만날 시 방향을 전환, 낭떠러지에서 떨어지지 않음

거북이 1, 2:

–플레이어에게 밟힐 시 등껍질에 들어감

–좌우로 이동하며, 벽을 만날 시 방향을 전환, 낭떠러지에서 떨어지지 않음

–플레이어가 등껍질을 밟을 시 등껍질의 중앙을 기준으로 좌우로 나뉘며, 밟힌 방향으로 날아감, 등껍질에 맞을 시 어떤 오브젝트던 체력 –1

천사 거북이:

–날개가 달려있어 공중에서 위아래로 천천히 떠있음

–플레이어가 한 번 밟을 시 날개가 떨어져 일반 거북이 1로 변함

–나머지 위 거북이1,2 와 동일

쿠파:

–랜덤으로 좌우로 이동 및 점프

–랜덤으로 불꽃을 발사 / 불꽃은 일직선으로 중력 및 블럭의 영향없이 좌측으로 날아감

–플레이어가 불꽃에 충돌할 시 플레이어 데미지 1

–체력: 100

–쿠파 사망 시 우측 엔딩으로 가는 블록 제거

6. 조작

– 방향키: 이동

– SPACE: 아이템 스킬

– Z: 아이템 스킬 2

7. 상호작용

–플레이어가 점프를 하여 플레이어 상단이 아이템블록에 충돌할 시 해당 블록의 아이템이 해당블록의 상단으로 생성

–플레이어가 몬스터의 상단쪽을 밟을 시 몬스터의 체력 –1 (쿠파는 해당되지 않음)

–플레이어가 버섯, 꽃, 티노, 스타 등의 아이템에 충돌할 시 해당 아이템의 효과를 얻음

8. 멀티플레이 추가 기능

–3인 플레이 게임

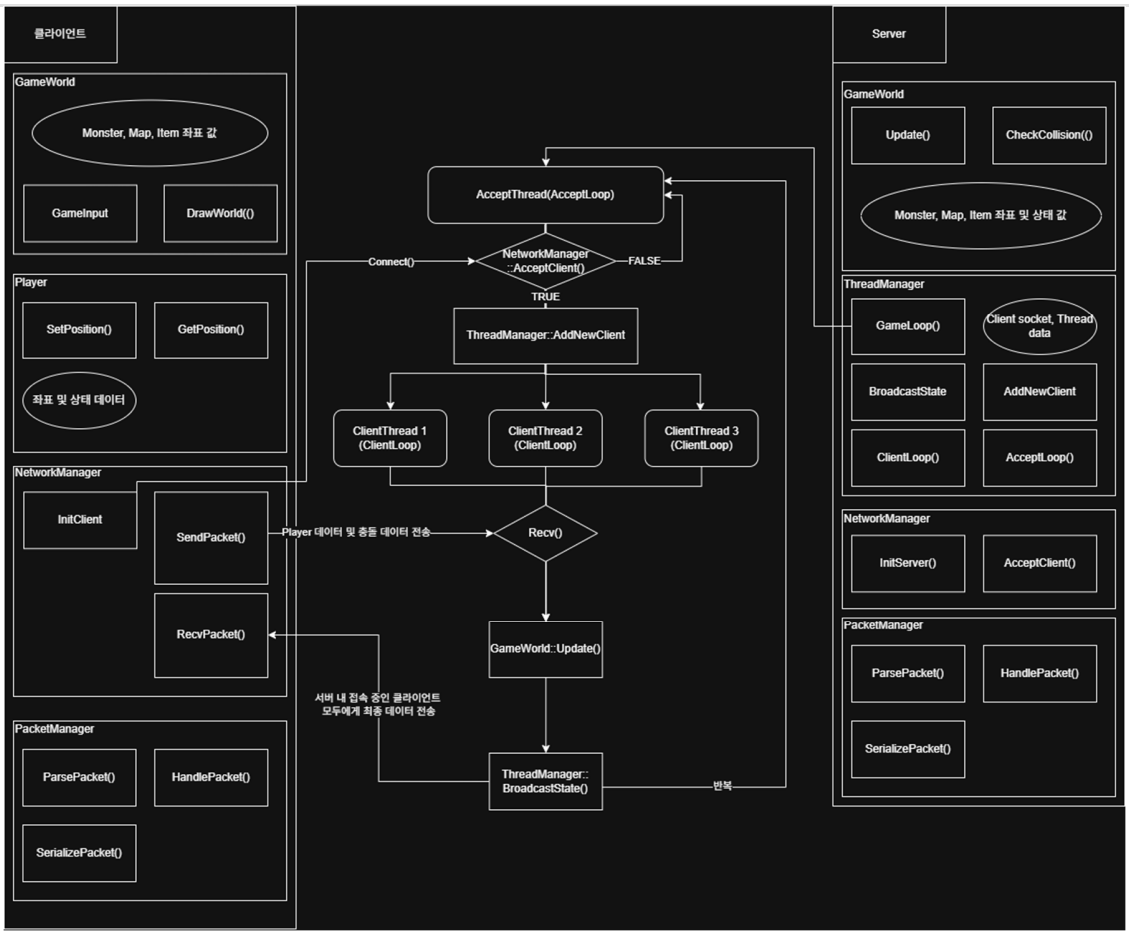
–각각의 플레이어가 각각의 카메라 좌표를 가지고있어, 각각의 화면이 다름

–게임 클리어 조건 변경: 게임에 참여하고있는 플레이어 모두가 깃발에 충돌 시 게임 클리어

–목숨 공용으로 변경

–플레이어간 상호작용: 각각의 머리를 밟아 더 높게 뛰어 오를 수 있음

Ⅱ. High－Level 디자인



Threads:

1. Main Thread:

– 시작/종료 관리

– 게임 내의 로직계산함수(Update()) 처리

– BroadcastState() 함수 처리

– 패킷관리 함수(handlePacket(), ParsePacket()) 처리

2. Accept Thread:

– 새로운 클라이언트의 접속 AcceptLoop() 함수만을 처리

3. Client Thread:

– 각 클라이언트가 1:1 로 서버와의 송수신 처리

Ⅲ. Low－Level 디자인

클라이언트		
GameWorld	void GameInput()	클라이언트의 입력 데이터를 처리
	void DrawWorld()	플레이어, 몬스터, 맵 타일, 아이템, UI 등 화면에 그리는 전체 렌더링 호출
Player	void SetPosition()	GameInput()으로부터 전달 받은 값(vx, vy 및 x,y 값)으로 플레이어 객체의 위치를 설정
	int GetPosition()	플레이어 객체의 현재 x 또는 y 좌표 값을 반환
NetworkManager	bool SendPacket()	프로토콜 송신. GameInput(),CheckCollision() 에서 값을 전달받아 SerializePacket()을 통해서 전송용 패킷으로 만들어 소켓을 통해 서버로 전송
	int RecvPacket()	프로토콜 수신. 서버로부터 데이터를 수신할 버퍼를 준비하고, 소켓을 통해 수신
	bool InitClient()	WinSock 초기화하고, 서버의 IP 주소와 포트 번호를 사용하여 서버에 연결(connect() API 호출)을 시도하고 연결 성공 여부를 반환
PacketManager	bool ParsePacket()	네트워크로 수신한 바이트 스트림을 구조체/객체 형태로 변환
	bool HandlePacket()	파싱된 패킷 내용을 실제 게임 로직에 반영
	void SerializePacket()	구조체/객체 형태의 데이터를 바이트 스트림으로 변환하여 네트워크 전송용 패킷으로 만듦

서버

NetworkManager	bool InitServer()	WinSock 초기화하고 리스닝 소켓 생성, 바인딩 및 대기(listen)
	void AcceptClient()	accept() 함수를 호출하여 연결 수락, Accept Thread 가 이 함수를 호출하여 접속 처리를 전담
PacketManager	bool ParsePacket()	네트워크로 수신한 바이트 스트림을 구조체/객체 형태로 변환
	bool HandlePacket()	파싱된 패킷 내용을 실제 게임 로직에 반영
	void SerializePacket()	구조체/객체 형태의 데이터를 바이트 스트림으로 변환하여 네트워크 전송용 패킷으로 만듦
GameWorld	void Update()	게임 맵 및 오브젝트 업데이트 함수
	void CheckCollision()	플레이어와 블록, 플레이어와 몬스터 간의 상호작용 및 충돌을 검사
ThreadManager	void GameLoop()	서버의 시작버튼과 같은 함수로 acceptThread AcceptLoop 할당 및 mainThread 에서의 핵심인 로직함수, 패킷큐처리 함수, 브로드캐스트 함수를 Loop 로 처리
	void AcceptLoop()	accept 했을 때 INVALID-SOCKET 이 아니라면 AddNewClient(새 클라이언트에 스레드 및 playerID 할당) 호출
	void BroadcastState()	상태 동기화 함수로 서버 내 접속해 있는 모든 플레이어에게 동시 전송
	int AddNewClient()	0~MAX-PLAYERS 의 빈 값을 찾아 Client 에게 PlayerID 값을 할당 및 해당 ID 의 플레이어를 활성화, 소켓 정보를 등록해준 후 ClientLoop 를 할당받은 스레드를 생성해줌
	void ClientLoop()	recv 를 통해 받은 값을 패킷큐에 저장

Application Protocol

서버		클라이언트	
InitServer()	소켓 초기화 및 논블록 소켓 할당	InitClient()	소켓 초기화, 할당 및 연결
BroadcastState()	상태 동기화 함수로 서버 내 접속해 있는 모든 플레이어에게 동시 전송	SendPacket()	클라이언트의 입력 데이터를 서버로 전송
AddNewClient()	PlayerID 값 소켓정보 할당 및 해당 ID 의 플레이어를 활성화 클라이언트스레드 생성	RecvPacket()	서버가 브로드캐스트한 데이터를 수신 및 패킷큐에 저장

스레드 함수 정의

1. ThreadManager::GameLoop() (Main Thread)

역할: 서버의 메인 게임 루프를 실행합니다.

기능: ProcessInputQueue()를 통해 패킷을 처리하고, CalculateGamePhysics()로 로직을 계산하며, BroadcastState()로 상태를 전파하는 서버의 심장역할을 합니다.

스레드 유형: 일반적으로 메인 스레드로 불리며, 서버의 모든 로직 처리와 상태 일관성을 책임집니다.

2. ThreadManager::AcceptLoop() (Accept Thread)

역할: 새로운 클라이언트의 접속 요청을 처리합니다.

기능: 논블로킹 방식으로 NetworkManager::AcceptClient()를 지속적으로 호출하여 새 연결을 수락하고, 성공하면 해당 클라이언트를 위한 ClientLoop 스레드를 생성합니다.

스레드 유형: I/O 스레드의 일종으로, 접속 처리만을 전담합니다.

3. ThreadManager::ClientLoop(int playerID, SOCKET clientSock) (Client Thread)

역할: 할당된 단일 클라이언트와의 네트워크 I/O 를 전담합니다.

기능: recv()함수를 통해 클라이언트의 패킷을 수신하고, ParsePacket 역할(코드 내 인라인 구현)을 수행하여 PacketInfo 를 생성한 후, 패킷 큐(m_packetQueue)에 삽입하는 역할까지 수행합니다.

스레드 동기화 방법

1. 동기화 자원

- packetQueue: 가장 중요한 공유 자원으로 ClientLoop()쓰기와 GameLoop() 읽기가 동시에 접근

- currentmap, monsters, items: 모든 플레이어가 공유하는 자원으로 위와 마찬가지로 동시에 접근

2. 동기화 매커니즘

- std::mutex m-mtx: 모든 공유 자원 접근 시 사용되는 lock 객체입니다.

PacketManage:

– PacketInfo 구조체

	멤버변수	설명
PacketInfo	int playerID	해당 패킷을 송수신하게 될 ID
	char type	패킷의 타입 ex) MOVE = 1, ATTACK = 2, HIT = 3, BLOCK_ATTACK = 4
	MOVE	
player	int x, y	플레이어의 좌표
	int vx, vy	플레이어의 이동 속도
	char state	플레이어의 상태
	ATTACK	
COLLISION	int targetID	해당 몬스터의 ID
	int damage	입힐 데미지
	HIT	
	int damage	받을 데미지
	BLOCK_ATTACK	
	int blockID	해당 블록의 ID
	int block_x, int block_y	해당 블록의 좌표

Ⅳ. 팀원 별 역할분담

팀원	주요 역할	세부 담당 내용
이용재	클라이언트 리펙토링 및 클라이언트 NETWORKMANAGER 관리 작성, GAMEWORLD/checkCollision 작업	– 클라이언트에서 NetworkManager 초기화 (InitClient()) – 패킷 송수신 (SendPacket(), RecvPacket()) – 서버와의 연결 테스트 및 오류 처리 – 클라이언트 쪽 네트워크 통신에 집중 – 로컬 입력과 패킷 전송 연동 가능
이현석	서버-클라이언트 PacketManager 표준 관리 및 작업, GAMEWORLD/map, player 작업	– 패킷 파싱 (ParsePacket()) – 플레이어 이동/상태 처리 (handlepacket()) – 서버와 클라이언트 모두의 패킷 처리 로직을 통합 – 패킷 포맷, 타입 정의와 검증 책임
황태규	GAMEWORLD/Object 관리 및 서버 Manager, NetworkManager 작업	– 서버에서 NetworkManager 초기화 (InitServer()), 클라이언트 접속 관리 – 서버 상태 관리 (GameWorld, Player 등) – 동기화 상태 전송, 충돌/게임 로직 처리 후 패킷 전송 – 데이터 관리와 네트워크 전송을 동시에 담당 – 스레드 생성 및 플레이어 ID 관리, 스레드 역할 할당

V. 개발 환경

개발 도구: Visual Studio 2022

API: Windows API (WinAPI)

통신 프로토콜: TCP/IP

버전 관리: Git, GitHub

VI. 개발 일정

1. 이용재 (클라이언트 네트워크)

일	월	화	수	목	금	토
	10/27	10/28	10/29	10/30	10/31	11/1
	추진계획서 제출	high-levle 디자인 및 packetInfo 구조체 계획 작성	추진계획서 수정	추진계획서 작성		프로젝트 C -> C++ 스타일 리펙토링 작업
11/2	11/3	11/4	11/5	11/6	11/7	11/8
프로젝트 리펙토링 완성		프로젝트 구조 파악 클래스 정의 확인 Networtk Manager 클래스 틀 작성	Init() 작성 및 테스트용 로컬 서버 코드 작성	로컬서버 accept() connect() 테스트	테스트용 Send(), Recv() 함수 구현	싱글 -> 멀티 로직 분리
11/9	11/10	11/11	11/12	11/13	11/14	11/15
서버 checkCollision() 구현		프로젝트 싱글플레이 + 멀티플레이 베이스 단계	프로젝트 싱글플레이 + 멀티플레이 베이스 단계		SendPacket() 구현	SendPacket() 구현
11/16	11/17	11/18	11/19	11/20	11/21	11/22
		SendPacket() 구현	RecvPacket() 구현		RecvPacket() 구현	간단한 패킷 전송 로그 확인 및 테스트
11/23	11/24	11/25	11/26	11/27	11/28	11/29
RecvPacket() 구현 패킷 수신 확인		PacketHandler 연동	PacketHandler 연동 테스트		Recv + Send 통합 테스트	Recv + Send 통합 테스트

11/30	12/1	12/2	12/3	12/4	12/5	12/6
패킷 손실/지연 테스트		패킷 손실/지연 테스트	여러 클라이언트 동시 접속, 위치 동기화 확인	여러 클라이언트 동시 접속, 위치 동기화 확인		최종 서버/클라이언트 연동 테스트
12/7	12/8	12/9	12/10	12/11		
최종 서버/클라이언트 연동 테스트				최종 제출		

2. 이현석 (패킷 핸들러)

일	월	화	수	목	금	토
	10/27	10/28	10/29	10/30	10/31	11/1
	추진계획서 제출		추진계획서 수정	추진계획서 작성		
11/2	11/3	11/4	11/5	11/6	11/7	11/8
	GameWorld 서버화	GameWorld 서버화	GameWorld::Map 수정		GameWorld::Map 수정	
11/9	11/10	11/11	11/12	11/13	11/14	11/15
GameWorld::Player 수정	GameWorld::Player 수정	프로젝트 싱글플레이 + 멀티플레이 베이스 단계	프로젝트 싱글플레이 + 멀티플레이 베이스 단계			클라, 서버 공용 ParsePacket() SerializePacket()
11/16	11/17	11/18	11/19	11/20	11/21	11/22
클라, 서버 공용 ParsePacket() SerializePacket()	클라 HandlePacket()	클라 HandlePacket()	서버 HandlePacket()		서버 HandlePacket()	
11/23	11/24	11/25	11/26	11/27	11/28	11/29
		PacketHandler 연동	PacketHandler 연동 테스트		Recv + Send 통합 테스트	Recv + Send 통합 테스트
11/30	12/1	12/2	12/3	12/4	12/5	12/6

패킷 손실/ 지연 테스트		패킷 손실 / 지연 테스트	여러 클라이언트 동시 접속, 위치 동기화 확인	여러 클라이언트 동시 접속, 위치 동기화 확인		최종 서버/ 클라이언트 연동 테스트
12/7	12/8	12/9	12/10	12/11		
최종 서버/ 클라이언트 연동 테스트				최종 제출		

3. 황태규 (서버 로직)

일	월	화	수	목	금	토
	10/27	10/28	10/29	10/30	10/31	11/1
	추진계획서 제출		추진계획서 수정	추진계획서 작성		추진계획서 수정
11/2	11/3	11/4	11/5	11/6	11/7	11/8
	InitServer() 작성	AcceptClient() 작성	GameLoop() 작성	GameLoop() 작성	AcceptLoop() 작성	AddNewClient() 작성
11/9	11/10	11/11	11/12	11/13	11/14	11/15
ClientLoop() 작성	GameLoop() 검토 및 수정	프로젝트 싱글플레이 + 멀티플레이 베이스 단계	프로젝트 싱글플레이 + 멀티플레이 베이스 단계		Broadcast State() 작성	GameWorld::Monster 수정
11/16	11/17	11/18	11/19	11/20	11/21	11/22
GameWorld::Monster 수정					GameWorld::Item 수정	GameWorld::Item 수정
11/23	11/24	11/25	11/26	11/27	11/28	11/29
		PacketHandler 연동	PacketHandler 연동 테스트		Recv + Send 통합 테스트	Recv + Send 통합 테스트
11/30	12/1	12/2	12/3	12/4	12/5	12/6
패킷 손실/지연 테스트		패킷 손실/지연 테스트	여러 클라이언트 동시 접속, 위치 동기화 확인	여러 클라이언트 동시 접속, 위치 동기화 확인		최종 서버/클라이언트 연동 테스트

12/7	12/8	12/9	12/10	12/11		
최종 서버/클라이언트 연동 테스트				최종 제출		