네트워크 게임 프로그래밍 **Term Project** 추진계획서 **V2**

탱크 **TFS** 게임

2025.10.31

게임공학과 2021184042 홍성호
게임공학과 2021182005 김영규
게임공학과 2019180042 한건우

애플리케이션 기획

(3d게임프로그래밍1- 한건우**)**

1. 게임 설명

3인용 1vs1vs1 TFS 게임이며 서로 랜덤한 위치에서 시작해서 적을 찾아 죽이고 혼자 살아남으면 승리하는 게임입니다.

2. 게임 컨셉

장르	멀티 TFS게임
플레이 방식	마우스 + 키보드 조작
플랫폼	PC
재미 요소	한정된 시야에서 실시간으로 적을 찾아다니며 공격 및 방어

3. 월드**(**환경**)**

한정된 공간 및 시야를 방해하는 장애물, 벽 존재

4. 플레이어 정보

각 플레이어(탱크)는 다음과 같은 상태 정보를 가집니다:

- 체력 (HP)
- 탱크 위치 및 회전
- 발사체의 위치
- 현재 피격 가능 여부

5. 조작 방식

동작	입력 키	설명
전진 / 후진	W / S	탱크를 앞으로 또는 뒤로 이동
좌 / 우 회전	마우스 이동	좌우/방향 조정
포격	A	발사체를 바라보는 방향으로 발사
방어	D	장애물, 발사체의 충돌을 무시

6. 전투 시스템

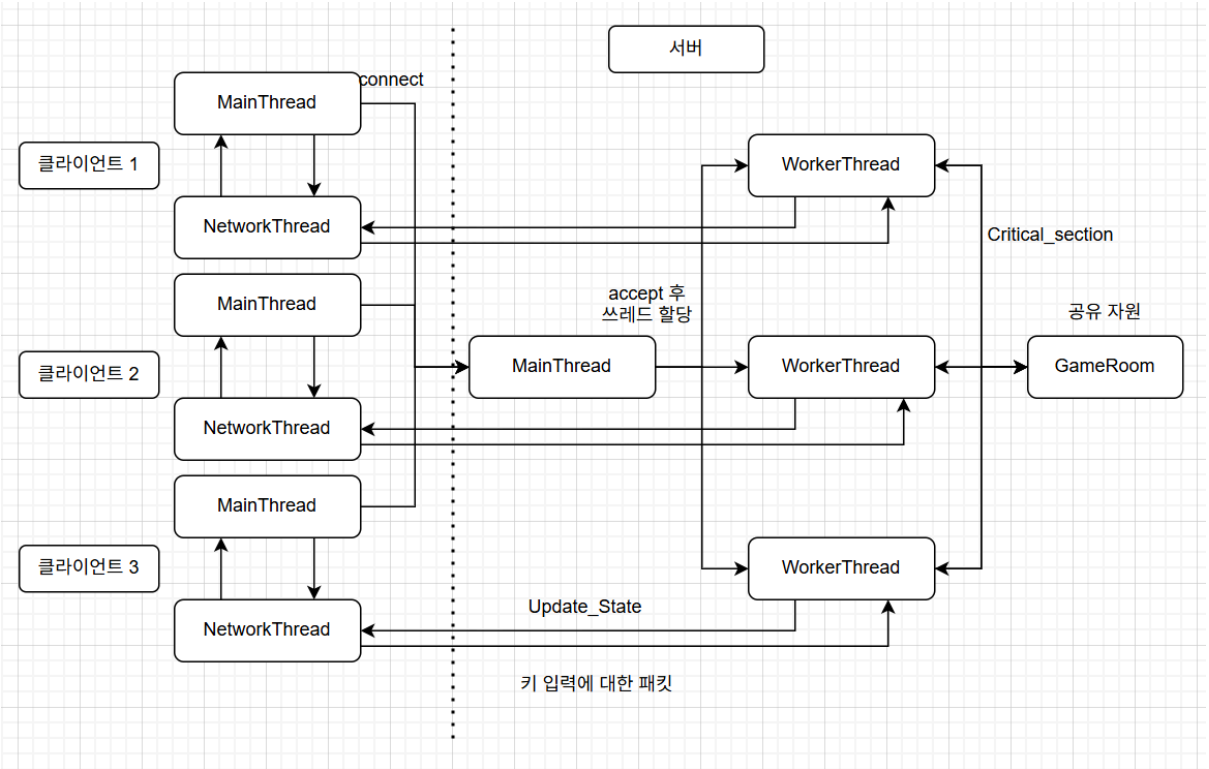
마우스, 키보드를 사용하여 탱크를 움직여 맵을 탐사하고, 적 발견시 포격합니다. 이때 적은 날아오는 발사체를 방어할 수 있습니다.

7 엔딩

체력이 0이 되면 탱크가 파괴되며, 단 하나의 탱크가 남으면 게임이 종료됩니다.

High-Level 디자인

흐름도



- 서버

- 메인 쓰레드
  역할 : 서버의 메인 쓰레드로 클라이언트의 접속 요청을 받아들이는 역할을 합니다.
  흐름 : listen 상태로 대기하다가 accept()를 통해 클라이언트의 접속 요청이 들어오면 해당 클라이언트와 통신할 워커 쓰레드를 생성하고 클라이언트의 소켓 정보를 넘겨준 뒤 다시 accept() 대기 상태로 돌아갑니다.

- 워커 쓰레드
  역할 : 접속한 클라이언트당 하나씩 생성되어 해당 클라이언트와의 모든 통신을 전담합니다.
  흐름: 할당 된 클라이언트로부터 recv하고 수신된 패킷(데이터)를 바탕으로 GameRoom의 상태를 변경하는 로직을 처리합니다. 또한 주기적으로 최신 상태를 클라이언트에게 전송하여 동기화 합니다.

- 게임 룸(GameRoom)
  역할 : 모든 워커 쓰레드가 공유하는 자원입니다. 현재 게임에 참여 중인 모든 플레이어의 상태 정보(Player 객체)를 관리합니다. 게임의 핵심적인 로직을 처리하는 함수를 제공하며, 공유 자원에 대한 접근은 임계영역을 설정하여 보호합니다.

- 클라이언트

- 메인 쓰레드
  역할 : 사용자의 키 입력을 처리하고 화면을 렌더링하며 게임의 전반적인 로직을 실행합니다.

  흐름 : 매 프레임마다 키보드/마우스 입력을 감지하여 Sendqueue에 넣고, Recvqueue에 네트워크 쓰레드로부터 온 데이터가 있는지 확인하여 게임 상태를 갱신합니다. 갱신하고 난 후, 갱신된 상태를 바탕으로 화면에 캐릭터와 월드를 그립니다.

- 네트워크 스레드
  역할 : 서버와의 모든 통신을 전담합니다.

  흐름 : 서버로부터 데이터를 수신하기 위해서 대기하다가 데이터가 오면 Recvqueue에 넣습니다. 또한 주기적으로 Sendqueue를 확인하여 메인 스레드가 요청한 데이터가 있다면 서버로 send합니다.

- Sendqueue, Recvqueue
  역할 : 메인 쓰레드와 네트워크 쓰레드 간의 데이터 교환을 위한 통로입니다. 두개의 큐로 구성되어있고 임계영역설정으로 보호합니다.

- CGameFramework::HandlePacket()

역할: Recvqueue에 수신된 네트워크 패킷을 패킷 ID에 따라 해석하고, 해당 데이터를 바탕으로 플레이어의 위치, 회전, 발사체 입력 등을 업데이트 한다.

Low-Level 디자인

- 서버

Main()	
	1.wsastartup, socket,bind, listen 2.GameRoom 생성 3. while 루프 진입 4. accept()로 접속 대기 5. 접속 성공 시, player객체 생성 GameRoom에 등록 6.CreateThread()로 해당 Player를 전담할 WorkerThread 생성 및 실행. 7.다시 accept() 대기

WorkerThreadMain()	
	1.while루프 진입 2.GameRoom->Update_State() 호출하여 내 클라이언트에게 모든 Player들의 상태 데이터를 전송 3. recv() (50ms 타임아웃) 호출 4. 수신 데이터가 있다면 GameRoom->HandlePacket 호출하여 게임 로직 처리

class GameRoom	
private	PlayerManager CRITICAL_SECTION _lock GameState _gamestate
public	Check_GameState() HandlePacket() AddPlayer() RemovePlayer() Update_State() Broadcast() Move() Fire() Rotate() Raycast()

class Player	
public	Player_ID SOCKET Get position() Get Look() Get Up() Get Right() Get HP() CheckFire_Flag() ResetFire_Flag()
private	XMFLOAT3 position XMFLOAT3 Velocity XMFLOAT3 Look XMFLOAT3 Up XMFLOAT3 Right int HP bool Fire_Flag

- 클라이언트

class Sendqueue	
public	Push() Pop()
private	queue<key> _queue Critical_Section _lock

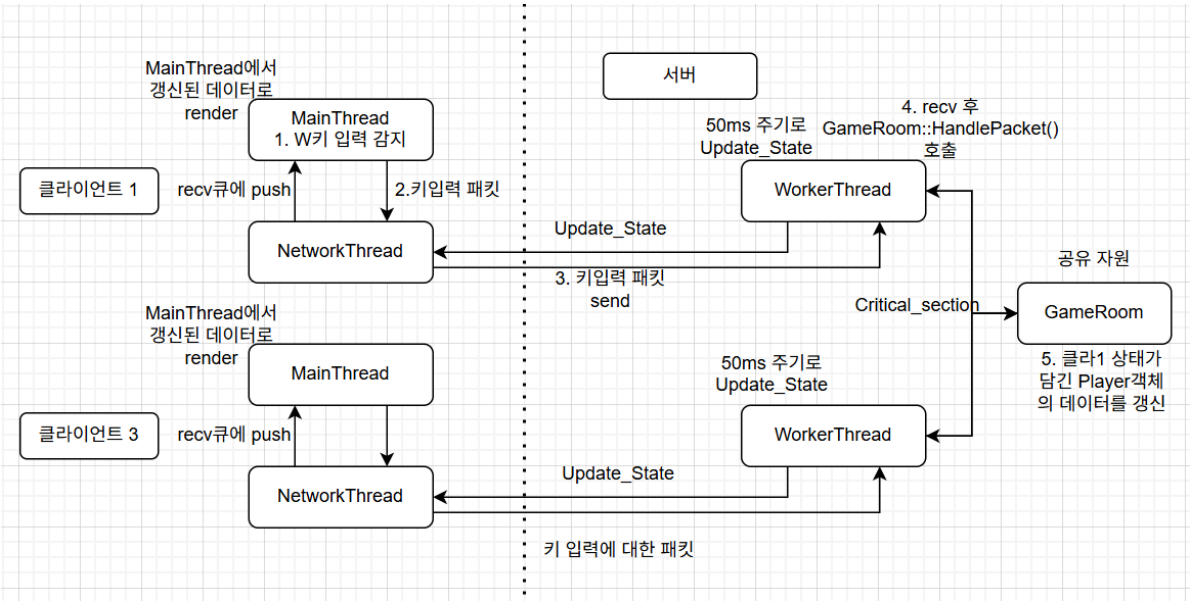
struct key

char id; // 요청 정보
char data[] // 실제 데이터

class Recvqueue	
public	Push() Pop()
private	queue<State> _queue Critical_Section _lock

struct State
char id; char data //…

CGameFramework::HandlePacket()



어플리케이션 프로토콜

enum class GameState
WAITING RUNNING END

struct PacketHeader
uint16_t size char id // Packet_ID

struct KeyInputData
char KeyInput

struct PlayerState
char Player_ID char hp float pos_x float pos_y float pos_z float yaw char Shield bool fire

struct CursorPosData
POINT CursorPos

enum Packet_ID
KEY_INPUT = 1 MOVE_MOUSE = 2 FIRE_OBJECT = 3 PLAYER_STATE = 4

팀원 별 역할 분담

홍성호	김영규	한건우
서버 측 Main()	클라이언트 NetworkThread	class Player
WorkerThreadMain()	SendQueue	GameRoom::Move()
GameRoom::AddPlayer	RecvQueue	GameRoom::Rotate()
GameRoom::RemovePlayer	CGameFramework::	GameRoom::Raycast()
GameRoom::HandlPacket	HandlePacket()	GameRoom::Fire()
GameRoom::Check_GameState()	클라이언트 로컬 연산 제거	Player 지형 발사체 충돌처리
GameRoom::Broadcast()	및 서버 권한 구조 전환	GameRoom::Update_State

개발환경

GitHub

Visual Studio

일정

홍성호

1주

11/02(일)	11/03(월)	11/04(화)	11/05(수)	11/06(목)	11/07(금)	11/08(토)
서버 측 Main() 구현					project progress report 작성	WorkerThreadMain() 구현

2주

11/09(일)	11/10(월)	11/11(화)	11/12(수)	11/13(목)	11/14(금)	11/15(토)
GameRoom ::AddPlayer () 구현				팀원들과 동시 접속 테스트.	코드 리뷰, project progress report 작성	

3주

11/16(일)	11/17(월)	11/18(화)	11/19(수)	11/20(목)	11/21(금)	11/22(토)
GameRoom ::BroadCast ()				GameRoom ::HandlePacket() 구현	팀원들과 공유자원 동기화 여부 테스트	GameRoom ::Check_GameState()

4주

11/23(일)	11/24(월)	11/25(화)	11/26(수)	11/27(목)	11/28(금)	11/29(토)
팀원 회의 및 점검.		GameRoom ::RemovePlayer() 구현			project progress report 작성	디버깅 및 테스트

5주

11/30(일)	12/01(월)	12/02(화)	12/03(수)	12/04(목)	12/05(금)	12/06(토)
최종 project progress report 작성			최종 통합 및 시연 준비.			

6주

12/07(일)	12/08(월)	12/09(화)	12/10(수)	12/11(목)	12/12(금)	12/13(토)

김영규

1주

11/02(일)	11/03(월)	11/04(화)	11/05(수)	11/06(목)	11/07(금)	11/08(토)
클라이언트 로컬 연산 제거 및 서버 권한 구조 전환		클라이언트 NetworkThread			project progress report 작성	

2주

11/09(일)	11/10(월)	11/11(화)	11/12(수)	11/13(목)	11/14(금)	11/15(토)
SendQueue		RecvQueue		팀원들과 동시 접속 테스트.	코드 리뷰, project progress report 작성	

3주

11/16(일)	11/17(월)	11/18(화)	11/19(수)	11/20(목)	11/21(금)	11/22(토)
CGameFramework::HandlePacket()					팀원들과 공유자원 동기화 여부 테스트	

4주

11/23(일)	11/24(월)	11/25(화)	11/26(수)	11/27(목)	11/28(금)	11/29(토)
팀원 회의 및 점검.					project progress report 작성	디버깅 및 테스트

5주

11/30(일)	12/01(월)	12/02(화)	12/03(수)	12/04(목)	12/05(금)	12/06(토)
최종 project progress report 작성			최종 통합 및 시연 준비.			

6주

12/07(일)	12/08(월)	12/09(화)	12/10(수)	12/11(목)	12/12(금)	12/13(토)

한건우

1주

11/02(일)	11/03(월)	11/04(화)	11/05(수)	11/06(목)	11/07(금)	11/08(토)
			Player() class 구현	GameRoom ::Move()	project progress report 작성	

2주

11/09(일)	11/10(월)	11/11(화)	11/12(수)	11/13(목)	11/14(금)	11/15(토)
	GameRoom ::Rotate()		GameRoom::Update_STate() 구현	팀원들과 동시 접속 테스트.	코드 리뷰, project progress report 작성	

3주

11/16(일)	11/17(월)	11/18(화)	11/19(수)	11/20(목)	11/21(금)	11/22(토)
	GameRoom ::Fire()		GameRoom ::Raycast()		팀원들과 공유자원 동기화 여부 테스트	

4주

11/23(일)	11/24(월)	11/25(화)	11/26(수)	11/27(목)	11/28(금)	11/29(토)
팀원 회의 및 점검.		Player 지형 충돌처리			project progress report 작성	디버깅 및 테스트

5주

11/30(일)	12/01(월)	12/02(화)	12/03(수)	12/04(목)	12/05(금)	12/06(토)
최종 project progress report 작성			최종 통합 및 시연 준비			

6주

12/07(일)	12/08(월)	12/09(화)	12/10(수)	12/11(목)	12/12(금)	12/13(토)