**네트워크 프로그래밍  
Term Project 추진계획서**

2020182042 최준하  
2020182004 고태경  
2020180026 이상유

**목차**

1. 애플리케이션 기획

2. High-Level 디자인

3. Low-Level 디자인

4. 팀원 별 역할 분담

5. 개발 환경

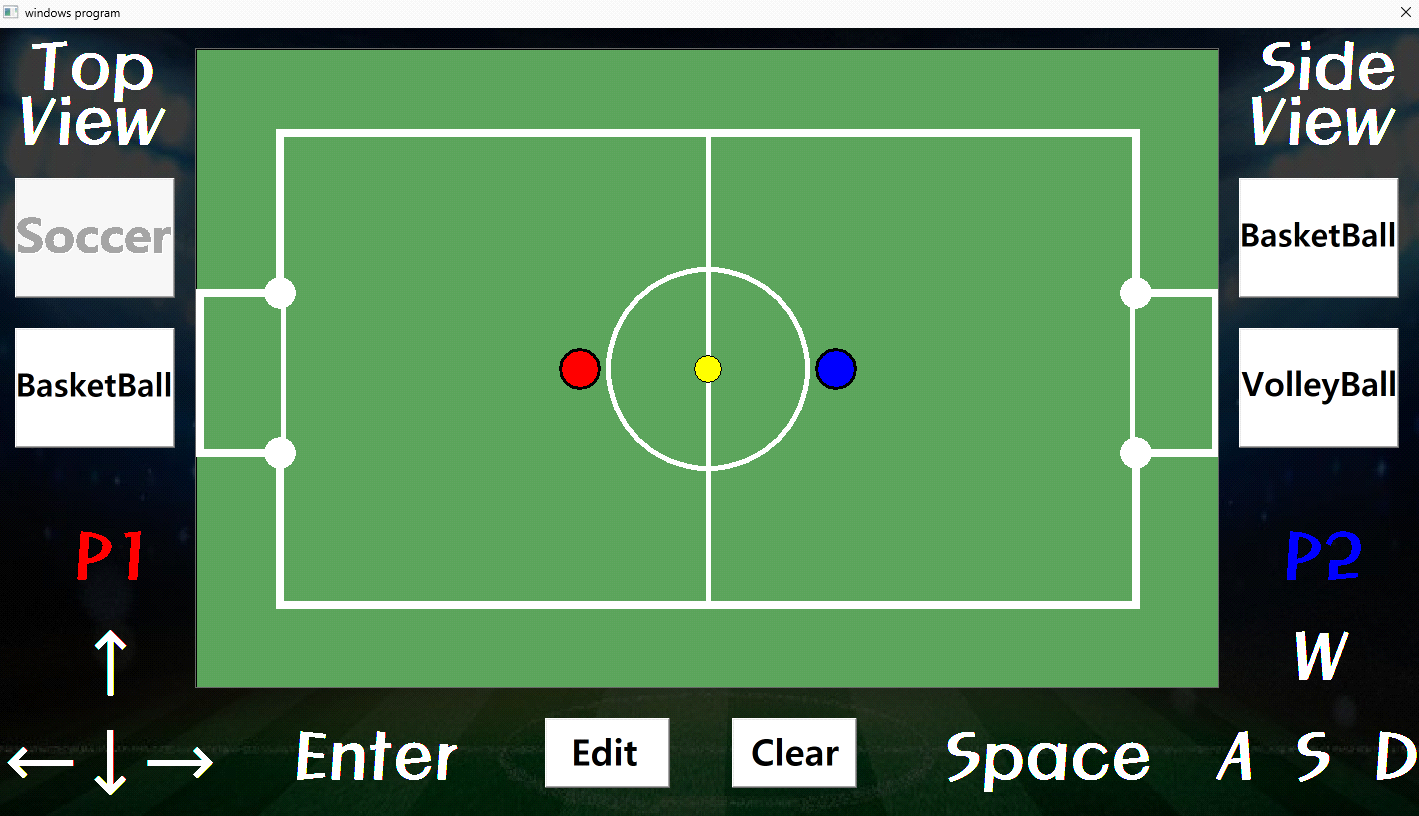
6. 개인 별 개발 일정

**애플리케이션 기획**  
고태경 학우가 2021-1 윈도우 프로그래밍에서 C++로 제작한 ‘Hit Ball’ 게임을 이용하여 프로젝트를 제작할 예정입니다.  
  
**게임소개**  
해당 게임은 ‘HaxBall’ 이라는 게임을 바탕으로 제작한 게임입니다.  
n대m 게임이며 방향키로 조작하여 상대방 골대에 골을 넣으면 이기는 게임입니다.  
탑뷰 시점으로 축구와 농구 2가지 종류를 플레이 할 수 있습니다.

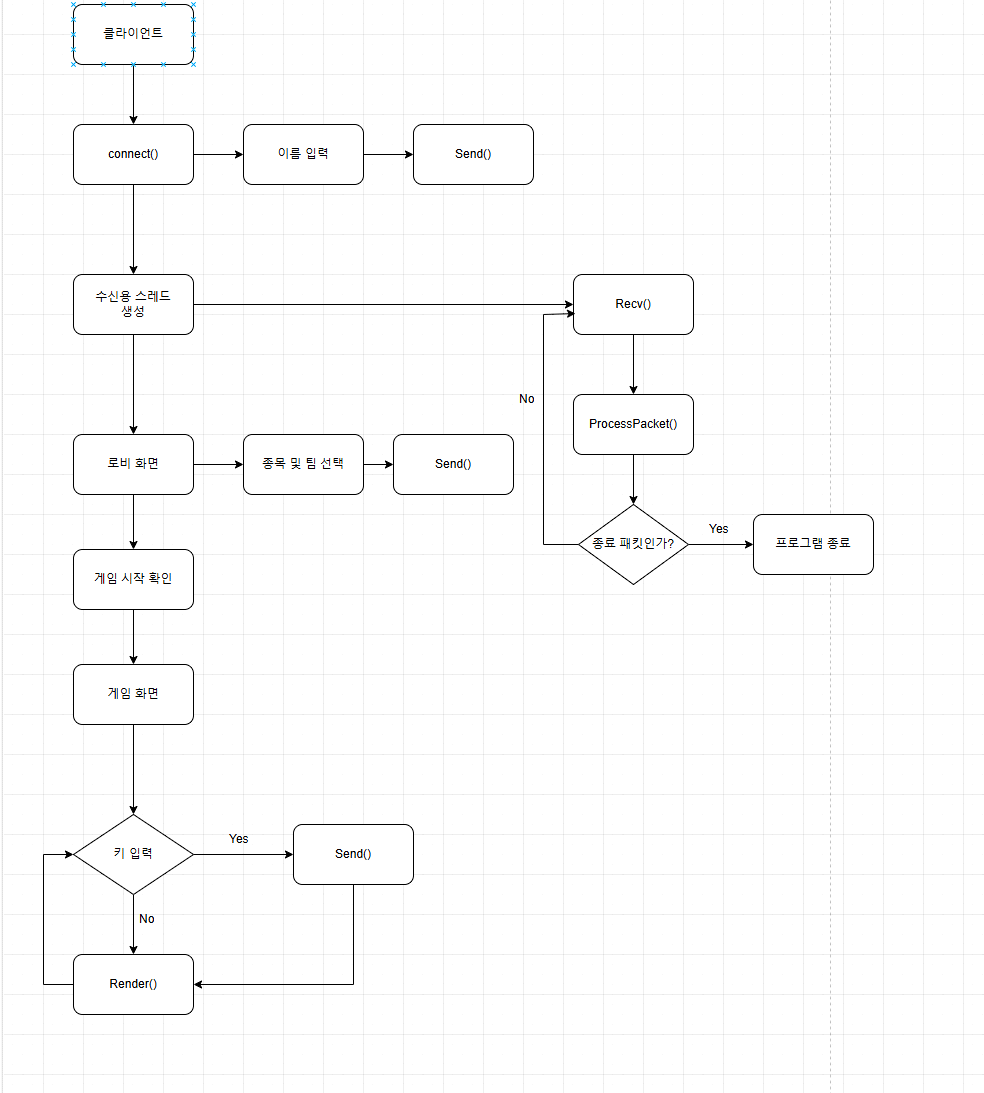
**조작 방법**  
방향키로 이동하며, 스페이스바를 누르면 슈팅을 할 수 있습니다.

**인게임 스크린샷(예시)**

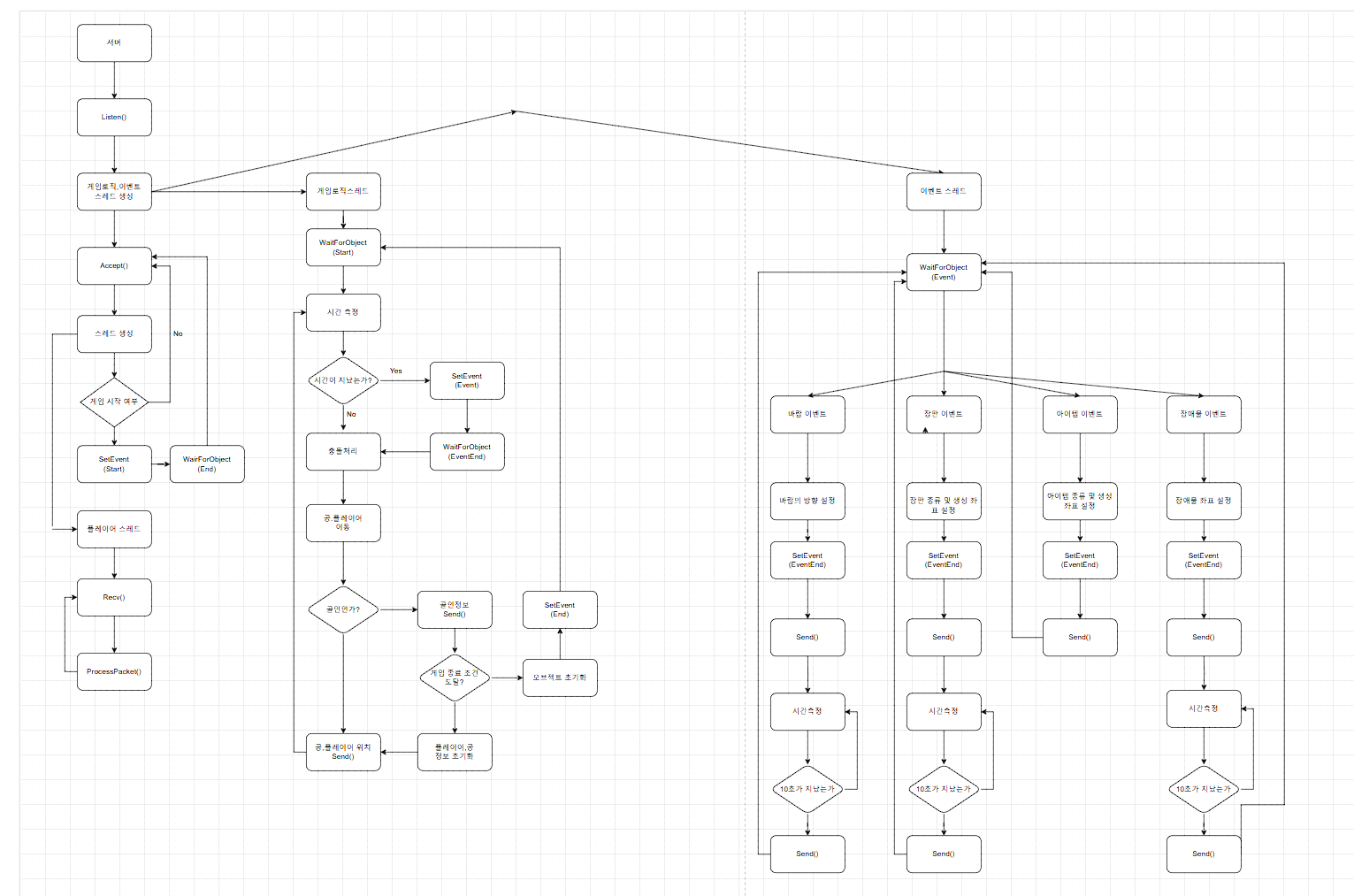
로비  
인게임

한 프로그램에서 2명의 플레이어가 동작하는 것을 네트워크를 활용하여 제작할 예정입니다.

**High-Level 디자인 (클라이언트)**



**High-Level 디자인 (서버)**



**Low-Level 디자인 (클라이언트)**

Void PacketThread()  
{

Bool DoRecv(); // 패킷 수신

Void ProcessPacket(char\* packet, int id); // 패킷 처리

}

// 이름 패킷 전송

bool SendNamePacket();

// 맵 선택 패킷 전송

bool SendMapChoicePacket();

// 팀 선택 패킷 전송

bool SendTeamChoicePacket(E\_TeamColor teamcolor);

// 키 패킷 보내기

bool SendKeyPacket(short keynum);

// 게임 시작 패킷 전송

bool SendStartGamePacket();

**Low-Level 디자인 (서버)**

Void PlayerThread(SOCKET player\_s)  
{  
 bool DoRecv(); // 패킷 수신  
 void ProcessPacket (char\* packet, int id); // 패킷 처리  
}

Void GameLogicThread()  
{  
void CheckCollision(); // 충돌 처리  
void MoveObject(); // 오브젝트 이동  
bool IsGameEnd(); // 게임 종료 조건 체크  
void ResetGameInfo(); // 게임 초기화  
void ResetPosition(); // 위치 초기화  
}

Void EventThread()  
{

Void WindEvent(); // 바람의 방향 설정  
 void FloorEvent(); // 장판 종류 및 생성 좌표 설정  
 void ItemEvent(); // 아이템 종류 및 생성 좌표 설정  
 void ObstacleEvent(); // 장애물 좌표 설정

}  
bool SendBallPositionPacket(Point pos); // 볼 위치 전송  
bool SendPlayerPositionPacket(Point pos); // 플레이어 위치 전송  
bool SendGoalPacket(); // 골인 정보 전송  
bool SendPlayerInfoPacket(CPlayer player); // 플레이어 정보 전송  
bool SendPlayerLogoutPacket(int id); //플레이어 로그아웃 전송  
bool SendStartGamePacket(); // 게임시작 전송  
bool SendMapPacket(E\_Map ); // 맵 종류 전송  
bool SendEventOnPacket(E\_EVENT ); // 이벤트 켜짐 전송  
bool SendEventOffPacket(E\_EVENT ); // 이벤트 꺼짐 전송

스레드 동기화

클라이언트  
메인스레드의 Render와 수신 스레드의 ProcessPacket이 같은 좌표라는 자원을 사용하고 있으므로 임계영역을 사용하여 동기화 할 예정입니다.

서버  
플레이어 스레드의 ProcessPacket과 게임로직 스레드의 충돌처리 함수에서 같은 플레이어 정보를 사용하고 있으므로 임계영역을 사용하여 동기화 할 예정입니다.

이벤트 스레드는 평소에 Wait\*() 함수를 통해 대기 상태에 있다가 게임로직스레드에서 시간을 측정하여 일정시간이 지날 때만 SetEvent()함수로 깨우도록 설계하였습니다.

이벤트가 시작하고 난 후 다시 게임로직 스레드에서 충돌처리와 이동을 처리하도록 설계하였습니다.

팀원 별 역할 분담  
최준하: 서버 프레임워크 구현, 플레이어 메인 스레드 구현  
고태경: 클라이언트 프레임워크 구현, 공 처리 관련 스레드 구현  
이상유: 클라이언트 추가 기능 구현, 클라이언트 -> 서버 네트워크 기능 구현

개발 환경

언어: C++  
개발도구: Visual Studio 2022  
버전관리 툴: GitHub  
운영체제: Windows

**개발 일정**

최준하

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 11/1 | 2 |
| 3  부분 통과로 인한 추진계획서 수정 | 4  부분 통과로 인한 추진계획서 수정 | 5 | 6  Listen() accept() 구현 | 7 | 8  클라이언트 처리 담당 스레드 생성 DoRecv() 구현 | 9  Class Packet  (Base,Name, Team, Map) 제작 |
| 10  ~~1주차~~  ~~피드백~~ ProcessPacket() (Team)  구현 | 11 | 12 | 13  SendPlayer TeamPacket()  구현 | 14 | 15  ProcessPacket() (Map) 구현  SendPlayer TeamPacket()구현 (연기) | 16 SendMapPacket() 구현 |
| 17  2주차  피드백 | 18  ProcessPacket() (name) 구현 | 19 | 20  SendNamePacket() 구현 | 21 | 22  Class Packet (Key, Start) 제작 | 23  ProcessPacket() (Start) 구현 |
| 24  3주차  피드백 | 25  Class Packet (Pos) ProcessPacket() (Key) 제작 | 26  SendStart GamePacket() SendPosPacket()  구현 | 27 | 28 | 29  Class Packet (Event) 제작 부족한 부분 수정 | 30  SendEvent Packet()  구현 |
| 12/1  4주차  피드백 | 2  Start\_event 구현 | 3  End\_event 구현 | 4 | 5 | 6  Event\_event 구현 | 7 |
| 8  최종  피드백 | 9 | 10 | 11  최종 점검 | 12 | 13 |  |

고태경

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 11/1  Player Class 설계 | 2  Object(Ball, Goalpost) class 설계 |
| 3  Map, Scene  Class 설계 | 4  Player의  이동, kick  입력 처리  (Play  Scene) | 5 | 6 | 7 | 8  Timer  추가 | 9 |
| 10  ~~1주차~~  ~~점검 및~~  부족한  진도 보강 | 11 | 12 | 13  Event class 설계  -> 10/14 | 14  Input Manager  (Start  Scene) | 15  Input Manager  (Lobby  Scene) | 16  Input Manager  (Play  Scene) |
| Event class  설계 (연기) |
| 17  2주차  점검 및  부족한  진도 보강 | 18  Client Manager  (Start  Scene) | 19  Client Manager (Lobby  Scene) | 20  Input Manager  (Lobby  Scene)(연기) | 21  Client  Manager  (Play  Scene) | 22 | 23 |
| 24  3주차  점검 및  부족한  진도 보강 | 25  Client Framework  (PlayScene  입력 처리) | 26  Client Framework (PlayScene  Update) | 27  Client Framework (PlayScene Render) | 28  Client Framework  (Start  Scene  입력 처리) | 29  Client Framework  (Lobby  Scene)  입력 처리) | 30 |
| 12/1  4주차  점검 및  부족한  진도 보강 | 2  Server에서  Collision Check  (Ball, Map)  구현 | 3 | 4  Server에서  Collision  Update  구현 | 5  SendBall  Position  Packet  (position) | 6  Send  Player  Update  Packet (position, velocity) | 7 |
| 8  5주차  점검 및  부족한  진도 보강 | 9 | 10 | 11  최종 점검 | 12 | 13 |  |

이상유

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 11/1 | 2 |
| 3 | 4 | 5 | 6 | 7  로비화면  추가 | 8 | 9  팀 선택,  종목 선택  추가 |
| 10  ~~1차~~  ~~피드백~~ | 11 | 12 | 13 | 14  바람 이벤트 구현  (12/5) | 15  Connect  함수 구현 | 16 |
| 17  2차  피드백 | 18 | 19  종목 선택 Send()  구현 | 20 | 21  수신용 스레드 생성  및 Recv()  구현 | 22 | 23  팀 선택 Send()  구현 |
| 24  3차  피드백 | 25 플레이어  이름  Send함수 구현 | 26  수신용 스레드 Update 구현 | 27 | 28  아이템 이벤트 구현 | 29  장판 이벤트 구현 | 30 |
| 12/1  4차  피드백 | 2  장애물 이벤트 구현 | 3  Process  Packet  볼 처리 관련 구현 | 4 | 5  바람 이벤트 구현 | 6 | 7 |
| 8  최종 피드백 | 9 | 10 | 11  최종 점검 | 12 | 13 |  |