**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

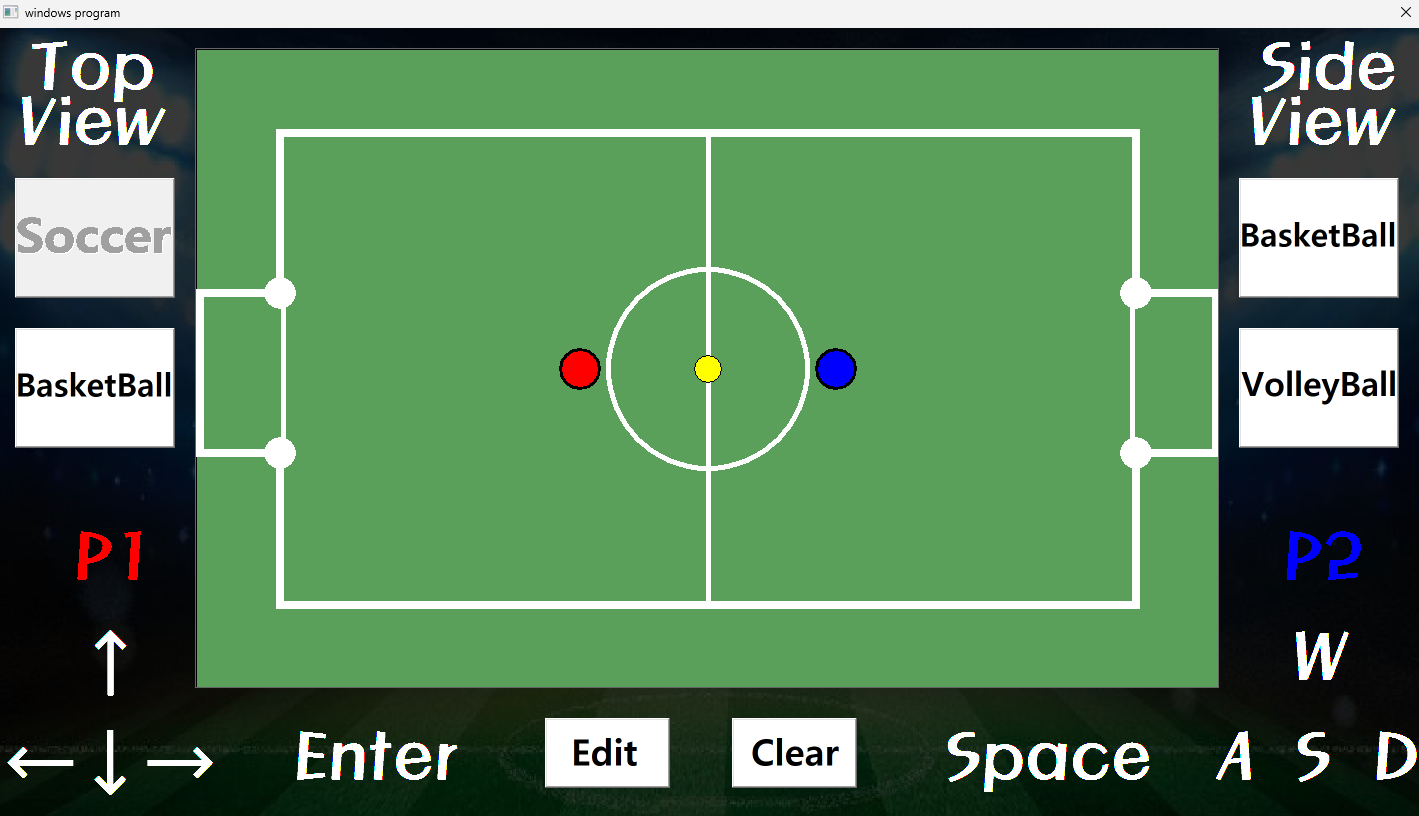
6. 개인 별 개발 일정

**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.

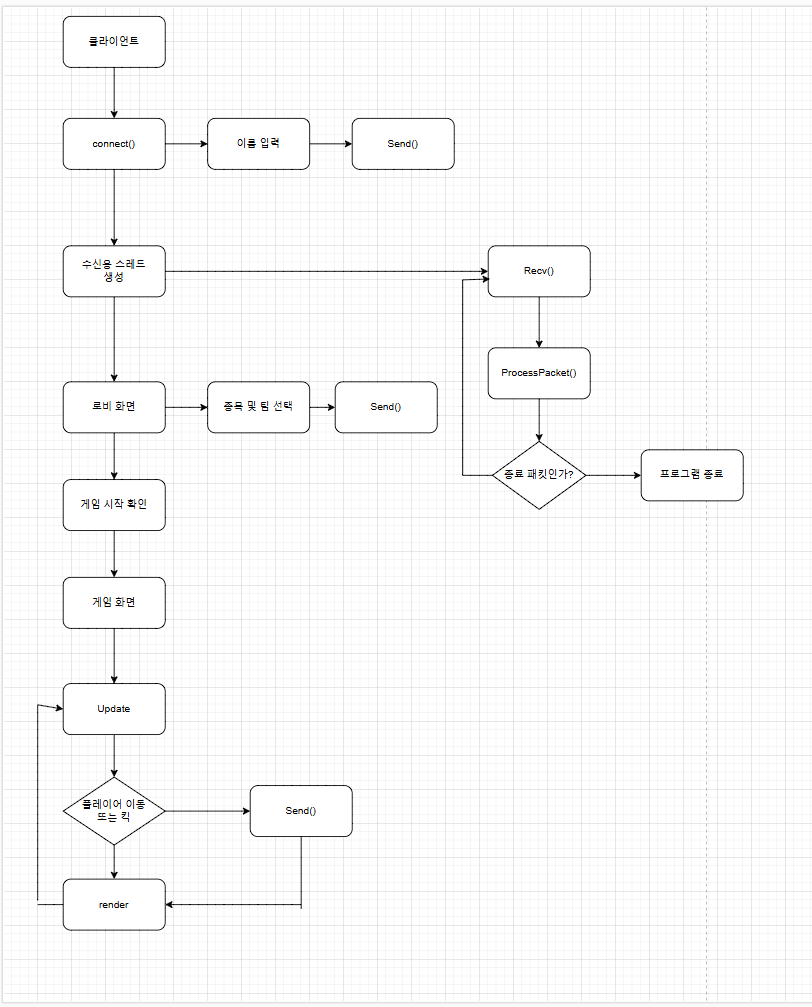
**조작 방법**  
방향키로 이동하며, 스페이스바를 누르면 슈팅을 할 수 있습니다.

**인게임 스크린샷(예시)**

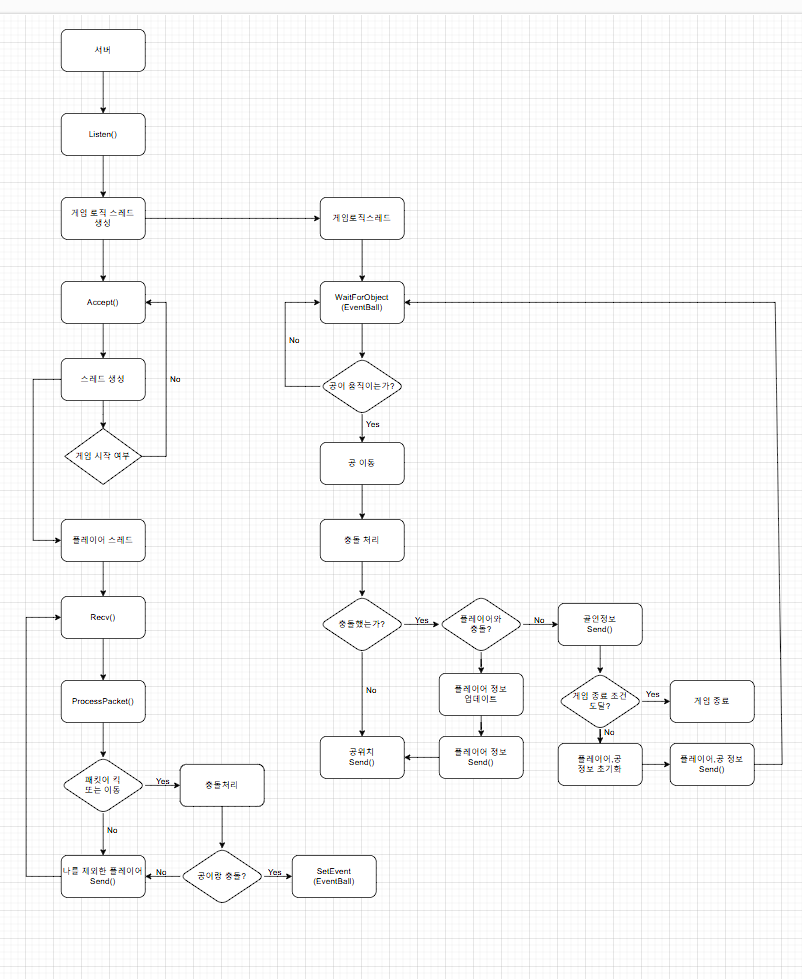
로비  
인게임

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

**High-Level 디자인 (클라이언트)**



**High-Level 디자인 (서버)**



**Low-Level 디자인 (클라이언트)**

// 서버에서 패킷 수신

bool DoRecv();

// 수신한 패킷에 따라 처리

void ProcessPacket (char\* packet,int id);

// 이름 패킷 전송

bool SendNamePacket();

// 맵 선택 패킷 전송

bool SendMapChoicePacket();

// 팀 선택 패킷 전송

bool SendTeamChoicePacket(E\_TeamColor teamcolor);

// 이동할 때 마다 좌표값 및 속력 전송

bool SendPlayerUpdatePacket(Point pos, Point velocity);

// 킥 패킷 전송

bool SendPlayerKickPacket();

// 게임 시작 패킷 전송

bool SendStartGamePacket();

**Low-Level 디자인 (서버)**

// 클라에서 패킷 수신

bool DoRecv();

// 수신한 패킷에 따라 처리

void ProcessPacket (char\* packet,int id);

// 플레이어 위치 전송

bool SendPlayerUpdatePacket(Point pos, Point velocity);

// 볼 위치 전송

bool SendBallPositionPacket(Point pos);

// 골인 정보 전송

bool SendGoalPacket();

// 플레이어 정보 전송

bool SendPlayerInfoPacket(CPlayer player);

// 플레이어 로그아웃 전송

bool SendPlayerLogoutPacket(int id);

// 게임시작 전송

bool SendStartGamePacket();

// 맵 종류 전송

bool SendMapPacket(E\_Map );

// 충돌처리

void CheckCollision();

// 공 이동

bool MoveBall();

// 공을 찼는지 확인하는 조건

bool IsKickBall();

// 게임 끝나는 조건

bool IsGameEnd();

**Low-Level 디자인 (패킷, 구조체)**

constexpr short BUFSIZE = 1024;

constexpr char NAMELEN = 128;

// 클라이언트에서 서버로 보내는 패킷 타입

constexpr char CS\_LOGIN = 0;

constexpr char CS\_MOVE = 1;

constexpr char CS\_KICK = 2;

constexpr char CS\_TEAMCHOICE = 3;

constexpr char CS\_MAPCHOICE = 4;

constexpr char CS\_START =5;

// 서버에서 클라이언트로 보내는 패킷 타입

constexpr char SC\_PLAYERDATA = 1;

constexpr char SC\_PLAYERPOS = 2;

constexpr char SC\_BALLPOS = 3;

constexpr char SC\_GOALIN = 4;

constexpr char SC\_START = 5;

constexpr char SC\_LOGOUT = 6;

constexpr char SC\_MAPTYPE = 7;

열거자

enum E\_TeamColor { Red, Blue };

enum E\_Map { Soccer, BasketBall };

struct XY

{

double x;

double y;

}

// 패킷 종류

class BasePacket{

unsigned char size;

char type;

int id;

};

class LoginPacket : public BasePacket {

char\* name[NAMELEN];

};

class LogoutPacket : public BasePacket {

};

class PosPacket : public BasePacket {

XY pos;

XY velocity;

};

class TeamPacket : public BasePacket {

E\_TeamColor color;

};

class MapPacket : public BasePacket {

E\_Map map;

};

class KickPacket : public BasePacket {

};

class StartPacket : public BasePacket {

};

class GoalinPacket : public BasePacket {

};

스레드 동기화  
서버  
 플레이어 스레드 충돌처리 함수와 게임 로직 스레드 에서 공의 이동처리 함수가 같은 공의 좌표를 사용하므로 임계영역을 활용하여 동기화 하였습니다.

클라이언트

ProcessPacket 함수에서 플레이어의 좌표나 공의 좌표를 활용하는 경우  
메인 스레드의 Update와 Render에서 같은 자원을 사용하므로 임계영역을 활용하여 동기화 하였습니다.

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 일 | 월 | 화 | 수 | 목 | 11/1 금 | 2 토  데이터 베이스 과제 |
| 3  Listen 소켓 생성 | 4  서버 패킷  수신 구현  accept | 5  인공지능 시험 준비 | 6  클라이언트 처리 담당 스레드 생성 | 7 | 8  DoRecv() 구현 | 9  ProcessPacket() (Name, Start)  구현 |
| 10  1주차  피드백 | 11  데이터베이스 시험 준비 | 12  데이터베이스 시험 준비 | 13  Class Packet  (Base,Name, Team, Start) 제작 | 14 | 15 SendPlayer NamePacket()  구현 | 16 SendStart GamePacket()  구현 |
| 17  2주차  피드백 | 18  ProcessPacket() (Team) 구현 | 19  SendPlayer TeamPacket() 구현 | 20 | 21 | 22  Class Packet (PlayerPos, Map) 제작 | 23  ProcessPacket() (Map) 구현 |
| 24  3주차  피드백 | 25  ProcessPacket() (Map) 구현 | 26  SendMapPacket() 구현 | 27 | 28 | 29  ProcessPacket() (PlayerPos) 구현 | 30 |
| 12/1  4주차  피드백 | 2  IsKickBall() 구현 | 3  Class Packet (BallPos, Goalin) 패킷 제작 | 4 | 5 | 6  SetEvent WaitForObject 제작 | 7  게임 시작 후  플레이어 접속 처리 구현 |
| 8  최종  피드백 | 9  코드 리팩토링  및 버그 수정 | 10  코드 리팩토링 및 버그 수정 | 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  오픈소스  중간고사  준비 | 7 | 8  Timer  추가 | 9 |
| 10  1주차  점검 및  부족한  진도 보강 | 11  데이터베이스,  오픈소스 이론  시험 준비 | 12 | 13 | 14  Input Manager  (Start  Scene) | 15  Input Manager  (Lobby  Scene) | 16  Input Manager  (Play  Scene) |
| 17  2주차  점검 및  부족한  진도 보강 | 18  Client Manager  (Start  Scene) | 19  Client Manager (Lobby  Scene) | 20 | 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  코드  최적화 및  리팩토링 | 10  코드  최적화 및 리팩토링 | 11  최종 점검 | 12 | 13 |  |

이상유

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 11/1  다렉 과제 준비 | 2  다렉 과제 준비 |
| 3  다렉 과제 준비 | 4  다렉 과제 준비 | 5 | 6  오픈소스중간고사 준비 | 7  로비화면  추가 | 8 | 9  팀 선택  추가 |
| 10  1차  피드백 | 11  오픈소스  이론 시험  준비 | 12  데이터베이스 중간 고사 준비 | 13 | 14  종목 선택  추가 | 15  Connect  함수 구현 | 16 |
| 17  2차  피드백 | 18 | 19  플레이어이름Send함수 구현 | 20 | 21  수신용 스레드 생성 | 22 | 23  수신용 스레드 Recv() 구현 |
| 24  3차  피드백 | 25 | 26  수신용 스레드 Update 구현 | 27 | 28  Process  Packet 플레이어 관련 구현 | 29  종목 선택  Send()  구현 | 30 |
| 12/1  4차  피드백 | 2 | 3  Process  Packet  볼 처리 관련 구현 | 4 | 5  팀 선택 Send()  구현 | 6 | 7 |
| 8  최종 피드백 | 9  디버그 및 버그 수정 | 10  코드 최적화 | 11  최종 점검 | 12 | 13 |  |