네트워크 게임 프로그래밍 추진계획서

# 1. 개요

• **기존 게임: *방과 후 축구한판* (컴퓨터 그래픽스 / 임성훈)  
• 서버 환경: Windows Server  
• 통신 방식: TCP/IP, UDP  
• 개발 환경: Visual Studio, C++, OpenGL**

# 2. 애플리케이션 기획

**방과 후 축구 한 판은 1대1 PVP 방식의 축구게임**으로서 각 플레이어는 물리 법칙이 존재하는 축구장에서 축구공을 찰 수 있으며, 드리블, 슈팅이 가능하다. (골키퍼는 AI)

게임 시작 전 **회원가입 그리고 로그인**을 해야 한다. **(ID/PASSWORD)**잔디, 야외, 들판, 식물이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**• 맵(World) 개요:  
상호작용하는 오브젝트는 총 5가지** 정도로  
**플레이어, 축구공, 축구장(큰 배경공간), 골대, 골키퍼**가 있다.

xz평면을 잔디로 채워 **축구장** 공간을 만들고 플레이어, 공, 골대, 골키퍼를 배치하였다.모든 오브젝트는 축구장 안에서 상호작용한다. 그리고 다른 오브젝트와 충돌한다.

카메라의 경우 플레이어와 골대를 바라보는 방향으로 화면에 담는다.

**축구공**은 움직일 때 회전한다.  
큰 박스 형태의 경계선을 만들고 경계선에 닿으면 공이 튕기도록 하여 공이 너무 먼 곳으로 날아가지 않게 한다.  
물리 법칙이 존재하기에 중력이 적용한다.

**골대**는 축구공과 플레이어와 충돌하며, 축구공의 경우 골대 안으로 들어오게 되면 골대 안에서 공이 멈춘다. 그리고 함성 사운드가 재생된다.

**골키퍼**는 축구공과 충돌하며, 축구공의 위치에 따라 움직이면서 골대를 방어한다.

**플레이어**는 드리블, 슈팅으로 축구공과 상호작용하고, 여러가지 기능을 통해 다양한 상호작용이 가능하다. **드리블의 경우 공과 가까워지면 자동으로 드리블**이 가능하다.

골을 넣은 플레이어에게 점수가 1점이 기록된다. 게임을 시작하고 5분이 지나면 게임은 종료되고, **승패판정 후 경기결과**가 나온다. 이후 로그인 정보를 통해 경기 전적이 기록되어, 다음 게임에서 **경기 전적**이 보이게끔 설정한다.

게임 도중 **채팅 기능**을 사용할 수 있다.

• 플레이어(Player) 개요:조작키는 다음과 같다.

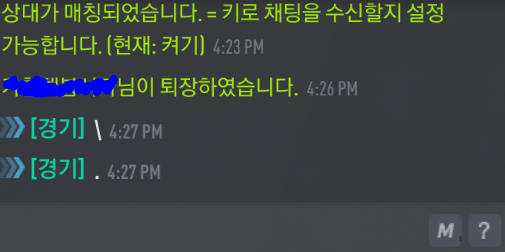
방향키: 플레이어의 기본적인 움직임이 가능하다. 공에 가까이 가면 드리블 상태가 되어 공을 드리블할 수 있다.

방향키: 이동  
E: 달리기  
D: 슛  
좌우 방향키와 슈팅을 조합하면 골대의 왼쪽 부분, 오른쪽 부분으로 찬다.  
Z와 조합하면 기본 슛이 아닌 감아차기가 나간다. (z, y 뿐 아니라 x도 바뀐다.)  
C와 조합하면 조금 더 낮고 강력한 슈팅을 발사한다.

**SPACE: 태클  
플레이어와 다른 플레이어의 위치, 공 소유 등을 계산 후 조건에 따라 공을 뺏을 수 있도록 한다.**



**Enter: 채팅 기능 사용  
채팅 UI를 제작하여 서로 채팅을 쓰고 볼 수 있도록 한다.**

****

# 3. 클래스 개요

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Player** | **Ball** | **Keeper** | **Goalpost** | **Camera** | **Light** |
| **vec3 velocity, direction,**  **float acceleration**  **bool hasball,[[1]](#footnote-1) keystates[][[2]](#footnote-2)** | **vec3 velocity, direction,**  **float acceleration**  **float rotationspeed** | **vec3 velocity**  **vec3 movement\_range[[3]](#footnote-3)**  **bool hasball** |  | **vec3 direction**  **vec3 up** | **vec3 color**  **vec3 viewpos**  **float ambientlight** |
| **Player(x, y, z)**  **~Player()**  **Move(), Draw()**  **Sprint(), Shoot() , Tackle()** | **Ball(x, y, z)**  **~Ball()**  **Move(), Draw()** | **Keeper(x, y, z)**  **~Keeper()**  **Move(), Draw()**  **Movement()** | **GoalPost(),**  **~GoalPost()**  **Draw()** | **Camera()**  **~Camera()** | **Light()**  **~Light()** |

|  |  |
| --- | --- |
| **Grond** | **Wall** |
|  |  |
| **Ground()**  **~Ground()**  **Draw()** | **Wall()**  **~Wall()**  **Draw()** |

|  |
| --- |
| **BackGround** |
|  |
| **BackGround();**  **~BackGround()**  **Draw()** |

|  |
| --- |
| **Object** |
| **glm::vec3 position{};**  **glm::vec3 rotation{};** |
| **Object();**  **~Object();** |

# 4. High Level Design

•**프로그램 흐름도**

FileSystem

F

2 ready?

connect

send(ID/PW)

recv

accept

Send(T/F, P1/2)

Listening

T

send(ready)

recv

Login?

F

T

MakeSession

recv

Update Score  
Position Reset

GameSession  
Destroy

recv\_Input

Input

key apply

Ball Calculate

Collision System

AI Goal keeper

Goal?

F

T

GameOver  
Win/Lose

Rendering

End?

F

T

Data send

UI Update  
Close

# 5. Low Level Design

## <프로토콜 설계>

• Server :

|  |  |  | **이름** | **데이터 (Payload)** | **목적 (Flowchart 매핑)** |
| --- | --- | --- | --- | --- | --- |
|  |  |  | recv\_IDPW | struct PacketLogin | 클라이언트로부터 ID와 PASSWORD값을 받음. |
|  |  |  | LoginResponse | bool bSuccess (1:성공, 0:실패)  int PlayerID, int Win, int Loss (성공 시)  string Message (실패 사유) | 로그인 요청에 대한 결과 응답. 5 |
|  |  |  | send\_login | struct PacketLoginResult | Login Response에서 반환된 bool 값을 클라이언트에 전송 |
|  |  |  | recv\_ready | struct PacketGameReady | 로그인에 성공한 클라이언트에서 받은 ready값을 반환 |
|  |  |  | 2ready | bool P1\_ready, P2\_ready | 두 플레이어에게 true값을 동시에 받을 때까지 ready값을 받는 것을 반복 |
|  |  |  | MatchFound | int GameUdpPort (게임용 UDP 포트) string OpponentID, int OpponentWins, int OpponentLosses | 매치가 성사되었음을 알리고, 접속할 **UDP 포트**와 상대 정보를 전달. |
|  |  |  | LoadGame | Vector P1\_StartPos, Vector P2\_StartPos... | 클라이언트에게 게임 맵 로드 및 오브젝트 초기 배치를 명령. |
|  |  |  | BroadcastChat | string SenderID, string Message | 채팅 메시지를 상대방에게 중계. 6 |
|  |  |  | recv\_input | int P1/P2\_keystates | 클라이언트에서 받은 input 값을 반환 |
|  |  |  | GameStateUpdate | uint SequenceNum Vector P1\_Pos, Vector P2\_Pos, Vector Ball\_Pos, Quaternion Ball\_Rot Vector GK1\_Pos, int ScoreP1, int ScoreP2 int SecondsRemaining | 서버가 물리/AI 연산을 마친 모든 동적 오브젝트의 최종 상태를 전송. Goal? 분기 로직을 반영하여, 점수와 시간을 이 패킷에 포함합니다. 클라이언트는 이 패킷을 받아 점수판(UI)을 갱신하고, 점수 변경을 감지하여 "골" 사운드를 재생합니다. (Update Score 로직 포함) 공과 플레이어의 위치도 골 직후 리셋된 위치가 포함되어 전송됩니다. |
|  |  |  | send\_playerdata | struct PacketPlayerData | 모든 연산 후 클라이언트에서 렌더링을 위한 데이터를 보내줌 |
|  |  |  | Player::Tackle() | uint SequenceNum Vector P1\_Pos, Vector P2\_Pos, Vector Ball\_Pos | 플레이어가 다른 플레이어의 공을 뺏을 수 있도록 기능 추가. |
|  |  |  | GameOver | time\_t start, end | 게임 종료 조건(time)이 만족하면 게임을 종료 |
|  |  |  | send\_gameover | struct PacketGameover | 게임이 끝나면 클라이언트에게 알림. |
|  |  |  | Update Score | string WinnerID  int FinalScoreP1, FinalScoreP2 | 게임 종료 후 경기 스코어에 따라 전적을 수정 |
|  |  |  | send\_score | struct PacketUserData | 수정한 전적을 파일 시스템에게 보냄. |
|  |  |  | ReturnToLogin | int NewWins, int NewLosses | 서버가 전적 파일 저장을 완료했음을 알리고, 로그인 씬으로 돌아가도록 명령. |

• Client :

| **이름** | **데이터 (Payload)** | **목적 (Flowchart 매핑)** |
| --- | --- | --- |
| LoginRequest | string ID, string Password | 회원가입 또는 로그인을 요청. |
| send\_IDPW | struct PacketLogin | 받은 ID와 PW값을 서버에 전송 |
| recv\_login | struct PacketLoginResult | 서버에서 받은 bool 값을 반환 |
| login | string ID, string Password | recv\_login에서 true를 받을 때까지 로그인 과정 반복 |
| ClientReady |  | 클라이언트가 로그인을 완료하고 매치를 찾을 준비가 되었음을 알림. |
| send\_ready | struct PacketGameReady | bool 변수 ready값이 변화할 때마다 서버에 전송 |
| PlayerChat | string Message (채팅 내용) | ENTER키 입력 시 채팅을 입력시킬 수 있는 UI가 나오고 채팅 메시지를 입력할 수 있음. |
| send\_chat | string Message (채팅 내용) | 입력된 채팅 메시지를 server에 보냄 |
| Ack\_GameOver | Bool IsReturnHome | "처음으로" 버튼을 눌렀음을 서버에 알림. |
| PlayerInput | uint SequenceNum InputBitmask (short) Vector LookDirection | 플레이어의 현재 키보드 입력 상태를 압축하여 전송. |
| recv\_gameover | struct PacketGameover | 서버로부터 Game이 끝났다는 bool값을 받음 |
| UI Update Close() |  | GameoverUI를 보여주며 로그인 화면으로 돌아갈 수 있도록 함 |

## <패킷 설계 예시>

**서버 -> 클라이언트**

struct PacketHeader {  
uint16\_t type; 패킷 종류 (ex. 로그인, 유저데이터, 키입력 등)  
uint16\_t size; 전체 패킷 크기  
};

struct PacketGameReady {  
PacketHeader header;  
uint8\_t ready  
};

struct PacketLoginResult {  
PacketHeader header;  
uint8\_t success;  
char message[64]; // 실패 시 이유 or 성공 메시지  
};

struct UserData {  
int totalMatch;  
int win;  
int lose;  
float winRate;  
};

struct PacketUserData {  
PacketHeader header;  
UserData data;  
};

struct PacketPlayerData {  
PacketHeader header;  
PlayerData data;  
};

struct PlayerData {  
int id;  
vec3 position;  
vec3 rotation;  
vec3 velocity;  
};

struct PacketGameover {  
PacketHeader header;  
uint8\_t gameover;  
};

**클라이언트 -> 서버**

struct PacketInput {  
PacketHeader header;  
uint32\_t key;  
};

struct PacketLogin {  
PacketHeader header;  
char userID[32];  
char userPW[32];  
};

# 6. 역할분담

**- A: 김용채**  
서버 통신, 네트워크 동기화

**- B: 박지성**  
클라이언트 UI, 멀티 스레드, 데이터 관리(filesystem)

**- C: 임성훈**  
채팅 UI, 게임 로직 구현, 패킷 구조 설계

# 7. 개발 일정

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | SUN | MON | TUE | WED | THU | FRI | SAT |
| **10** | 26 | 27 | 28 | 29 | 30 | 31  connect(), listening(), accept() | 1 |
| **11** | 2 | 3 | 4 | 5  send\_IDPW(), recv\_IDPW()  send\_login()  recv\_login() | 6 | 7 | 8 |
| 9 | 10  send\_ready(), recv\_ready(), 2ready? | 11 | 12 | 13 | 14 | 15  1차토의 피드백 후Project Progress Report 작성 |
| 16 | 17 | 18 | 19 | 20  recv\_input,() data\_send() | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 2차토의 |
| **12** | 30 | 1 | 2 | 3 | 4 | 5 | 6 3차토의 |
| 7 | 8 | 9 | 10  최종 테스트 및 결과물 제출 | 11 | 12 | 13 |

**A: 김용채**

**B: 박지성**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | SUN | MON | TUE | WED | THU | FRI | SAT |
| **10** | 26 | 27 | 28 | 29  ClientHandlerThread() | 30 | 31  RegisterUser() | 1 |
| **11** | 2 | 3 ValidateLogin() | 4 | 5 | 6 | 7  LoadLoginScene(); | 8 |
| 9 | 10 | 11 GameSessionThread() | 12 | 13 | 14  InitializeUDPSocket() RunGameLoop() | 15 1차토의 피드백 후Project Progress Report 작성 |
| 16 | 17 UdpReceiveThread() ProcessGameStateUpdate() | 18 | 19 LoadGameSceneUI() | 20 | 21  UpdateInGameUI() | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 2차토의 |
| **12** | 30 | 1 OnClick\_LoginScene () | 2 | 3 | 4 | 5 | 6 3차토의 |
| 7 | 8 | 9 | 10  최종 테스트 및 결과물 제출 | 11 | 12 | 13 |

**C: 임성훈**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | SUN | MON | TUE | WED | THU | FRI | SAT |
| **10** | 26 | 27 | 28 | 29  기존Code  Client/Server 나누기 | 30 | 31 | 1  PlayerInput() |
| **11** | 2 | 3 | 4 Gameover() | 5 | 6 | 7 | 8  send\_gameover()  recv\_gameover() |
| 9 | 10 | 11 Update Score() | 12  send\_score() | 13 | 14 | 15 1차토의 피드백 후Project Progress Report 작성 |
| 16 | 17 | 18 | 19  UI Update Close() | 20 | 21 | 22  Tackle() |
| 23 | 24 | 25 PlayerChat()  send\_chat() | 26  BroadcastChat() | 27 | 28 | 29  2차토의 |
| **12** | 30 ChatUI() | 1 | 2 | 3 | 4 | 5 | 6 3차토의 |
| 7 | 8 | 9 | 10  최종 테스트 및 결과물 제출 | 11 | 12 | 13 |

1. hasball: **볼 소유권** [↑](#footnote-ref-1)
2. keystates[]: **조작키 입력 값** [↑](#footnote-ref-2)
3. movement\_range: **골키퍼 움직임 반경** [↑](#footnote-ref-3)