Network Game Programming

Report

Team 10

이름

2015180040 채영문

* 목차

1. 애플리케이션 기획

* 게임 설명

1. High - Level Design
2. Low - Level Design

-----

1. 팀원 별 역할분담
2. 개발환경
3. 개발일정
4. 애플리케이션 기획



* 메인 화면 씬



* 게임 플레이 씬
* 게임이름 – 교과목 및 제작자

Power Ing (윈도우 프로그래밍 – 김연규)

* 게임목표

3명의 플레이어들이 오브를 튕겨 점수를 얻어 경쟁하는 게임

* 게임내용
* 각 플레이어는 메인 화면에서 모두 준비(레디)를 하면 애니메이션과 함께 게임이 시작
* 게임을 시작하면 오브가 움직이기 시작하고 플레이어들은 자신에게 부여된 색깔의 패널을 레일에 따라 움직인다.
* 패널을 움직여 오브를 튕기면 득점한다.
* 패널은 서로 부딪힐 수 있으며 겹치지 않는다.
* 일정 점수를 얻거나 오브가 경기장 밖으로 나가 오브가 모두 소모되면 게임을 종료한다.
* 종료 이후 우승자와 함께 모든 플레이어 점수를 알려준다.



* 게임조작

메인 씬

Ready: 레디 버튼 클릭하여 준비한다.

Quit: 게임을 종료한다.

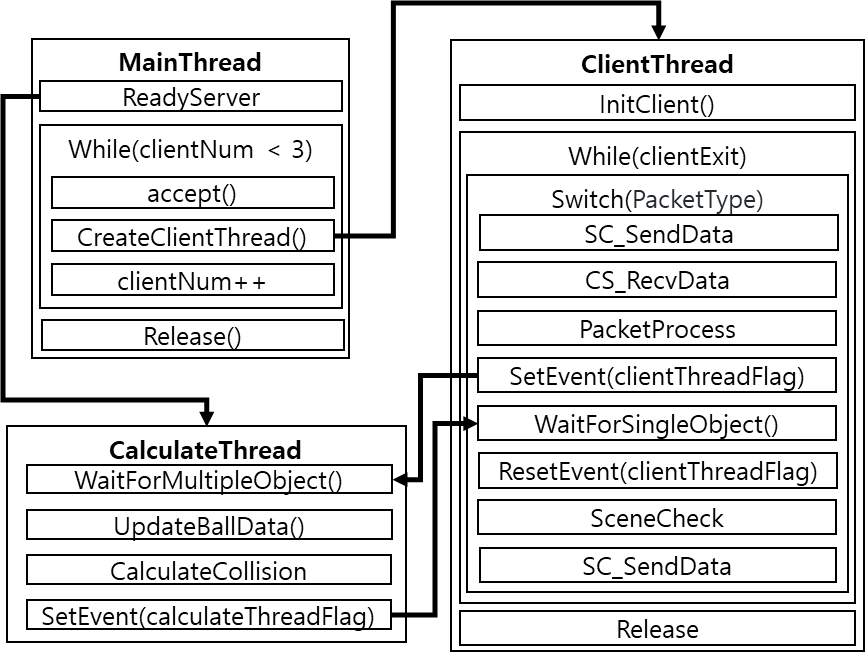
//옵션: 뺄 예정--

//모듈: 뺄 예정--

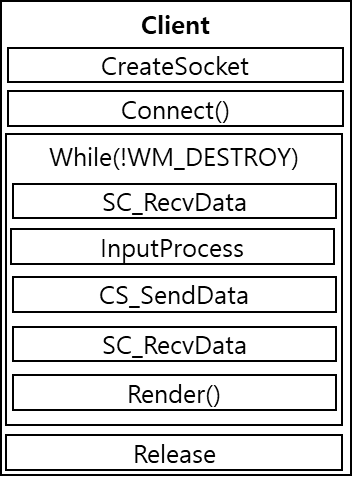
게임 플레이 씬

좌/우 방향키: 패널을 움직일 수 있다.

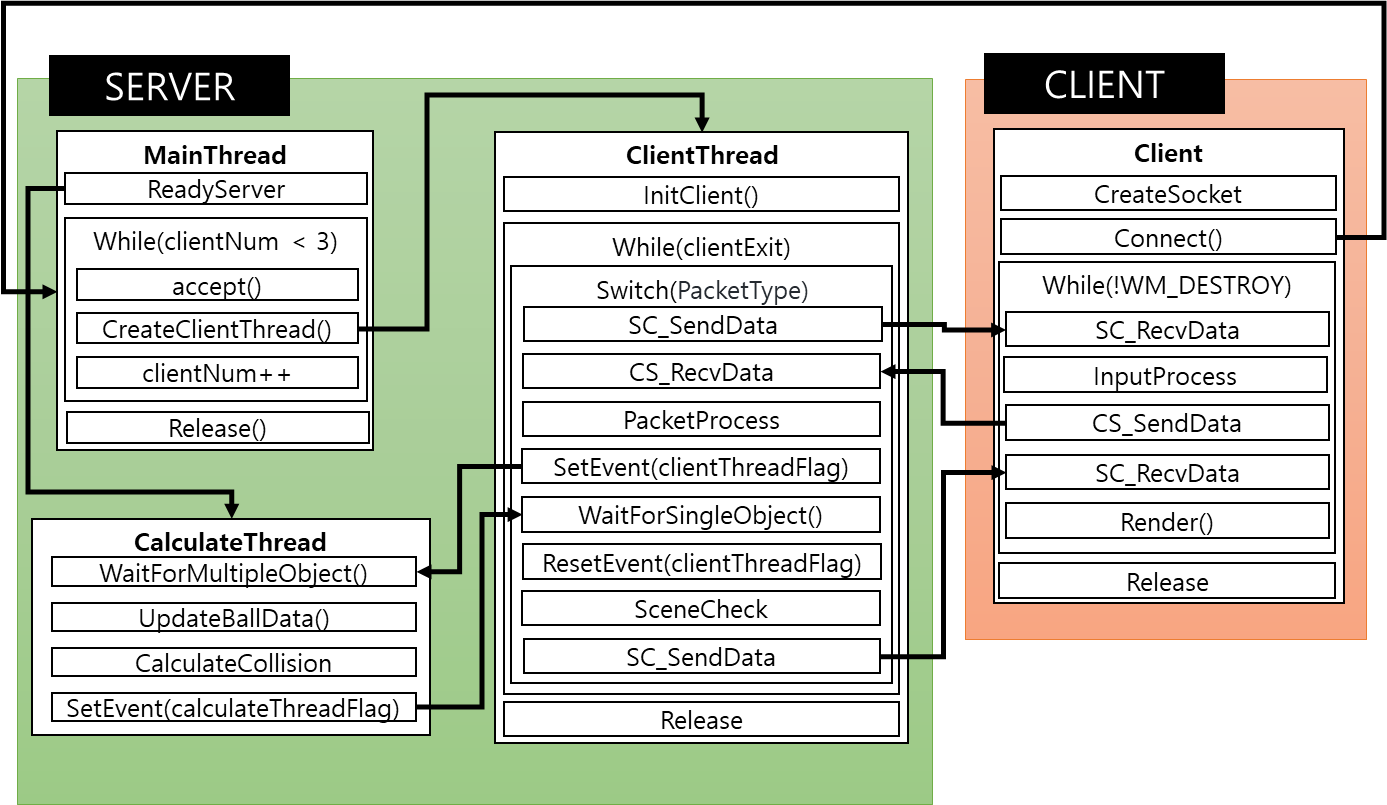
**2. High – Level Design**

**Server**

**Client**

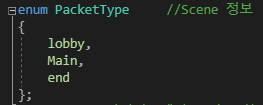


**Server-Client 관계도**

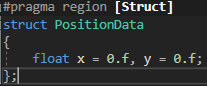


**3. Low – Level Design**

**3-1: Server & Client 공통 함수, 구조체**

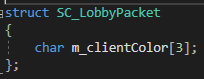


패킷 타입: 각 Scene의 이름으로 타입을 구별. 고정 길이로서 서버에서 Send()

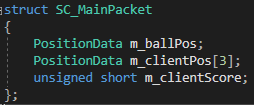


2D 클라이언트 기반 위치정보 구조체

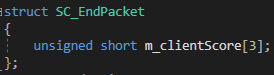
**3-2: Server 전용 구조체, 함수**

****

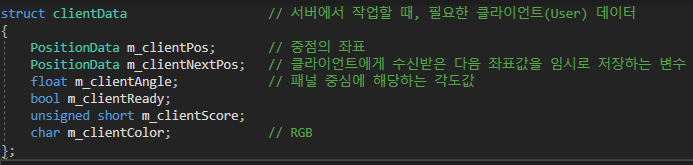
로비 Scene에서의 Server to Client 패킷 구조체: 클라이언트 ID



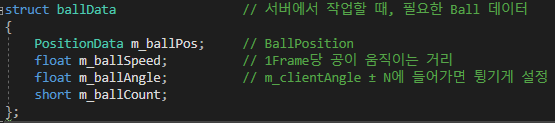
메인 Scene에서의 Server to Client 패킷 구조체: 공 위치, 패널 3개위치, 점수

****

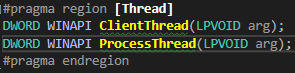
엔드 Scene에서의 Server to Client 패킷 구조체: 모든 플레이어 점수 송신



서버에서 관리하는 클라이언트 데이터 구조체, 컨테이너 타입 array[3], 전역변수로 관리



서버에서 관리하는 공 데이터 구조체, 전역변수로 관리



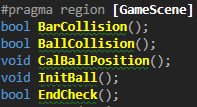
**ClientThread(LPVOID arg):** 클라이언트 통신 쓰레드, 클라이언트 연결 요청마다 호출

**ProcessThread(LPVOID arg):** 서버내의 모든 객체 위치 값 계산, 충돌처리 등 게임 내 Update를 위한 함수



**bool ReadyCheck():** 플레이어들의 준비 상태 값을 수신 받아 ReadyCount가 3이상인지 확인 후 MainScene으로 넘어갈 수 있는지 확인하는 함수

**void InitClient():** 클라이언트 접속시에 데이터(PID) 초기화



**bool BarCollision():** 플레이어 간 충돌 처리 함수. 충돌 시 True를 반환

**bool BallCollision():** 플레이와 공 간 충돌 처리 함수. 충돌 시 True를 반환

**void CalBallPosition():** 공의 다음 위치를 계산 후 이를 저장

**void InitBall():** 공 속도 및 각도(벡터) 초기화, 남은 공 개수 감소

**bool EndCheck():** 게임이 종료 여부 확인



**int SC\_SendData(SOCKET soc**k): 클라이언트로 패킷을 송신

**int CS\_RecvData(SOCKET sock):** 클라이언트로부터 패킷을 수신, 이후 패킷 데이터 저장

**void GeneralReset():** 공, 버튼, 타이머, 씬, 점수, 이펙트 타임 등 변수 Looby초기화

**Power\_Orb\* OrbPosition(Power\_Or**b\*): 속도에 따른 공의 다음 위치 계산

**Power\_Orb\* OrbSpeed(Power\_Orb\*):** 공 속도 계산

**Void OrbCreate(Power\_Orb\*, int, bool, float x, float y, float radian)**

**:** 객체 생성 초기화

**Void OrbRemove(Power\_Orb\* postOrb, Power\_Orb\* orb):** 공 1개 삭제

**Void OrbClear(Power\_Orb\* orb):** 공 전체 삭제 및 리셋

**Power\_Orb\* OrbApply(Power\_Orb\*, int Type, bool, float x, float y, float radian)**

: 공 생성 후 크기 값 초기화

**void CollisonDetect(Power\_Orb\* orb):** 공 충돌 처리 함수

**Power\_Orb\* ReflectReflectorOrb(Power\_Orb\*, Power\_Reflector\*)**

: 충돌 후 공 위치 값 조정

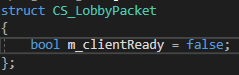
**Power\_Orb\* ReflectOrb(Power\_Orb\* orb, float angle)**

**:** 충돌한 공의 각도(방향) 조정

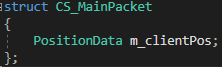
**void ReflectDetect(Power\_Orb\*, Power\_Reflecto**r\*): 패널 충돌 처리

**void ReflectReflector(Power\_Orb\*, Power\_Relector\*):** 패널 충돌 처리 후 부가작업(점수 계산, 이펙트 애니메이션) 처리 함수

**3-3: Client 전용 구조체, 함수**



로비Scene에서의 Client to Server 패킷 구조체: 클라이언트의 Ready값 송신



메인 Scene에서의 Client to Server 패킷 구조체: 클라이언트 Position값 송신

텍스트이(가) 표시된 사진

자동 생성된 설명

클라이언트 렌더링을 위한 클라 내 패널 정보 구조체

텍스트이(가) 표시된 사진

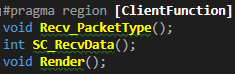
자동 생성된 설명

클라이언트 렌더링을 위한 클라 내 공 정보 구조체

텍스트이(가) 표시된 사진

자동 생성된 설명

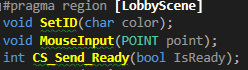
클라이언트 렌더링을 위한 클라 내 이펙트 관리 구조체



**void Recv\_PacketType():** 서버로부터 고정 길이 데이터인 패킷 타입을 받는 함수

**int SC\_RecvData():** 서버로부터 받은 가변 길이 패킷 데이터를 클라이언트 메모리에 저장하는 함수

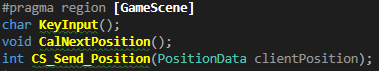
**void Render():** 메모리에 저장된 데이터로 이를 출력하는 함수



**void SetID(char color):** 서버로부터 부여 받은 클라이언트 ID를 저장하고, 이를 토대로 본인 색을 지정하는 함수

**void MouseInput(POINT point):** 마우스 입력값을 토대로 버튼 충돌처리 이후 클라이언트 상태값을 바꾸는 함수

**int CS\_Send\_Ready(bool IsReady):** 서버로 현재 클라이언트 상태정보를 Send하는 함수



**char KeyInput(char chInput):** 메인Scene에서 어떠한 키를 입력했는지 반환하는 함수

**void CalNextPosition();** 입력 값을 토대로 클라이언트 다음 위치 값을 갱신하는 함수

**int CS\_Send\_Position(PositionData clientPos):** 키가 눌렸다면 클라이언트의 다음 예상 위치 값을 전송하는 함수.

**3-4: 쓰레드 동기화**

이벤트를 사용하여 서버 프로그램 내의 쓰레드 간 동기화 구현 예정: