**네트워크 게임 프로그래밍**

**Term Project 추진 계획서**

**게임공학과 2015180040 채영문**

**게임공학과 2017180052 황지원**

**목차**

**1. 애플리케이션 기획**

**2. High-Level Design**

2-1. 클라이언트

2-2. 서버

**3. Low-Level Design**

3-1. 클라이언트

3-2. 서버

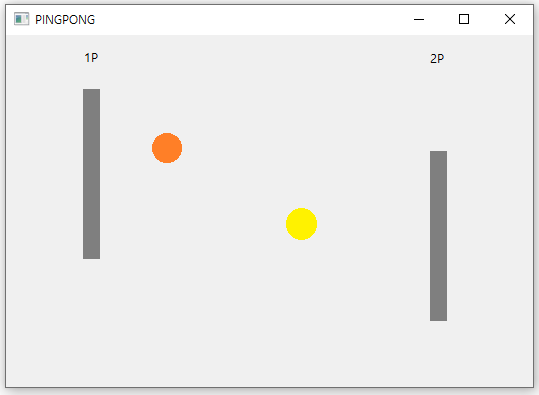
**4. 개발 환경 및 세부 기획**

4-1. 개발 환경

4-2. 역할 분담

4-3. 개발 일정

1. **애플리케이션 기획**

****

<예상 플레이 화면>

-게임 이름: 핑퐁게임

-게임 내용: 속도가 다른 두 개의 공을 주고받는 게임

-게임 조작:

↑ : 플레이어 위쪽 이동

↓ : 플레이어 아래쪽 이동

1. **High Level Design** 
   1. **High Level 클라이언트 구상안**

**A. 서버연결**

-프로그램 실행 시 윈도우 소켓 초기화 후 서버에 connect() 함수로 연결 요청

**B**. **게임플레이Scene**

-게임이 시작되면 각 플레이어는 update() 함수 호출 시마다 자신의 정보를 서버에 전달한다.

-서버에서 처리한 정보를 받아 공의 상태 및 상대 플레이어의 위치를 갱신하여 화면에 그린다.

-게임 종료 조건을 만족한다면 게임 결과 화면을 각 클라이언트에 렌더링.

**2.2 High Level 서버 구상안** – 멀티스레드를 이용한 데이터 통신

**A. 서버 연결**

-실행 후 클라이언트 접속 처리를 위한 소켓을 생성하여 서버의 IP주소와 포트번호를 bind() 함수로 설정한다.

-클라이언트 접속 대기를 위한 소켓의 TCP포트 상태를 listen() 함수를 사용하여 listening 상태로 바꿔준다.

-클라이언트가 connect() 함수를 통해 접속을 요청하면, 이를 처리할 소켓을 생성한다.

-그 이후 각 클라이언트 패킷을 처리할 쓰레드를 생성한다.

**B. 게임 플레이**

-먼저 들어온 플레이어는 상대 플레이어를 기다린다

-클라이언트 접속이 완료되면 게임 시작 패킷 전송

-각 클라이언트는 Update함수가 호출될 때 마다 보내는 정보를 recv() 함수로 수신한다.

-처리할 내용: 공-플레이어 충돌처리, 각 플레이어의 위치 정보, 공의 상태 갱신

-클라이언트에게 받은 정보를 처리해서 send() 함수로 송신한다.

-한쪽이 공을 놓치면 모든 클라이언트에게 게임 종료에 대한 정보를 send() 함수로 송신한다.

1. **Low Level Design**

**3-1 Low Level 클라이언트 구상안**

1. **TCP 클라이언트-서버 통신을 위한 기본 함수**
   * void err\_quit(const char\* msg)
     1. const char\* msg : 문자열을 받아올 인자
     2. 현재 발생한 오류 메시지를 메시지 박스로 표시하고, 프로그램을 종료하는 함수
   * void err\_display(const char\* msg)
     1. const char\* msg: 문자열을 받아올 인자
     2. 통신 중 에러가 발생한 경우 해당 오류 메시지를 메시지 박스 대신 printf()로 출력하고, 프로그램을 종료하지는 않는다.
   * void recvn(SOCKET \_s, const char\* \_chBuf, int \_ilen, int \_iFlag)
     1. recv() 함수는 TCP 프로토콜의 특징 상 데이터 경계를 구분하지 않는다. 이 때문에 클라이언트가 서버로부터 자신이 보낸 len값의 크기만큼 데이터를 받기 전까지 대기시간이 생기는데, recvn() 사용자 정의 함수를 통해 recv() 함수를 지속적으로 호출하여 이를 보완할 수 있다.
   * void init\_Client\_Socket()
     1. 사용할 변수
        + - WSADATA wsa: 클라이언트에서 사용할 윈도우 소켓
          - SOCKET sock: TCP 서버와 통신할 TCP 소켓
          - SOCKADDR\_IN serverAddr: 통신할 TCP 서버의 주소 체계와 IP주소, 포트번호를 담고 있는 소켓 주소 구조체
          - HANDEL bReadEvent, bWriteEvent : 스레드 동기화와 순서제어를 위한 신호/비신호 상태의 이벤트 두 개를 생성한다.
     2. 서버와의 통신에 관련된 변수들을 초기화한다
     3. 통신을 위한 기본적인 기능을 담고 있다

**B. 클라이언트 공통 함수**

1. void update(): 클라이언트 내 업데이트 작업을 수행 후, sendData() 함수와 recvData() 함수를 호출하여 서버와 통신한다.
2. void render(): 프레임별 Scene을 그리는 함수
3. void sendData(): 각 클래스에서 정의한 데이터 구조체를 송신하기 위한 함수
4. void recvData(): 각 클래스에서 정의한 데이터 구조체를 수신하기 위한 함수

**C. 게임 플레이 Scene**

* + **송신할 데이터 구조체[cs\_packet\_mainGame]**
    1. int iPlayerID: 각 클라이언트 구별용 ID
    2. Point2D p2dPosition: 플레이어의 위치 정보
  + **중심 함수**
    1. void render(): 화면을 렌더링한다.
    2. void update():
       - * 플레이어를 애니메이션 한다.
         * 서버와 통신하여 각 플레이어들의 정보를 받는다.
         * 서버에서 게임 종료 정보를 받으면 결과를 출력한다
    3. void processInput()
       - * String input: 입력값을 받아 저장한다.
         * ↓↑: 플레이어의 이동 방향을 설정한다.

**3-2. Low Level 서버 구상안**

**A. TCP 서버-클라이언트 통신을 위한 기본 함수**

* + **void err\_quit(const char\* msg)**
    1. const char\* msg : 문자열을 받아올 인자
    2. 현재 발생한 오류 메시지를 메시지 박스로 표시하고, 프로그램을 종료하는 함수
  + **void err\_display(const char\* msg)**
    1. const char\* msg: 문자열을 받아올 인자
    2. 통신 중 에러가 발생한 경우 해당 오류 메시지를 메시지 박스 대신 printf()로 출력하고, 프로그램을 종료하지는 않는다.
  + **void recvn(SOCKET \_s, const char\* \_chBuf, int \_ilen, int \_iFlag)**
    1. recv() 함수는 TCP 프로토콜의 특징 상 데이터 경계를 구분하지 않는다. 이 때문에 클라이언트가 서버로부터 자신이 보낸 len값의 크기만큼 데이터를 받기 전까지 대기시간이 생기는데, recvn() 사용자 정의 함수를 통해 recv() 함수를 지속적으로 호출하여 이를 보완할 수 있다.

**B. 게임 플레이 Scene**

* + **DWORD WINAPI MainGameThread(LPVOID arg)**
    1. void playerUpdate()
       - * 각 클라이언트가 보내는 상태 정보와 위치 정보를 이용하여 플레이어의 위치 정보를 갱신한다.
    2. void notifyCollisions()
       - * 플레이어와 공 사이의 충돌처리:

두 공은 플레이어와 닿을 때마다 가속

* + 1. void BallUpdate()
       - * 주고받는 공의 방향과 속도를 계산
  + **DWORD WINAPI EndGameThread(LPVOID arg)**
    1. void init\_Game()
       - * 게임 화면 초기화

게임 결과 출력 후 일정시간 경과, 게임 초기화

* + 1. void End\_Game()
       - * 종료 조건을 만족했을 때 (ex. 플레이어 범위 밖, 콘솔창 밖)
  + **송신할 데이터 구조체[sc\_packet\_mainGame]**
    1. Point2D p2dPosition: 공의 위치 정보
    2. int iSpeed: 공의 속도
    3. float2 fVector: 공의 방향벡터
  + **송신할 데이터 구조체[sc\_packet\_EndGame]** 
    1. bool bResult: 게임 결과에 대한 정보

**4. 개발 환경 및 세부 기획**

**4-1. 개발 환경**

* OS: Windows 10
* IDE (통합 개발 환경): Visual Studio 2019
* 개발 언어: C / C++
* 통신 프로토콜: TCP (TCP/IP)
* 라이브러리: WinAPI, WinSock 2.2
* 버전 관리 시스템: GitHub

**4-2. 역할 분담**

|  |  |
| --- | --- |
| 채영문 | [클라이언트] 공통함수 구현 – update(), render(), sendData(), recvData(),  [서버] MainGame Class - notifyCollisions(), BallUpdate()  [서버] GameStart() |
| 황지원 | [클라이언트] EndGameManager – End\_Game()  [클라이언트] EndGameManager - init\_Game()  [서버] 공통함수 구현 - init\_Client\_Socket(), recvn(), err\_quit(), err\_display()  [서버] MainGame Class – playerUpdate() |

|  |  |
| --- | --- |
| 주차 별 목표 | |
| 1주차 | 추진계획서 수정, 클라이언트 프레임워크 설계 |
| 2주차 | 클라이언트 데이터 구조체, 클라이언트-서버의 공통 함수 구현, |
| 3주차 | 서버 데이터 구조체 정의 및 데이터 송수신 기능 구현, 클라이언트 키 입력 정보 송수신 기능 처리, 클라이언트-서버 데이터 전송 구현, |
| 4주차 | 업데이트 된 데이터 송수신 구현, 각 플레이어의 상태에 따른 데이터 송수신 및 처리 구현, 게임의 결과 송수신 구현 |
| 5주차 | 종료 후 게임 초기화 구현, 종합적인 상태 검토, 코드 병합 및 버그 수정 |
| 6주차 | 코드 병합 및 버그 수정, 최종 디버깅, 최종 보고서 작성 |

**4-3. 개발 일정**

**11월**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 일 | 월 | | 화 | 수 | 목 | | 금 | 토 |
| 1주차 | 10/31 | 1 | | 2 | 3 | 4 | | 5 | 6 |
| 채영문 | 추진계획서  마무리 | 추진계획서  1차 검토 및  제출 | | 추진계획서  1차 검사 |  | 추진계획서 재작성 및 클라이언트-서버 설계 | | | |
| 황지원 |
| 2주차 | 7 | 8 | | 9 | 10 | 11 | | 12 | 13 |
| 채영문 |  | 추진계획서  2차 검토 및 제출 | |  | 클라이언트 공통함수 구현 | | | 클라이언트  데이터 구조체 정의 |  |
| 황지원 | 1주차 정리  수정사항 기록 |  |  | | 통신을 위한 공통함수 구현  연결 확인 | | 서버 데이터  구조체 정의 |
| 3주차 | 14 | 15 | | 16 | 17 | 18 | | 19 | 20 |
| 채영문 |  | 추진계획서  3차 검토 및 제출 |  | | 클라이언트 정보 송수신  기능 구현 | | |  | 중간점검 |
| 황지원 | 2주차 정리  수정사항 기록 |  | | 서버의 송수신 기능 구현 | [playerUpdate] 플레이어 위치 송수신 | |  |
| 4주차 | 21 | 22 | | 23 | 24 | 25 | | 26 | 27 |
| 채영문 | [notifyCollisions] 공과 플레이어 사이의 충돌처리 | [notifyCollisions] 충돌처리 후 공의 가속 처리 | | | [BallUpdate]  공의 범위 이탈 처리 | | | [GameStart]  클라이언트에게 게임 시작을 알리는 기능 |  |
| 황지원 | 3주차 정리  수정사항 기록 | 업데이트 된 상태에 따른  데이터 송수신 및 처리 구현 | | | |  | | [playerUpdate] 게임 결과를  각 클라이언트에게 전송하는 기능 | |
| 5주차 | 28 | 29 | | 30 |  |  | |  |  |
| 채영문 | 종합적인 상태 검토 | | |  |  |  | |  |  |
| 황지원 | 4주차 정리  수정사항 기록 | [init\_Game] 게임 종료 후  게임 초기화 기능 | | |

**12월**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 5주차 |  |  | 11/30 | 1 | 2 | 3 | 4 |
| 채영문 |  | |  | 코드 병합 | 버그 수정 및 최종 디버깅 | | 최종 보고서  작성 |
| 황지원 |
| 6주차 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 채영문 |  | 텀 프로젝트 코드 및  최종 계획서 제출 | |  |  |  |  |
| 황지원 | 5,6주차 정리  최종 보고서 준비 |