**네트워크 게임 프로그래밍**

**텀 프로젝트 추진계획서**

**[Mini Pac-Man]**

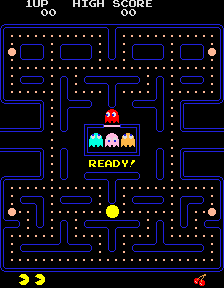
게임공학과 2017180045 홍찬우

게임공학과 2019182026 이수민

게임공학과 2019182030 이화수

**목차**

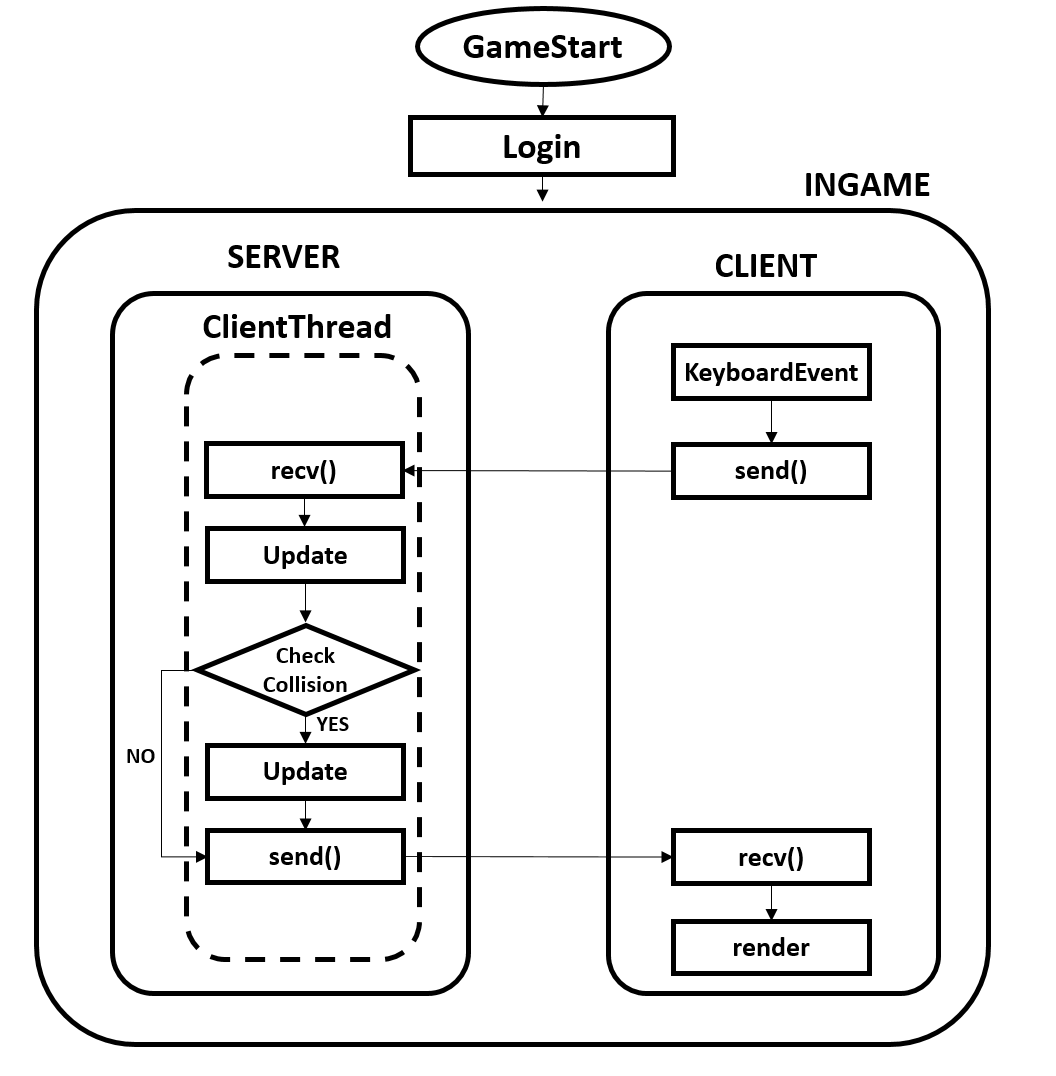
1. 어플리케이션 기획 3
   1. 게임 설명
   2. 플레이 방법
   3. 조작 방법
2. High-Level Design 4
   1. Game Flow-Chart
   2. Login State 5
3. Low-Level Design 6
   1. Define
   2. Object struct / class 7
   3. Struct 8
   4. Function 9
   5. 동기화
4. 개발 환경 10
5. 역할 분담
6. 개발 일정 11
7. **애플리케이션 기획**
   1. **게임 설명**
      1. 게임 제목 : Mini Pac-Man
      2. 게임 소개

  
( 사진 출처 : https://namu.wiki/w/%ED%8C%A9%EB%A7%A8(1980)?from=PAC-MAN)

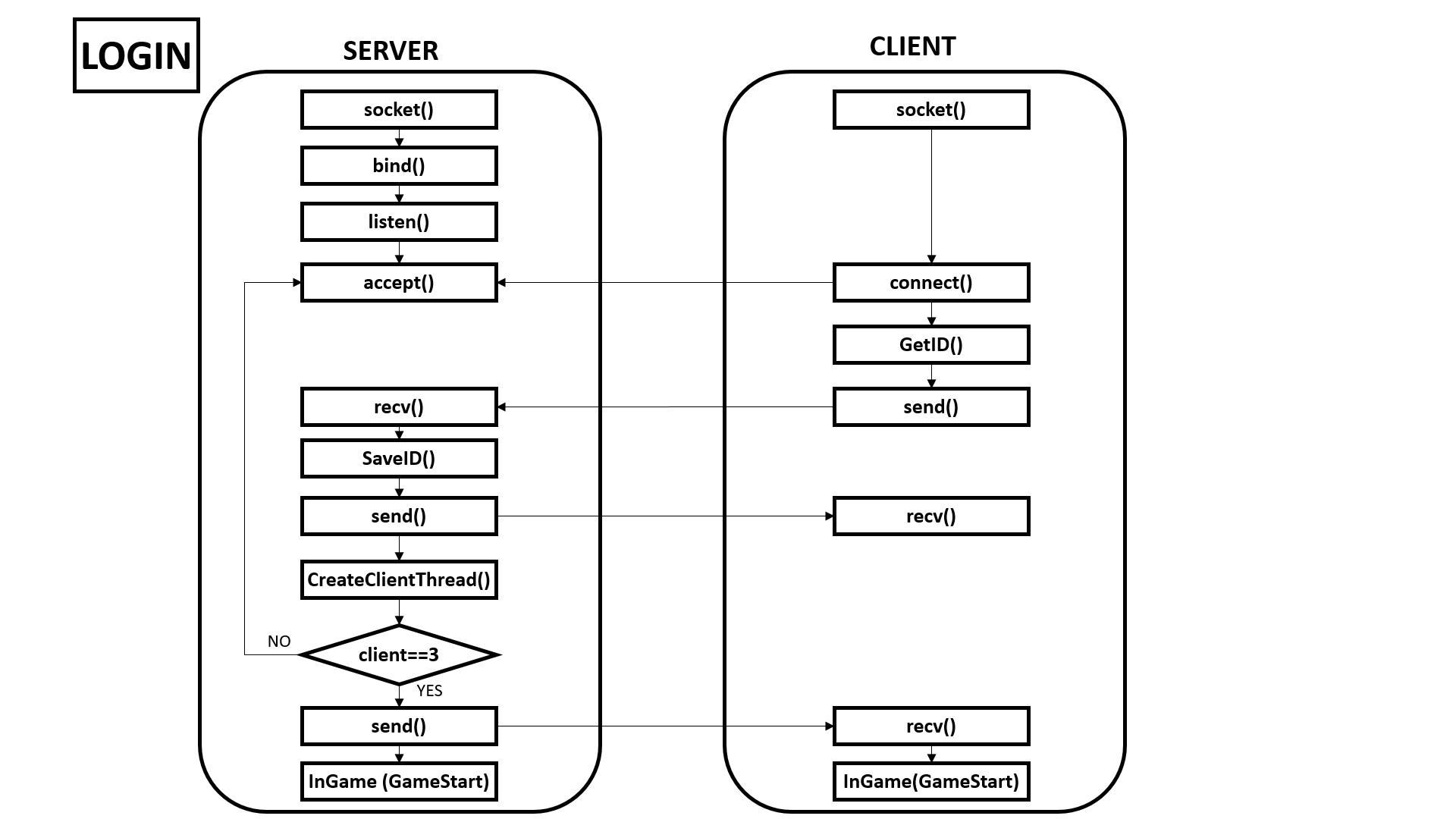
* + - * 각 플레이어는 주어진 체력 내에서 맵 곳곳에 위치한 Enemy를 잡으며 살아남아야 한다.
    1. 플레이 방식
       - 3인 멀티 플레이 게임
  1. **플레이 방법**
     1. 게임 접속 (서버와의 연결, 로그인)
     2. 키 조작을 통해 Enemy를 잡는다.
     3. HP가 모두 소모되는 플레이어는 GAMEOVER, 더 많은 Enemy를 잡은 플레이어가 승리
  2. **조작 방법**

|  |  |
| --- | --- |
| 키 입력 | 결과 |
| W | 위쪽으로 이동 |
| A | 왼쪽으로 이동 |
| S | 아래쪽으로 이동 |
| D | 오른쪽으로 이동 |

1. **High-Level Design**
   1. **Game Flow-Chart**

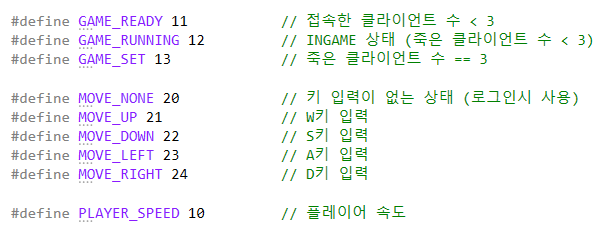


* + - * 클라이언트에서 키보드 입력이 발생하면 이를 서버에게 전송하여 해당 클라이언트의 위치 정보를 업데이트해주고, 주변 객체와의 충돌 체크를 진행하고 위치 정보 업데이트 여부를 결정한 후, 모든 클라이언트의 위치정보를 각 클라이언트에게 전송한다. 각 클라이언트는 해당 정보를 기반으로 화면에 모든 클라이언트의 플레이어를 렌더링한다.
      * ~~실시간 상호작용과 정확한 플레이어의 위치정보 갱신을 위해 deltaTime()을 이용한다.~~
  1. **Login State**



* + - * 각 클라이언트에서 ID를 입력 받아 서버에게 넘겨주면 일회용 로그인을 진행한다.
      * 3개의 클라이언트가 접속하여 로그인한 상태가 되면 서버는 게임 시작 플래그를 모든 클라이언트에게 전송하여 게임을 시작한다.

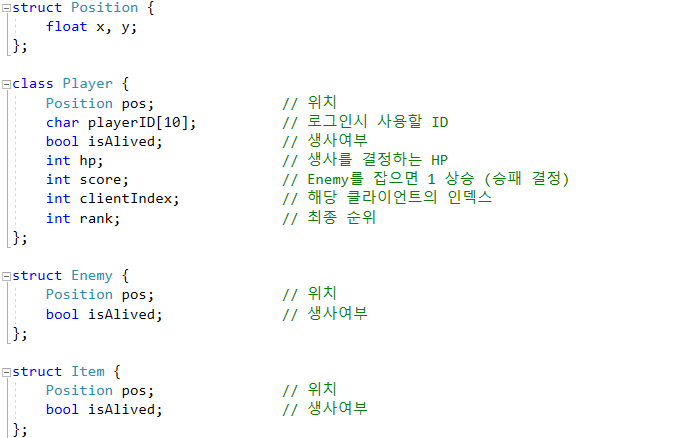
1. **Low-Level Design**
   1. **Define**
      1. INGAME



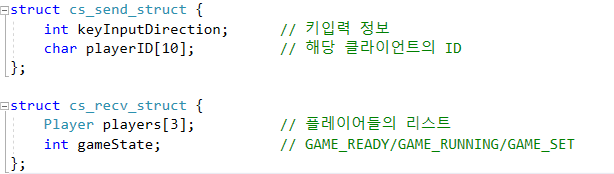
* + 1. 클라이언트



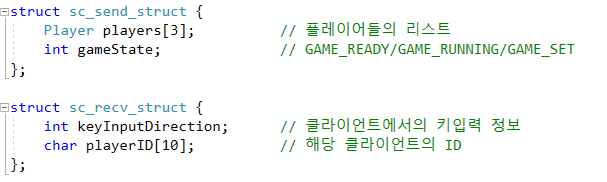
* 1. **Object Struct / Class**



* 1. **Struct**
     1. Client -> Server 송수신 담당 구조체



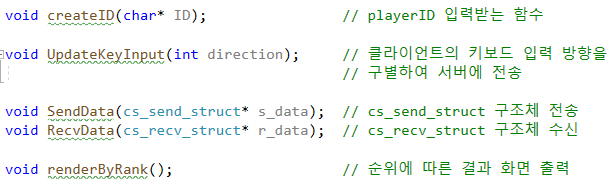
* + 1. Server -> Client 송수신 담당 구조체



* 1. **Function**
     1. Server



* + 1. Client



* 1. **동기화**
     + - 이벤트를 이용하여 충돌 체크 이후 스레드가 공유 자원인 Enemy와 Item에 접근하면 이벤트를 신호 상태로 만들고, 관련 처리 이후 수동 리셋으로 비신호상태로 만든다.

1. **개발 환경**

|  |  |
| --- | --- |
| 구분 | 내용 |
| 동작 OS | Windows 10 |
| IDE | Visual Studio 2019 |
| 사용 프로토콜 | TCP/IP |
| 사용 API | Windows API / WSA |
| 개발 언어 | C/C++ |

1. **역할 분담**

|  |  |  |
| --- | --- | --- |
| 구분 | 내용 | 담당 |
| 클라이언트 | 프레임워크 구축 | 이화수 |
| 오브젝트 구현 | 이화수 |
| 키보드 입력 | 이화수 |
| 데이터 송수신 작업 | 홍찬우 |
| 로그인 / 종료 화면 구현 | 홍찬우 |
| 서버 | 프레임워크 구축 | 이수민 |
| 데이터 송수신 작업 | 이수민, 이화수 |
| 로그인 | 이수민 |
| 충돌처리 | 홍찬우, 이화수 |
| 순위결정 | 홍찬우 |
| ~~TimeThread~~ | ~~이수민~~ |
| 동기화 (이벤트 사용) | 이수민 |

1. **개발 일정**

|  |  |  |  |
| --- | --- | --- | --- |
| 일자 | 홍찬우 | 이수민 | 이화수 |
| 10/26 | 기획서 작성 | | |
| 10/27 |
| 10/28 |
| 10/29 |
| 10/30 |
| 10/31 |
| 11/01 |
| 11/02 | 기획서 제출 및 검수 | | |
| 11/03 | 클라이언트 sendData() |  |  |
| 11/04 |  |  |  |
| 11/05 | 클라이언트 sendData() | 서버 구조체 | 클라이언트 제작 |
| 11/06 | 클라이언트 recvData() |  |  |
| 11/07 | CheckPlayerByWallCollision() |  |  |
| 11/08 | 진행사항 보고, 보완 및 일정 조율 | | |
| 11/09 |  | 서버 프레임워크 | 클라이언트  오브젝트, 구조체 구현 |
| 11/10 |  |  | 클라이언트  UpdateKeyInput(); |
| 11/11 |  | sendData() |  |
| 11/12 |  | recvData() | 클라이언트  Enemy, Item 생성 |
| 11/13 | CheckPlayerByPlayerCollision() |  |  |
| 11/14 | CheckPlayerByEnemyCollision() |  |  |
| 11/15 | 진행사항 보고, 보완 및 일정 조율 | | |
| 11/16 | CheckPlayerByItemCollision() | ClientThread | MovePlayer() |
| 11/17 | MakeRank() |  | CheckPlayerByWallCollision(); |
| 11/18 |  | ClientThread | CheckPlayerByPlayerCollision() |
| 11/19 |  | ~~TimeThread~~ 동기화 작업 |  |
| 11/20 | isGameOver() | ~~스레드 점검~~ 동기화작업 |  |
| 11/21 |  |  |  |
| 11/22 | 진행사항 보고, 보완 및 일정 조율 | | |
| 11/23 |  |  |  |
| 11/24 | 게임 시작 화면 구현 | Login | CheckPlayerByItemCollision(); |
| 11/25 |  | isPlayerAlived() | CheckPlayerByEnemyCollision(); |
| 11/26 |  | sendGameStart() |  |
| 11/27 | 게임 시작 화면 구현 |  |  |
| 11/28 | 게임 종료 화면 구현 - RenderByRank() |  |  |
| 11/29 | 진행사항 보고, 보완 및 일정 조율 | | |
| 11/30 | 테스트 및 디버깅 | | |
| 12/1 | 테스트 및 디버깅 | | |
| 12/2 | 테스트 및 디버깅 | | |
| 12/3 | 최종 테스트 | | |
| 12/4 | 미비점 보완 | | |
| 12/5 | 미비점 보완 | | |
| 12/6 | 최종 검수 | | |
| 12/7 | 프로젝트 제출 | | |