**네트워크 프로그래밍 프로젝트**

**- Speed Edge 게임 -**

20122382 박준은

20122484 채수민

20122338 김도현

**<목 차>**

**1. 게임 결과물**

**A. 게임 시나리오**

**(1) 시나리오와 터미널 화면**

**(2) 시나리오 protocol sequence**

**B. 메시지 구조 설계**

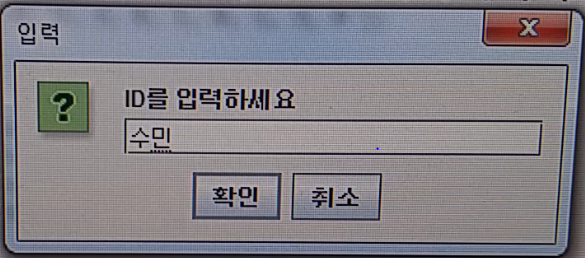
**2. 구현 내용 기술**

**3. 항목별 설계 대비 구현 여부**

1. **설계 결과물**
   1. **게임 시나리오**
2. **게임 시나리오와 게임 화면**
   * 1. ***첫 게임은 세 명의 클라이언트가 목표점수를 달성해서 한 게임이 종료***
     2. ***게임 대기 화면에서 한 명의 클라이언트가 나간다***
     3. ***두 번째 게임은 두 명의 클라이언트가 더 들어와서 게임이 시작하고 남은 게임의 수가 0이되서 게임 종료***

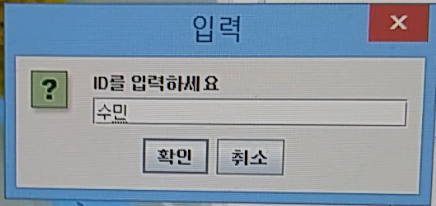
***i. 첫 게임은 세 명의 클라이언트가 목표점수를 달성해서 한 게임이 종료***

1. 클라이언트 1이 ‘수민’ 으로 접속한다.

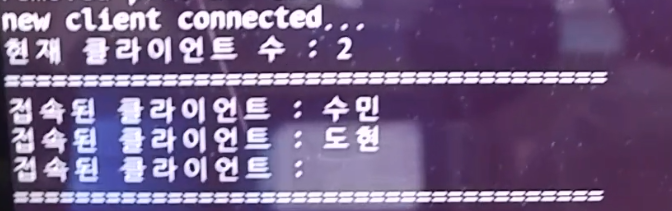




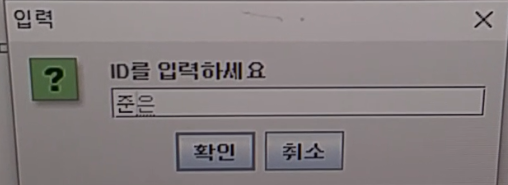
2. 클라이언트 2가 ‘수민’ 으로 접속한다. 중복처리를 해서 접속이 불가능. 다시 ‘도현’ 으로 접속

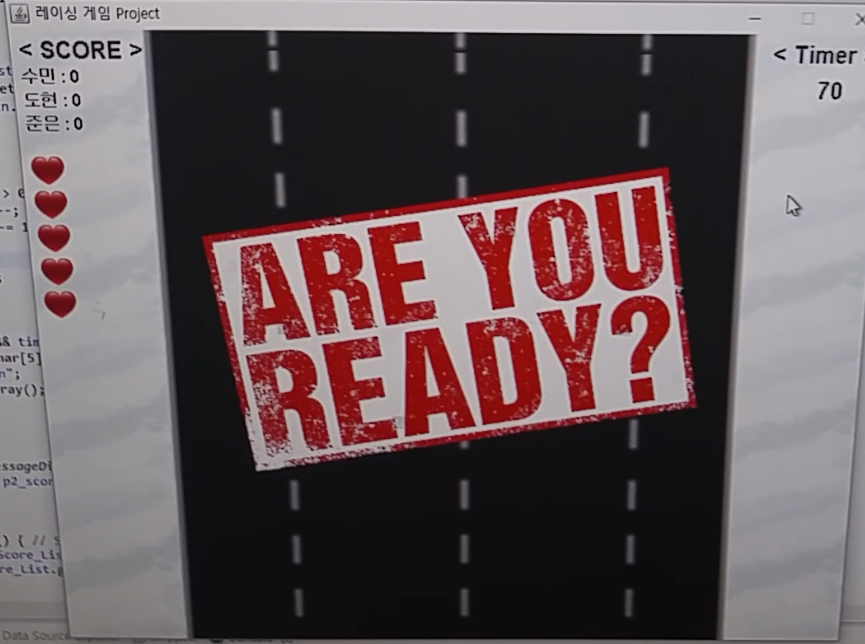




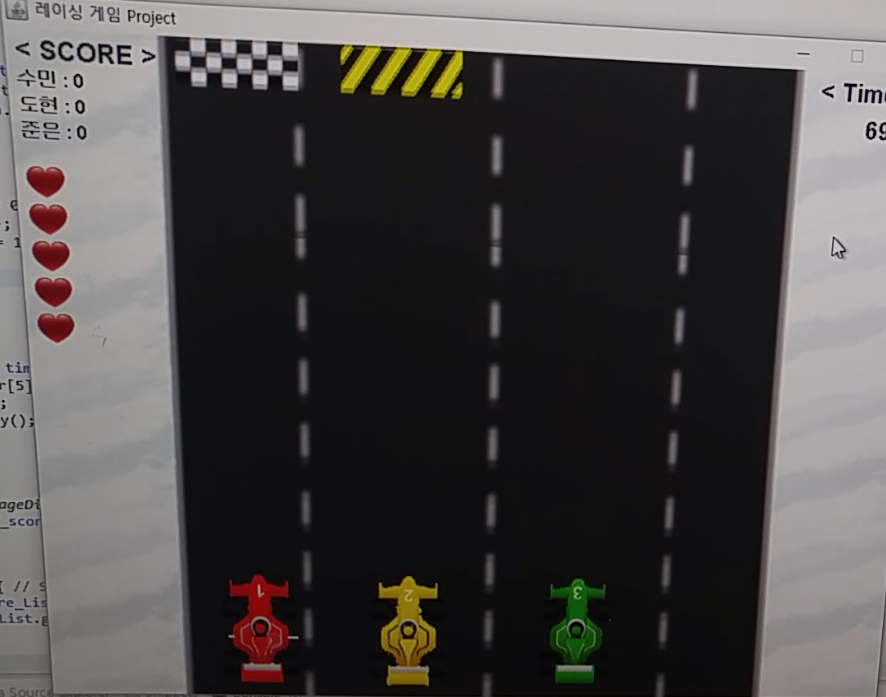


3. 클라이언트 3이 ‘준은’ 으로 접속한다.

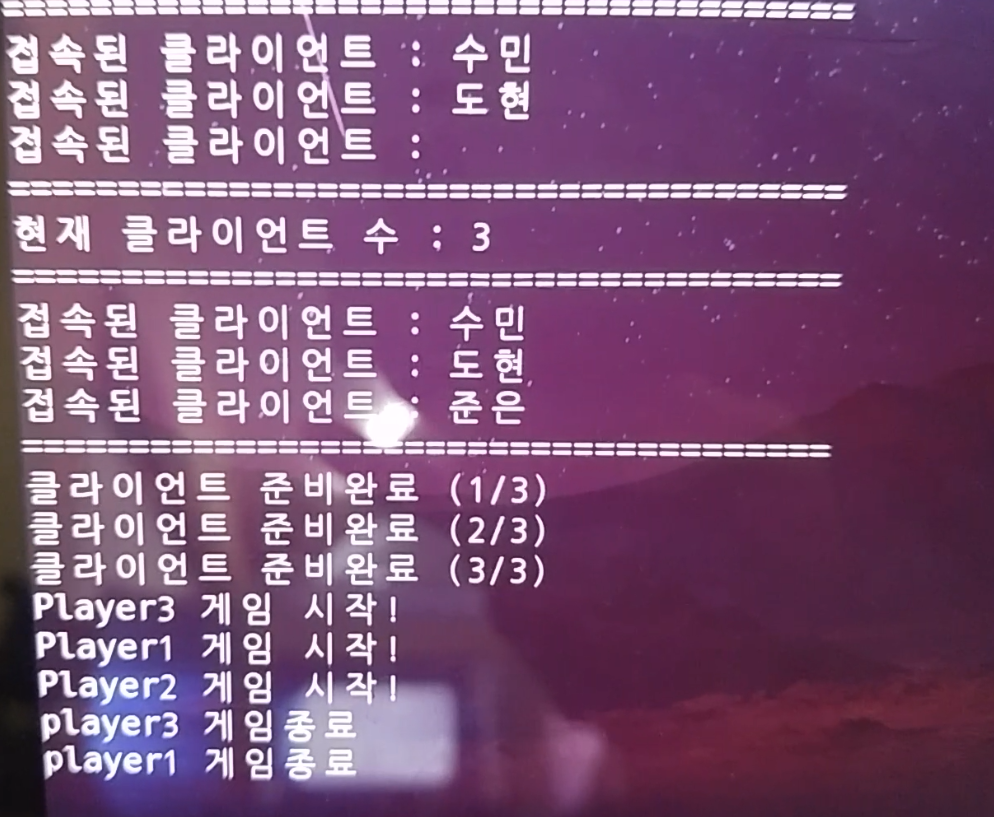




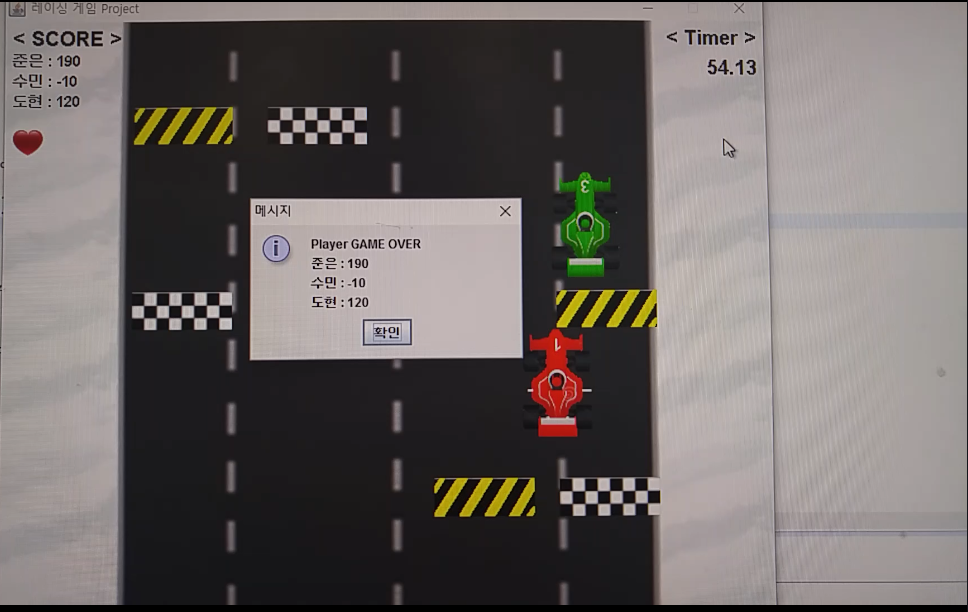
5. 모두 ‘준비’ 상태에 들어가면 게임을 시작한다.



6. 서버에서 접속한 내용을 확인한다.

. 

6. 게임종료 후 결과를 출력하고 재시작을 기다린다.





**B. 메시지 구조 설계**

1. 메시지 구조(서버 -> 클라이언트)
2. **DUP**
   * 닉네임 중복검사 결과 메시지
   * 중복일 경우 0, 중복이 아닐 경우 1을 전송
   * Ex)DUP 1 or DUP 0

|  |  |  |
| --- | --- | --- |
| DUP | | | 1 or 0 |

1. **NUM**
   * 내가 몇번째 플레이어인지 알려준다.
   * 첫번째 플레이어면 1, 두번째는 2, 세번재면 3
   * Ex) NUM|1 or NUM|2

|  |  |  |
| --- | --- | --- |
| NUM | | | Player number |

1. **NAME** 
   * 현재까지 접속한 플레이어의 아이디를 보내준다.
   * Player1부터 보낸다.
   * Ex) NAME|player1|player2|player3|

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NAME | | | Player1  name | | | Player2  name |  | Player3  name | | |

1. **START**
   * 현재 게임 대기자 상태 메시지(ready상태이면 1(bool형))
   * Ex) START|\0

|  |  |  |
| --- | --- | --- |
| START | | | \0 |

1. **LOC**
   * 플레이어의 좌표들과 점수들을 한번에 보낸다.
   * Ex) LOC|p1x|p1y|p2x|p2y|p3x|p3y|p1점수|p2점수|p3점수|

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LOC | P1x | | | P1y | | | P2x | | | P2y | | | P3x | | | P3y | | | P1점수 | | | P2 점수 | | | P3 점수 | | |

1. **END**
   * 클라이언트에게 끝을 알려준다.
   * Ex) END|

|  |  |
| --- | --- |
| END | | |

1. 메시지 구조(클라이언트 -> 서버)
2. **NAME**

* 닉네임 전송 메시지
* Ex) NAME|player1

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | | | 닉네임 | \n |

1. **QUIT**

* 게임 종료 메세지
* Ex) QUIT|아이디

|  |  |  |  |
| --- | --- | --- | --- |
| QUIT | | | 아이디 | \n |

1. **READY**

* 게임 레디 메세지
* Ex) READY|

|  |  |  |  |
| --- | --- | --- | --- |
| READY | | | 아이디 | \n |

1. **LOC**

* 플레이어의 좌표와 점수
* Ex) LOC|x좌표|y좌표|점수

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| LOC | | | X좌표 | | | Y좌표 | | | 점수 | \n |

1. **QUIT**

* 게임 종료 메시지
* Ex)QUIT

|  |  |
| --- | --- |
| QUIT | \n |

1. **END**

* 게임 오버 메세지
* Ex) END

|  |  |
| --- | --- |
| END | | |

1. **시나리오 protocol sequence**
2. 정상적인 게임 플레이 sequence



1. 체력 0으로 인한 게임 종료 sequence



**2. 구현 내용 기술**

**<server>**

멀티 프로세스를 통해 다중 접속을 구현했다. 서버에서 프로세스간 통신은 공유메모리 기법을 사용했다. 공유메모리를 통해 메모리에 공간을 만들고 그 공간에서 클라이언트의 이름과 점수 그리고 각 자동차의 좌표를 접근한다. 공유메모리를 사용할 경우 모든 프로세스가 종료되어도 한동안 그 값이 남아있으므로, 처음 서버가 시작되면 공유메모리를 초기화 해준다. 각 클라이언트가 자신의 좌표값과 점수를 서버로 전송하면, 서버는 그 정보를 구분해 공유메모리에 적고, 서버는 또 각 클라이언트에게 그 좌표와 점수를 전송한다. 정보를 구분하기위해 strtok() 함수를 사용해 클라이언트가 보낸 정보를 자른다.

* void error\_handling(char \*message);

에러가 발생했을 시 에러 메세지를 띄워주고 프로그램을 종료한다.

* void read\_childproc(int sig);

자식프로세스가 종료 시 어떤 프로세스가 종료되었는지 확인한다.

* void childProcess();

자식프로세스가 시작되는 함수이다.

* void ReadWrite();

게임 시작 후 서버와 클라이언트가 통신하는 함수이다.

* void idCheck();

클라이언트가 전송한 아이디를 중복체크하는 함수이다. 중복일경우 다시 전송을 받는다.

* void readyCheck();

클라이언트가 ready 버튼 혹은 quit 버튼을 눌렀을 때 이벤트를 처리하는 함수이다.

* void initName();

공유메모리의 이름과 레디수, 클라이언트 수를 초기화하는 함수이다.

* void initInfo();

공유메모리의 점수와 클라이언트들의 좌표를 초기화하는 함수이다.

* void shareName();

이름을 저장하는 구조체의 공유메모리를 생성하는 함수이다.

* void shareCnum();

클라이언트의 수와 ready가 눌린 수를 저장하는 공유메모리를 생성한다.

* void shareLoc();

좌표와 점수를 저장하는 공유메모리를 생성하는 함수이다.

* void start();

클라이언트가 레디를 해서 시작하기까지 기다려 시작했을 때 start하는 함수이다.

* void Log\_clnt(char\* to, char\* msg);

클라이언트 부분의 로그를 써주는 함수다. 함수 인자인 to는 클라이언트 이름, msg는 기록할 메시지다.

* void Log\_serv(char \*msg);

서버 부분의 로그를 써주는 함수다. 함수 인자는 msg는 서버가 기록할 메시지다.

**< client >**

클라이언트 게임을 구현하기 위해 언어는 자바를 사용하였으며, 자바 GUI를 이용하여 게임을 구현하였다. 게임을 시작하면 아이디를 입력 받아 서버에 접속한다. 서버에 접속하면 서버에서 현재 접속된 아이디들을 받아오고, 준비 버튼을 누르면 게임 준비 상태로 진입한다. 모든 플레이어들이 준비 상태가 되면 서버에서 시작을 알려주고 게임이 시작된다. 게임이 매끄럽게 진행되기 위해서 더블 버퍼링을 사용하였으며, 서버에서 각 플레이어들의 좌표와 스코어를 보내주고 이를 그려주며 게임이 진행된다. 게임이 끝나면 플레이어간의 점수를 통해 승자를 보여주며, 서버와의 접속을 끊기 위해서, Quit버튼을 통해서만 종료할 수 있게 해주었다.

● public Game()

- Game 클래스의 생성자이다.

● public void init()

- 게임 실행에 있어 필요한 변수들을 초기화 시켜준다.

● public void start(), public void run()

- 프로그램 실행 시, 시작되는 스레드이며, 게임의 전체적인 흐름이 진행되는 스레드이다.

● public void QuitProcess()

- 종료버튼이 눌러졌을 때 종료함을 서버에 보내준다.

● public void ReadyProcess()

- 레디 버튼이 눌러졌을 때 레디를 서버에 보내주고, 게임을 준비상태로 바꿔준다.

● public void CoinProcess()

- 터널이 내려올 좌표를 정해주고 생성된 터널의 배열에 저장해준다. 그리고 만약 플레이어가 터널을 통과할 경우, 점수를 올려준다.

● public void CornProcess()

- 방지턱이 내려올 좌표를 정해주고 생성된 방지턱의 배열에 저장해준다. 그리고 만약 플레이어가 방지턱에 부딪힐 경우, 점수를 내려준다. 그리고 체력이 0이 되었을 때 게임을 종료시켜준다.

● public void ScoreProcess()

- 플레이어가 통과한 터널에 +10 이펙트를 보여준다.

● public void Crash()

- 플레이어와 방지턱 또는 터널의 이미지가 부딪혔음을 계산하여 참/ 거짓을 판별해준다.

● public void upadate(Graphics g)

- update 함수 안에 쓰인 Draw\_background() ,Draw\_Timer(time), Draw\_Char(), Draw\_Coin()

, Draw\_Corn(), Draw\_CoinScore(), Draw\_StatusText() 을 통해 buffImage 안에 창에서 보여줄 이미지를 그리고, 이 buffImage를 창에 그려준다.

● public void KeyPressed(KeyEvent e), public void KeyReleased(KeyEvent e),

public void KeyProcess()

- 방향키를 눌렀을 때 차의 좌표를 이동해주기 위하여 override하여 사용하였다.

● public void Sound

- 터널을 통과 했을 때 효과음을 넣어주는 함수이다.

● public class LocWrite implements Runnable

- 위치를 서버로 보내주는 스레드이다.

● public class LocTest implements Runnable

- 서버로부터 읽어 들여 각 플레이어들의 좌표와 점수를 읽어들여 주는 스레드이다.

● public class Waiting implements Runnable

- 준비 후에 참여한 플레이어들의 ID를 받고, 스타트를 받아 게임을 시작해주는 스레드이다.

● class Coin

- 터널의 좌표와 터널이 부딪혔는지에 대한 변수를 정의하고 있는 클래스이다.

● class Corn

- 방지턱의 좌표와 방지턱에 부딪혔는지에 대한 변수를 정의하고 있는 클래스이다.

● class Score

- 점수를 획득 했을 때, 게임화면에 스코어 획득을 보여줄 좌표와 그 시간을 카운트 해줄 변수를 정의하고 있는 클래스이다.

● class CountDown

- 스레드를 통하여 시간 초를 카운트 다운 해주는 클래스이다.

**3. 항목별 설계 대비 구현 여부(표로 정리/구현 실패 항목에 대하여 원인 기술)**

|  |  |
| --- | --- |
| 항목별 설계 | 구현 여부 |
| 닉네임 중복검사 | O |
| 다중접속 구현 | O |
| 공유메모리 구현 | O |
| 장애물과 점수터널 구현 | O |
| 효과음 및 이펙트 구현 | O |
| 실시간 점수 바뀜 | O |
| 실시간 자동차 이동 | O |
| 체력 구현 | O |
| 타이머 구현 | O |
| 준비 및 시작 구현 | O |
| Quit 버튼 구현 | O |
| 타이머에 따른 속도차이 구현 | O |
| 경기 종료 시 순위발표 | O |
| 구현환경 리눅스 | O |
| GUI 게임 구현 | O |
| 로그파일 | O |
| 경기 종료시 재시작 가능 | O |

당초 계획에서 게임 재시작과, 체력 그리고 그 외 여러 GUI들을 추가함으로써 목표한 프로그램보다 게임의 질을 더 높게 만들 수 있었습니다.