TERM PROJECT

Air Hockey game 만들기

컴퓨터학부 2022111243 김두현

1. 주제소개

Linux 환경에서 사용할 수 있는 air hockey game 이다. 두 명의 플레이어가 각자의 스틱으로 공을 쳐내 상대방의 골대에 넣는 걸 목표로 하는 게임이다. 3 점을 먼저 득점하면 승리한다.

2. 기능 및 설계

2-1 사용한 라이브러리, 함수 및 코드 설명

2-1-1. 사용한 라이브러리

<stdio.h>, <curses.h>, <signal.h>, <unistd.h>, <string.h>, <sys/time.h>, <stdlib.h>, <sys/wait.h>

2-1-2. 새로 정의한 함수

void draw()

전반적인 공과 스틱의 이동을 담당한다. 이 게임은 공과 스틱의 현재 위치를 표시하고, 다음 위치를 일정 간격 이후에 스크린에 다시 표시하는 과정을 계속 반복하는 방식으로 진행된다. 공이 벽이나 스틱에 맞는 경우, 게임스코어 등 전반적인 출력과 진행 담당.

void draw_walls()

게임판의 경계와 골대를 그리는 함수이다.

void init()

플레이어 스틱의 위치, 공의 위치, 공의 방향 등 게임이 시작할 때 기본적인 옵션들을 설정해 주는 함수이다.

int set_ticker(int n_msecs)

일정한 시간 간격 동안 시그널을 생성하고, 게임의 상황을 업데이트해 주는 함수이다.

void ignore_signal()

Ctrl + c, Ctrl + / 등의 signal 을 무시하기 위해 활용한 함수이다.

void draw_goal_message()

골이 들어가고 나면 화면 중앙에 "Goal In" message 를 출력하는 함수이다.

void countdown_and_reset()

골이 들어가고 난 후 다음 판이 시작하기 전까지의 카운트다운을 해 주는 함수이다. 미리 설정된 countdown 값을 이용하며, 다음 판 시작까지 몇 초가 남았는지 화면에 출력된다.

void game_over_and_exit()

누군가 END_SCORE 에 도달했을 때 게임을 종료하는 함수이다. Child process 에서 수행되며, 마지막 화면 draw 를 수행하고 child process kill 을 담당한다.

2-1-3. 대략적인 코드 진행과 함수 설명

전처리기, 구조체와 전역변수

```c
#include <stdio.h>
#include <curses.h>
#include <signal.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include <stdlib.h>
#include <sys/wait.h>

//공 모양 설정
#define MESSAGE "o"
#define BLANK "     "

//윈도우 크기, 골대크기 설정
#define TOP_ROW 2
#define BOT_ROW LINES-2
#define LEFT_EDGE 3
#define RIGHT_EDGE COLS-3
#define UpperPost LINES / 2 + 2
#define LowerPost LINES / 2 - 2

//END_SCORE를 먼저 내는 사람이 이김.
#define END_SCORE 3

//tick 설정
#define TICKS_PER_SEC 50
```

```c
//player struct
typedef struct
{
    int x;
    int y;
    int score;
} Player;

Player p1, p2;

// ball features
int row;
int col;
int x_dir;
int y_dir;
int x_delay;
int y_delay;

//global var
int goal_in = 0;
int countdown = 3;
int winner=0;
pid_t pid;
```

Player struct, 공과 관련된 변수, 기타 게임 진행과 관련된 전역변수다.

Main

```
int main()
{
    int c;

    // hide the cursor
    curs_set(0);

    initscr();
    crmode();
    noecho();
    clear();
    ignore_signal();

    init();

    // 공의 초기 위치를 그림
    mvprintw(row, col, MESSAGE);

    // 시그널 핸들러 설정
    signal(SIGALRM, draw);
    set_ticker(x_delay);
    draw();

  //fork
  pid=fork();
```

```
  if(pid==0){
  //chile process
    if(winner){
      draw();
      game_over_and_exit();
    }
  }
```

Child process : 게임 종료조건에서만 사용된다.

```
else if(pid){
//parent process
  while (p1.score<END_SCORE && p2.score <END_SCORE)
  {
    //게임의 전반적인 진행을 담당하는 부분 : END_SCORE 도달할 때까지 반복
    draw();
      usleep(1000000);

      //하키스틱의 움직임을 조절하는 부분
      //Player 1 : w, a, s, d
      //Player 2 : i, j, k, l
      int ch=getch();
      switch (ch)
```

Parent process : 게임의 전반적인 진행을 담당한다. while 문에서 getch()로 계속해서 키보드 입력을 받고, 스틱과 공의 속도 등을 조절한다. switch 문의 사진 첨부는 생략한다.

```
        set_ticker(x_delay);
    }
    //child processs 수행을 기다림.
    waitpid(pid, NULL, 0);
    endwin();

    //윈도우가 닫히고 어떤 플레이어가 이겼는지 출력해 줌.
    printf("Plyer %d win !\n", winner);
  }
  else{
    perror("fork Fail!\n");
    return -1;
  }
    return 0;
```

```
int set_ticker(int n_msecs)
{
    struct itimerval new_timeset;
    long n_sec, n_usecs;

    n_sec = n_msecs / 1000;
    n_usecs = (n_msecs % 1000) * 1000L;

    new_timeset.it_interval.tv_sec = n_sec;
    new_timeset.it_interval.tv_usec = n_usecs;
    new_timeset.it_value.tv_sec = n_sec;
    new_timeset.it_value.tv_usec = n_usecs;

    return setitimer(ITIMER_REAL, &new_timeset, NULL);
}
```

main 함수는 while 문을 돌면서 키보드 입력을 받고, set_ticker 함수를 호출하며 화면에 계속해서 게임상황을 반영한다.

Draw() 함수

전반적인 공과 스틱의 움직임을 게임판에 반영한다.

공이 벽에 맞은 경우, 스틱에 맞은 경우 공의 움직임을 처리하는 부분은 첨부하지 않았다.

```
void draw()
{
    signal(SIGALRM, draw); // 시그널 핸들러 재설정

    // 현재 위치에 있는 공 및 스틱을 지움
    mvprintw(row, col, BLANK);

    mvprintw(p1.y, p1.x, "|");
    mvprintw(p2.y, p2.x, "|");


    // 공의 새로운 위치 계산
    col += x_dir;
    row += y_dir;

    // 새로운 위치에 벽과 공 및 스틱을 그림
    draw_walls();
    mvprintw(row, col, MESSAGE);

    // 점수 출력
    mvprintw(0, COLS / 2 - 30, "Player 1: %d", p1.score);
    mvprintw(0, COLS / 2 + 10, "Player 2: %d", p2.score);

    refresh();
    curs_set(0);

  if(winner){
    mvprintw(LINES/2, COLS/2 - 5, "Game Over");
    mvprintw(LINES/2+1, COLS/2 - 10, "Press any key to exit");
    return;
  }
```

```
    //공이 들어간 경우
    if(LowerPost <= new_row && new_row <= UpperPost && new_col==RIGHT_EDGE-1){
      p1.score+=1;
      goal_in = 1;
      if(p1.score ==END_SCORE){
        winner=1;
        return;
      }
        draw_goal_message();
        countdown_and_reset();
        }
    else if(LowerPost <= new_row && new_row <= UpperPost && new_col==LEFT_EDGE){
      p2.score+=1;
      goal_in = 2;
      if(p2.score ==END_SCORE){
        winner=2;
        return;
      }
        draw_goal_message();
        countdown_and_reset();
        }


}
```

Chile process 에서 실행되는 game_over_and_exit() 함수

game 종료 조건이 충족되면 종료 메시지를 출력하고 child process 를 종료해 주는 역할이다.

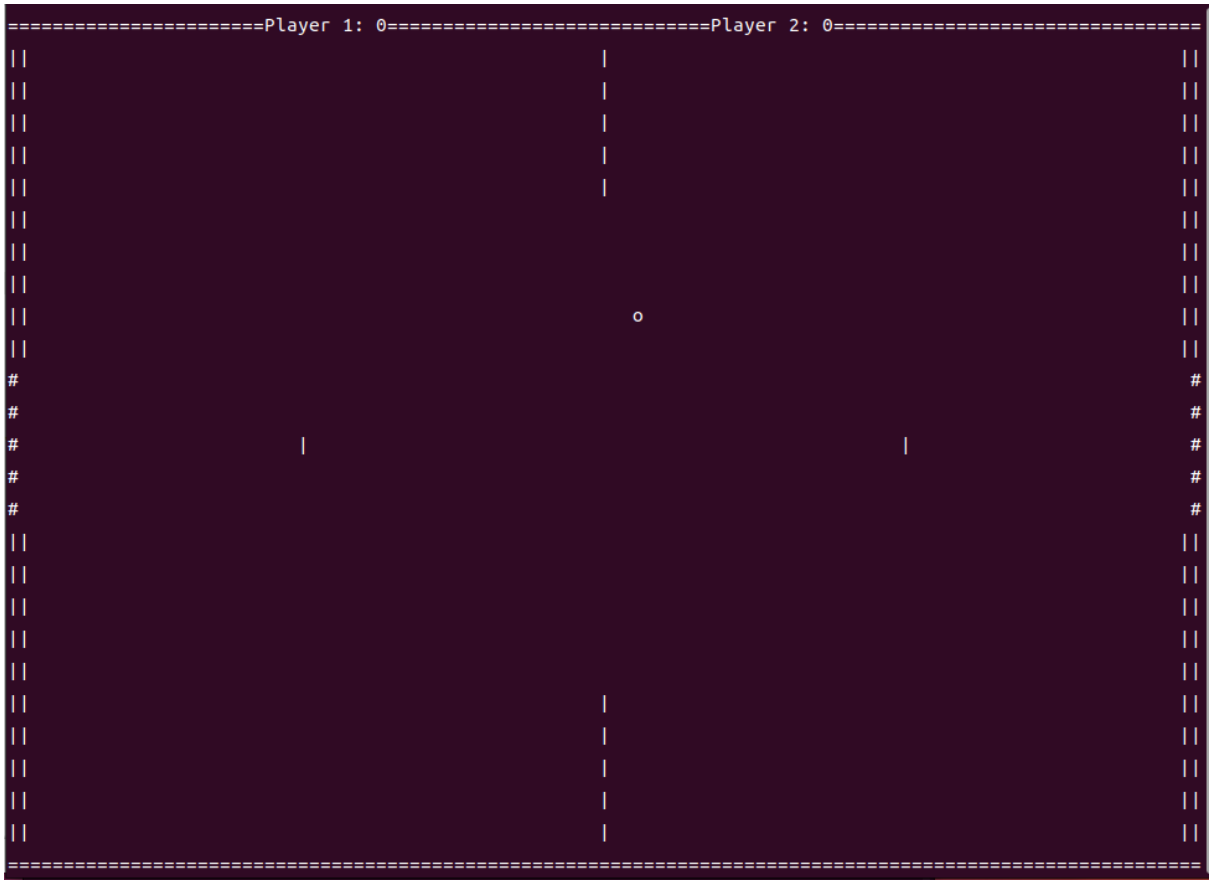
```
void game_over_and_exit() {
    if (pid > 0) {
        kill(pid, SIGTERM);  // Terminate the child process
    }

    refresh();
    sleep(5);
    exit(0);
}
```

2-2 주요 기능 및 수행화면

2-2-1. 게임판 설계와 공의 이동

게임만 경계는 <curses.h>라이브러리에 정의되어 있는 LINES, COLS 값을 이용했다. 상하좌우 각각 TOP_ROW, BOTTOM_ROW, LEFT_EDGE, RIGHT_EDGE 값으로 게임판 경계가 만들어져있고, '='과 '||' 문자로 벽을 표현했다. 중앙선은 '|'로 표현되어있고, 공을 튀기는 벽으로서의 기능은 없다.

공은 벽이나 플레이어의 스틱에 맞으면 이동방향의 반대방향으로 튕겨져나간다.
공의 속도는 'z' 키나 'm' 키 입력으로 조절할 수 있는데, 'z' 키는 공의 속도를 느리게 만들고, 'm' 키는 공의 속도를 빠르게 만든다.



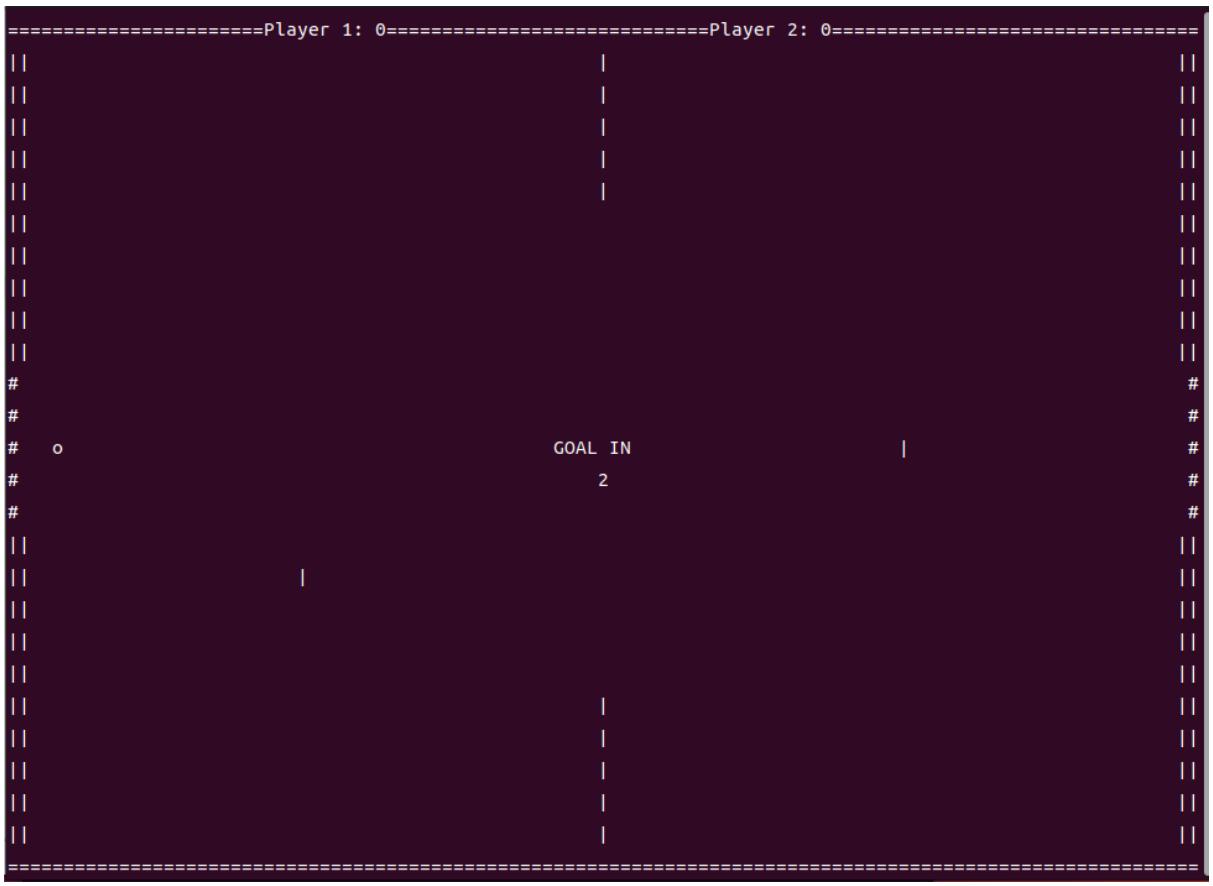
2-2-2. 플레이어 스틱의 조종과 득점

Player 1 : w(up), a(left), s(down), d(right)

Player 2 : i(up), j(left), k(down), l(right)

각 플레이어는 각자의 스틱을 조종해서 스틱으로 공을 쳐낼 수 있고, 상대방 골대에 공을 넣으면 점수가 올라감. 각 플레이어의 점수는 게임판 상단에 표시되며, END_SCORE 로 정의된 3 점을 먼저 내는 사람이 승리하게 된다.

한 점을 득점할 때마다 "Goal In" message 가 출력되며 게임은 멈춘다. 골이 들어간 후 다음 판 시작까지 몇 초가 남았는지 화면 중앙에 count down 이 표시된다. Count down 이 끝난 후에는 골을 먹힌 플레이어 쪽에서 공이 주어지며 게임이 다시 시작된다. 이 때 각 플레이어의 스틱도 기본 설정 위치로 돌아간다.

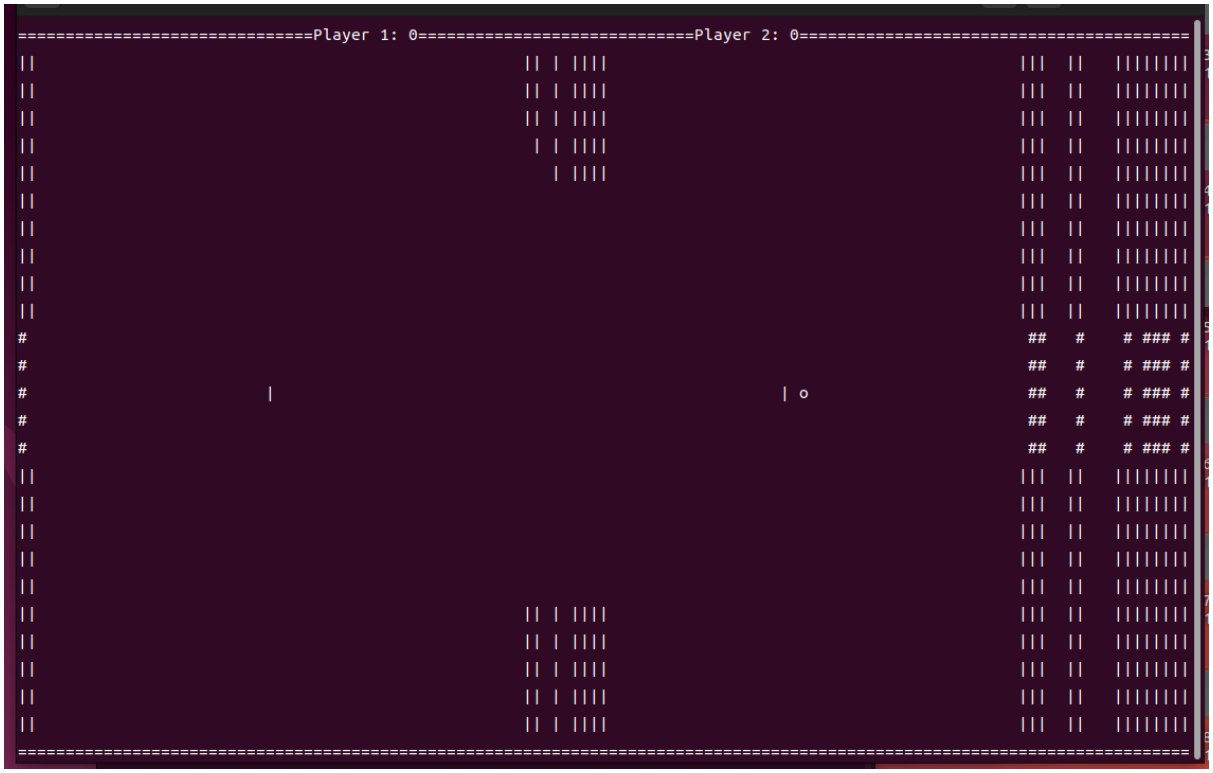


2-2-3. 게임 종료 조건

어떤 플레이어든 END_SCORE 에 도달하고 나면, parent process 에서는 waitpid()를 실행하고, child process 에서는 게임창에 "Game Over Press any key to exit" message 가 출력하며 게임 화면은 멈춘다. 이 때 아무 키나 입력하면 게임창이 닫히고, child process 가 종료된다. 이후 parent process 에서 터미널 창에 어떤 플레이어가 승리했는지 간단한 메시지가 출력한다.

3. 이슈사항

게임이 시작할 때의 COLS, LINES 값을 기준으로 게임판 크기를 설정하기 때문에, 게임 도중에 커맨드창의 크기를 바꾸면 게임판의 디자인이 이상해질 수 있다. 실제 게임이 진행되는 벽의 경계는 그대로 유지되기 때문에 기능상의 문제점은 없지만, 원활한 이용을 위해 임의로 윈도우의 크기를 조정해서는 안 된다.



4. 매뉴얼

Player 1 : w, a, s, d 로 스틱 조종

Player 2 : i, j, k, l 로 스틱 조종

Q : 강제종료

z : 공의 속도를 느리게 만듦

m : 공의 속도를 빠르게 만듦

각 플레이어의 스틱으로 공을 쳐서 상대방 골대(#으로 표시된 부분)에 넣으면 된다.

'Q' 또는 Ctrl + z 로 강제종료 할 수 있고, 'z'와 'm'으로 공의 속도를 조절할 수 있다.

전처리기 부분의 END_SCORE 값을 조절해서 몇 점을 목표로 할지 설정할 수 있다 (default 값은 3 이다.)