Technical University of Košice

Department of Computers and Informatics

Problem Set 6

DungeonWalking

Danyil Yedelkin

2020/2021

Assignment

Using the ncurses library, create a program (game, presentation or other artistic work), with the following minimal requirements:

- Project contains 2D world.
- Project meets at least 3 challenges:
  - Work with colors
  - Keyboard control (no Enter needed)
  - Multiple levels
  - Work in time (in the time the program is changed)
  - Work with command-line arguments
  - Work with files
- Project must be more complicated then the sample programs, with an adequate level of difficulty.

Game logic

The code consists of 3 functions:

```
void painting();                    // colors for painting
int main(int argc, char* argv[]); // the main function (with game logic etc.)
void MakeScreen();                 // make the game scene
```

Gameplay

The game is entirely coded inside the program.c file. To start the program, you also need to specify an additional empty file (moves.txt), into which information about the character's movement will be entered. From which we will read information in the future. You can move the character using the WASD keys and Key Up, Key Left, Key Down, Key Right. The variable “moves” reads the amount of movement of the main character and enters the data into the file “moves.txt”.

To run the game, you must write these commands:

```
@DESKTOP-RFDDG9I:~/zap$ gcc program.c $(ncursesw5-config --cflags) -o program -lncursesw -lm
@DESKTOP-RFDDG9I:~/zap$ ./program moves.txt
```


```
gcc program.c $(ncursesw5-config --cflags) -o program -lncursesw -lm
./program moves.txt
```


The map of the game:

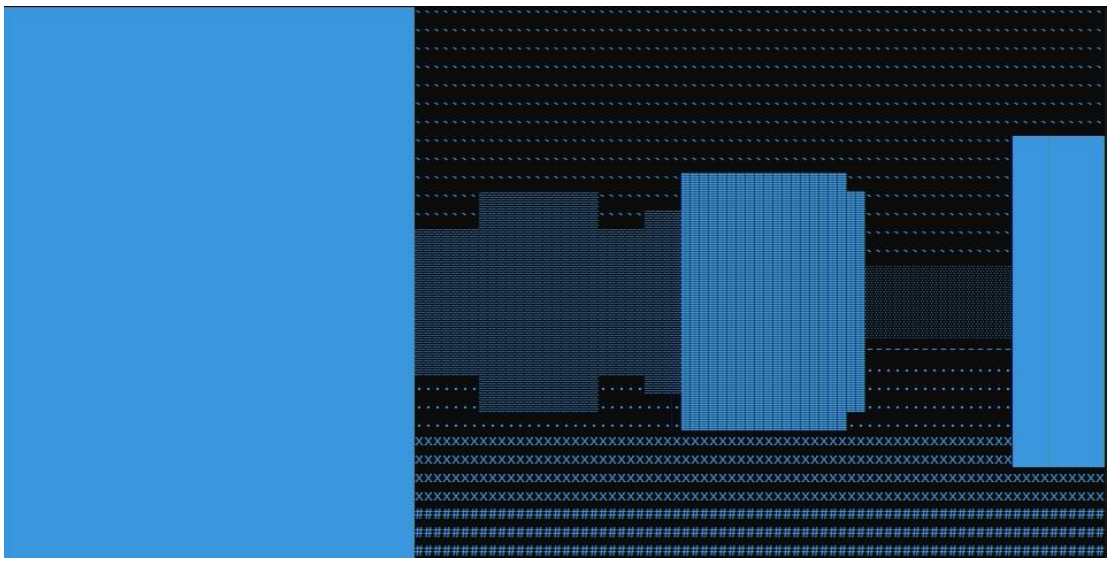
```
// creating a world map, where '#' - wall, '.' - space
char karta[] =
"################\
 #..............#\
 #..............#\
 #...########...#\
 #.......#..#...#\
 #..........#...#\
 #..............#\
 #...########...#\
 #...#...#..#...#\
 #...#...#..#...#\
 #..............#\
 #..............#\
 #...########...#\
 #..............#\
 #..............#\
 ################";
```

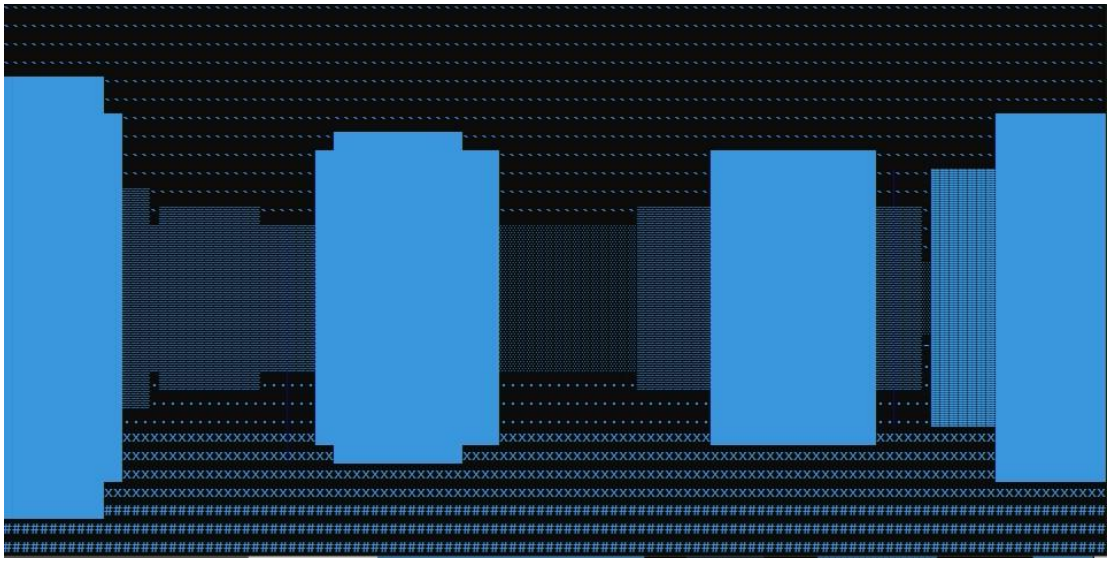
The walking system was created in the switch()

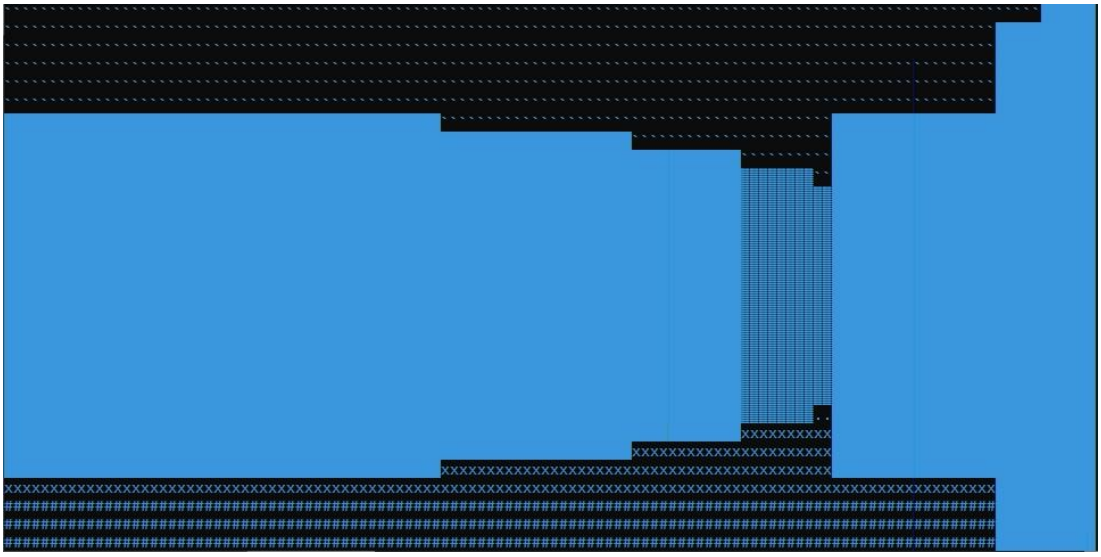
```
switch((input = getch())){
    //work with time
```

```
switch((input = getch())){
    //work with time
    timeout(300);

    // Move forward and collisions
    case KEY_UP:
    case 'W':
    case 'w':
        PervoyePolozenie = PervoyePolozenie + sinf(TreteyePolozenie) * 0.5f;

        VtoroyePolozenie = VtoroyePolozenie + cosf(TreteyePolozenie) * 0.5f;

        if(karta[(int)PervoyePolozenie * mapWidth + (int)VtoroyePolozenie] == stena){
            PervoyePolozenie = PervoyePolozenie - sinf(TreteyePolozenie) * 0.5f;
            VtoroyePolozenie = VtoroyePolozenie - cosf(TreteyePolozenie) * 0.5f;
        }
        moves++;

        break;

    case KEY_DOWN:
    case 'S':
    case 's':
        // Move backward and collisions
        PervoyePolozenie = PervoyePolozenie - sinf(TreteyePolozenie) * 0.5f;
        VtoroyePolozenie = VtoroyePolozenie - cosf(TreteyePolozenie) * 0.5f;

        if(karta[(int)PervoyePolozenie * mapWidth + (int)VtoroyePolozenie] == stena){
            PervoyePolozenie = PervoyePolozenie + sinf(TreteyePolozenie) * 0.5f;
            VtoroyePolozenie = VtoroyePolozenie + cosf(TreteyePolozenie) * 0.5f;
        }
        moves++;

        break;

    case KEY_LEFT:
    case 'A':
    case 'a':
        // Rotate left
        TreteyePolozenie = TreteyePolozenie - 0.1f;
        moves++;

        break;

    case KEY_RIGHT:
    case 'D':
    case 'd':
    // Rotate right
        TreteyePolozenie = TreteyePolozenie + 0.1f;
        moves++;

        break;

    default:{
        // Any key to exit
        vichod = 0;

        break;
    }
}
```

The gameplay’s screenshots:







Where you can see, that the distance to walls is different, and displays it by the symbols (‘#’, ‘x’, ‘,’, etc).

The distance was created by this part of the code:

```
// creating a game function
while(vichod == 1){
    for(int colona = 0; colona < ShirinaKarty; colona++){
        // for each column, calculate the projected ray angle into world space
        float dalnostZrenia = (TreteyePolozenie - zrenie / 2.0f) + ((float)colona / (float)ShirinaKarty) * zrenie;
        float rastoyanie = 0.0f;

        int StenkaVLob = 0;

        // creating unit vector for ray in player space
        float ZrenieX = sinf(dalnostZrenia);
        float ZrenieY = cosf(dalnostZrenia);

        // find distance to wall
        while(!StenkaVLob && rastoyanie < distancia){
            // crate resolution
            rastoyanie = rastoyanie + 0.5f;

            int proverkaColony = (int)(PervoyePolozenie + ZrenieX * rastoyanie);

            int proverkaRow = (int)(VtoroyePolozenie + ZrenieY * rastoyanie);

            // test our code if ray is out of bounds
            if(proverkaColony < 0 || (proverkaColony > mapWidth + 1) || proverkaRow < 0 || (proverkaRow > mapHeight + 1)){
                StenkaVLob = 1;
                rastoyanie = distancia;
            } else{
                if(karta[proverkaColony * mapWidth + proverkaRow] == stena){
                    StenkaVLob = 1;
                }
            }
        }

        // calculate distance to ceiling and floor
        int ceiling = (float)(VisotaKarty / 2.0f) - VisotaKarty / ((float)rastoyanie);
        int floor = VisotaKarty - ceiling;

        // shader walls based on distance
        const wchar_t* shade;

        // if we close to the wall
        if(rastoyanie <= distancia / 4.0){
            shade = L"\x2588";
        }
        // if we further to the wall
        else if(rastoyanie <= distancia / 3.0){
            shade = L"\x2593";

        } else if(rastoyanie <= distancia / 2.0){
            shade = L"\x2592";

        } else if(rastoyanie <= distancia){
            shade = L"\x2591";

        } else{
            shade = L" ";

        }

        for(int row = 0; row < VisotaKarty; row++){

            if(ceiling  + 1 > row){
                mvaddch(row, colona, '`');
```

```
    for(int row = 0; row < VisotaKarty; row++){

        if(ceiling  + 1 > row){
            mvaddch(row, colona, ' ');

        } else if(row > ceiling  && row < floor + 1){
            mvaddwstr(row, colona, shade);

        } else{
            // Shade floor based on distance
            float b = 1.0f - (((float)row - VisotaKarty / 2.0f) / ((float)VisotaKarty / 2.0f));

            if (b < 0.25){
                shade = L"#";

            } else if (b < 0.5){
                shade = L"x";

            } else if (b < 0.75){
                shade = L".";

            } else if (b < 0.9){
                shade = L"-";

            } else{
                shade = L" ";

            }

            mvaddwstr(row, colona, shade);
        }
    }

    refresh();
}
```

By pressing any another keys, you will finally close the game.

Conclusion

The game code has its drawbacks. When you enter Key Up, Key Left, Key Down, Key Right, the variable does not count your movements. The error exists due to the inability to translate Keys Up, Left, Down, Right into the system ASCII. The second error exists when entering any non-English keyboard keys. My code can be supplemented with a timer that counts the time of the passage of the game.