STAR TREK WARP WARS PRESENTATION

CME1252 PROJECT BASED LEARNING - II
PROJECT II

Prepared By:
2020510034 Yusuf GASSALOĞLU
2020510062 Özgür Doruk RÜSVENLİ
202050069 Mert Emin TEZCAN
2021510078 Zeynep ÇOĞALAN

22.04.2022

İZMİR

OUTLINE

01 Introduction

02 Progress Summary

03 Problems Encountered

04 Algorithms and Solution Strategies

05 Screenshots

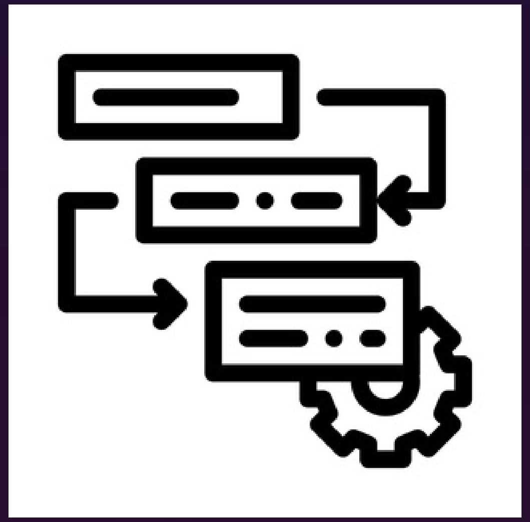
06 Conclusion

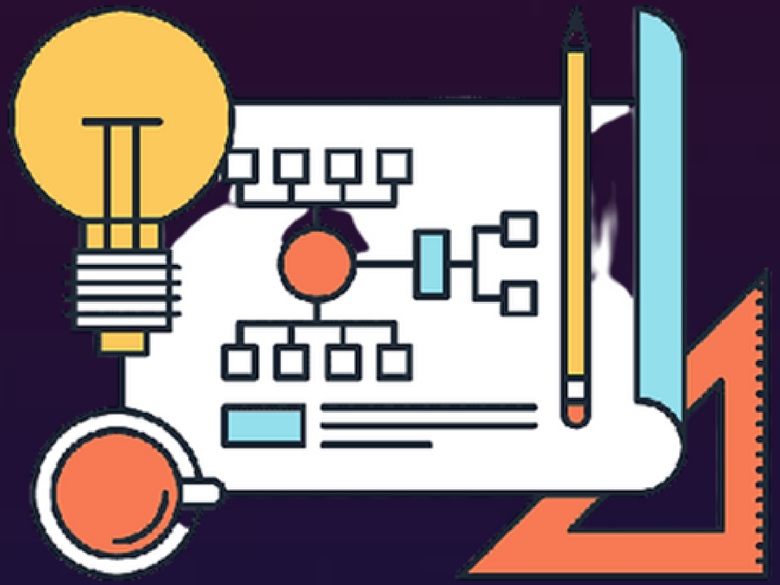
07 References

08 Questions

INTRODUCTION

- The aim of the project is to develop a software application for the "Star Trek" game.
- Star Trek is a single-player space maze game played against robots by collecting treasures.





REQUIREMENTS

1. Backpack Algorithm

2. Number Algorithm

3. Computer Algorithm

4. Device Algorithm

SCHEDULING

WEEK-1	WEEK-2	WEEK-3	WEEK-4	WEEK-5
PLAYER MOVEMENTS	BACKPACK	NUMBERS	DEVICES	COMPUTER
SCREEN	TIMING	BACKPACK	COMPUTER	SCREEN

COMPLETED TASKS AND TASK SHARING

No	Tasks	1.Week	2.Week	3.Week	4.Week	5.Week
1	Design of classes	🟨				
2	Menu Screen	🟦🟨			🟩	🟪
3	Player Movements	🟦				
4	Backpack Implementation		🟪	🟪		
5	Computer				🟨	🟨
6	Numbers (Static and Moving)			🟨		
7	Input queue implementation			🟩		
8	Timing		🟨			
9	Devices				🟪	

Yusuf: 🟨 Özgür: 🟩

Mert: 🟦 Zeynep: 🟪

INCOMPLETED TASKS

There is no incompleted task.

ADDITIONAL IMPROVEMENTS

MENU AND SOUND EFFECTS

COLORFUL INTERFACE AND GAMEPLAY

CHOOSING NUMBER OF COMPUTERS

HOW TO PLAY AND RULES MENU

PROBLEMS ENCOUNTERED

No problems were encountered during the project process.

ALGORITHMS AND SOLUTION STRATEGIES

BACKPACK

```
findValue() {
 for()
  for() //to check all array
   if(maze[px,py] != '  ' and objectArray[py,px] != null)
     backPack_Identical_Numbers(number)
     maze[px,py] = '  '
     objectArray[py,px] = null
     for() //to find the number in number array
      if(number's x == px and number's y == y)
         //to delete the number's object
         number[k] = null
          break; }
```

```
backPack_Identical_Numbers(Number number) {
    if(getBackpack().peek().equals('=' or '*')
      if(1)  playerScore + 1
      else()
      playerScore + numberScore
      push(number)
    else()
      switch(number)
      case 1: playerScore + 1
      case 2:
      playerScore + numberScore
      if(top element = 2)   //get the power or device
      else if(top element = other numbers)  //pop the top element
      else if(backpack.isEmpty)   //push number
```

TRAP

```
activateTrapDevice(x, y, true/false) {
  if(maze[x,y] == 'C')
     for() //to find computer's object
       if(computerX == x and computerY == y)
         //set computer.moving(true/false)
         break;
  else
     for() //to find numbers's object
       if(numberX == x and numberY == y)
          if(number == 4 or 5)
            //set number.moving(true/false)
         break;  }
```

WARP

```
activateWrapDevice(x, y) {
  if(maze[x,y] == 'C')
    for() //to find computer's object
      if(computerX == x and computerY == y)
        //Player + computerScore
        //delete the object
        break;
  else
    for() //to find numbers's object
      if(numberX == x and numberY == y)
          //Player + numberScore
          //delete the object
        break; }
```

TIMER

```
queueTimer = 3000;
time1 = System.nanoTime()
While(TRUE) :  // game loop
time2 = System.nanoTime()
time = (time2 - time 1)/1000
If (time > queueTimer):
//necessary functions
queueTimer += 3000;
```

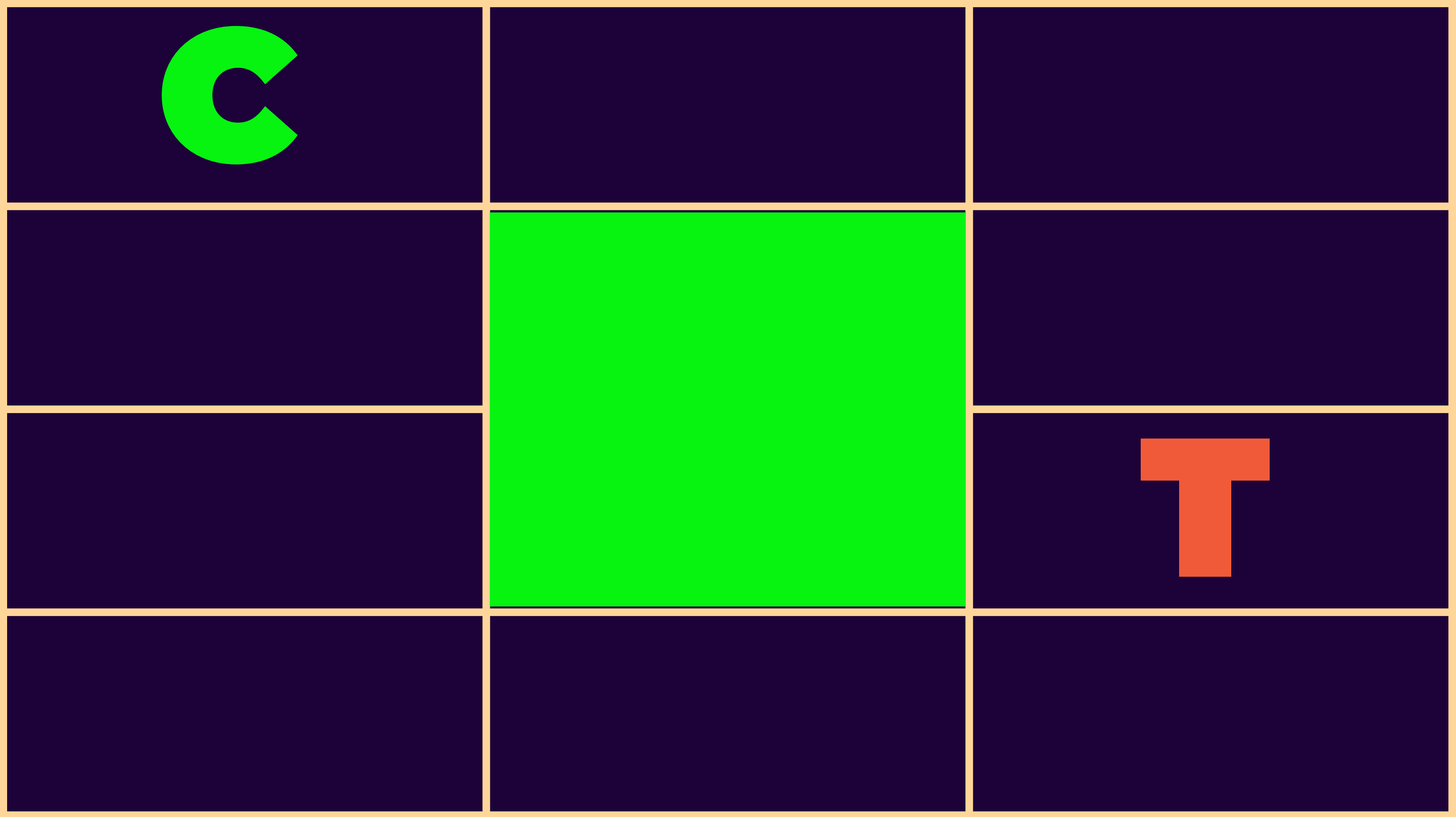
Queue timer pseudo code

NUMBER AND COMPUTER ALGORITHM

```
mnac= 500; //movingNumberAndComputerTimer
WHILE TRUE: //game loop
    if (time > mnac):
        FOR i=0 to 1265:
            IF numberArray[i] is moving number:
                move the number randomly
            pathFinder(computerArray[i]) //call pathfinder
        mnac += 500;
```

PATHFINDER

C: computer T: Target



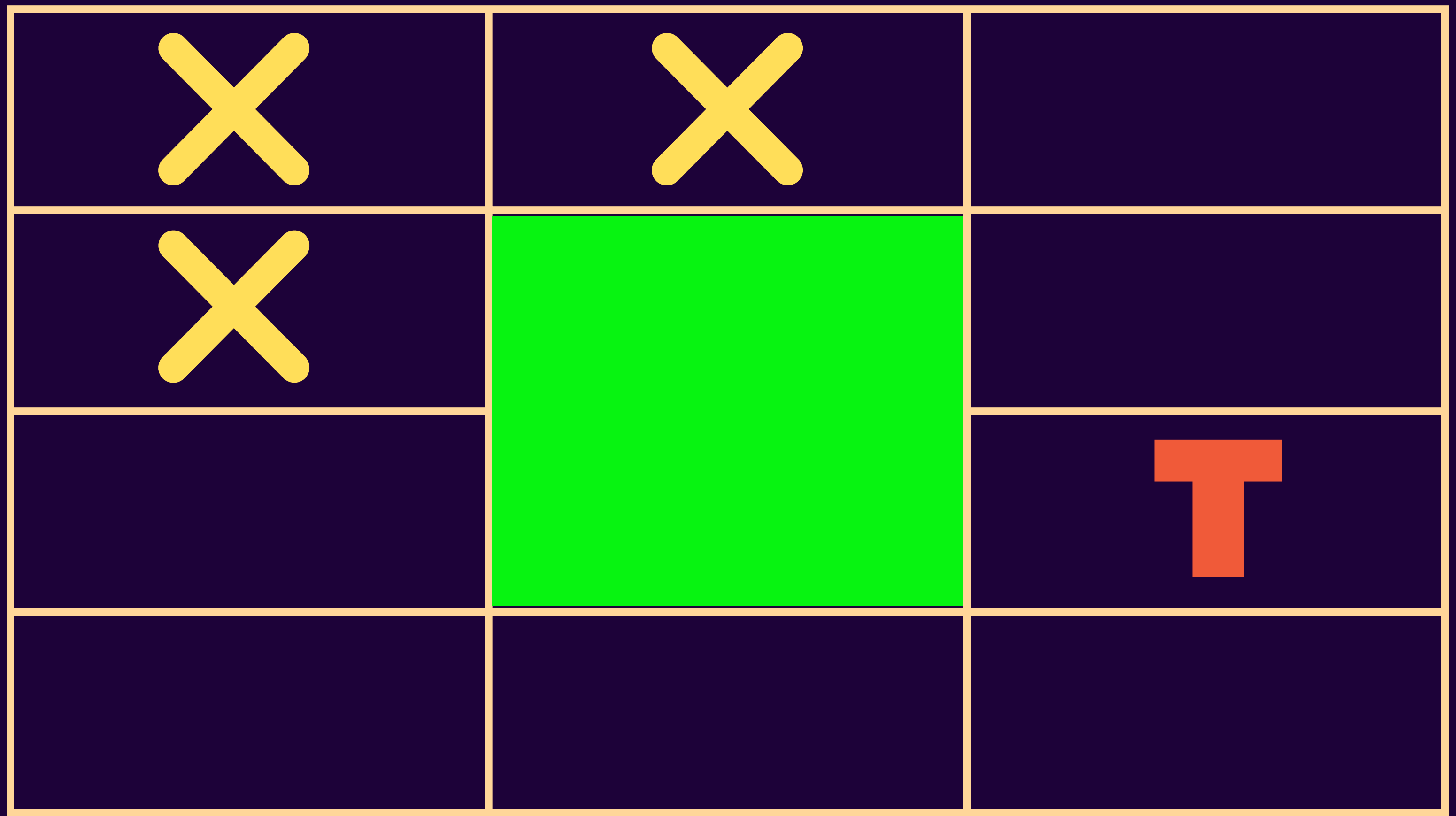
Queue

"R" , "D"

Check order:
Right , Up, Left , Down

PATHFINDER

Dequeue the element and walk corresponding way and check again the maze. Enqueue the new elements.



OLD PATH IS R

Queue

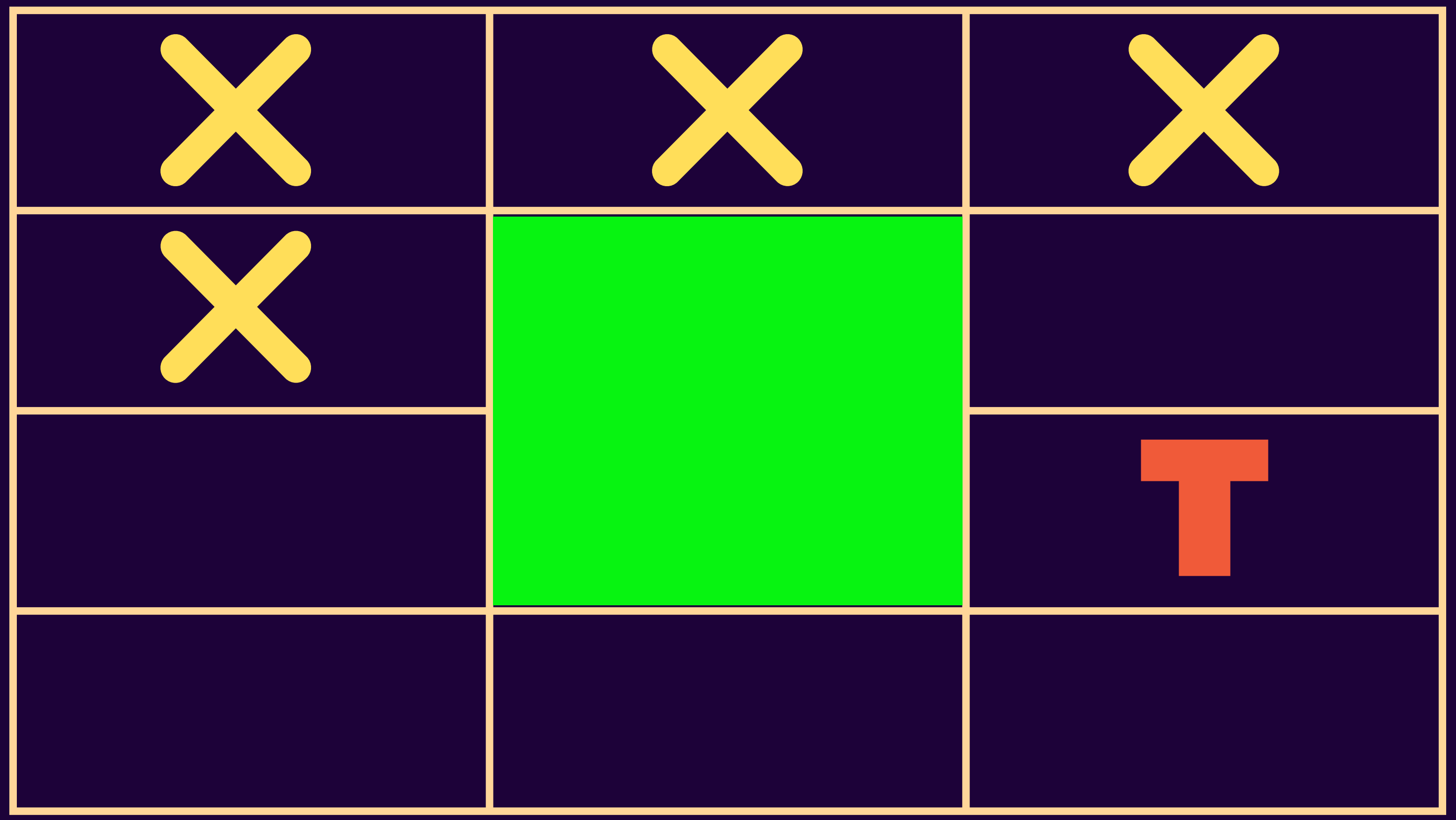
"D", "RR"

Check order:

Right , Up, Left , Down

PATHFINDER

Dequeue the element and walk corresponding way and check again the maze. Enqueue the new elements.



OLD PATH IS D

Queue

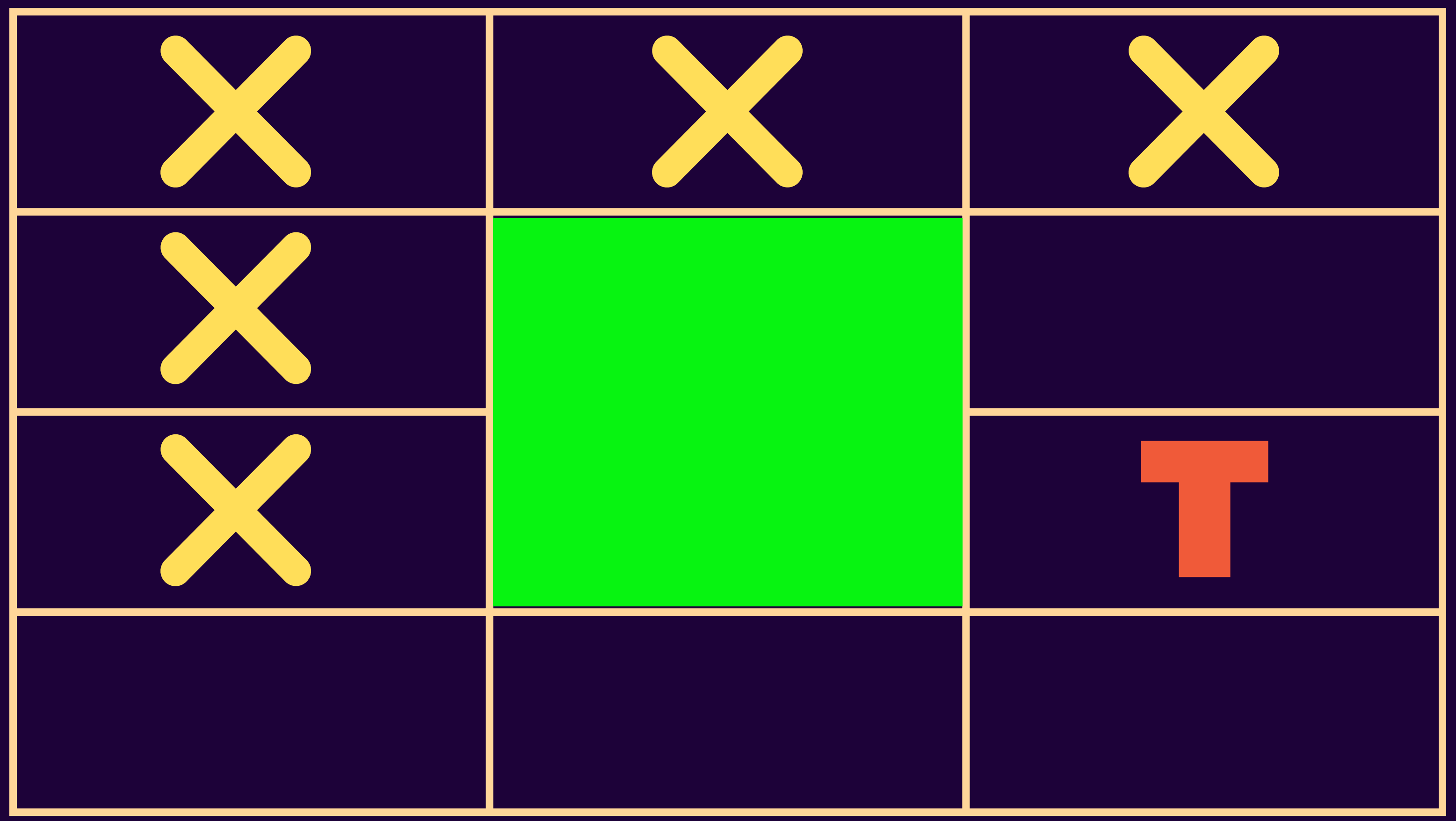
"RR" , "DD"

Check order:

Right , Up, Left , Down

PATHFINDER

Dequeue the element and walk corresponding way and check again the maze. Enqueue the new elements.



OLD PATH IS RR

Queue

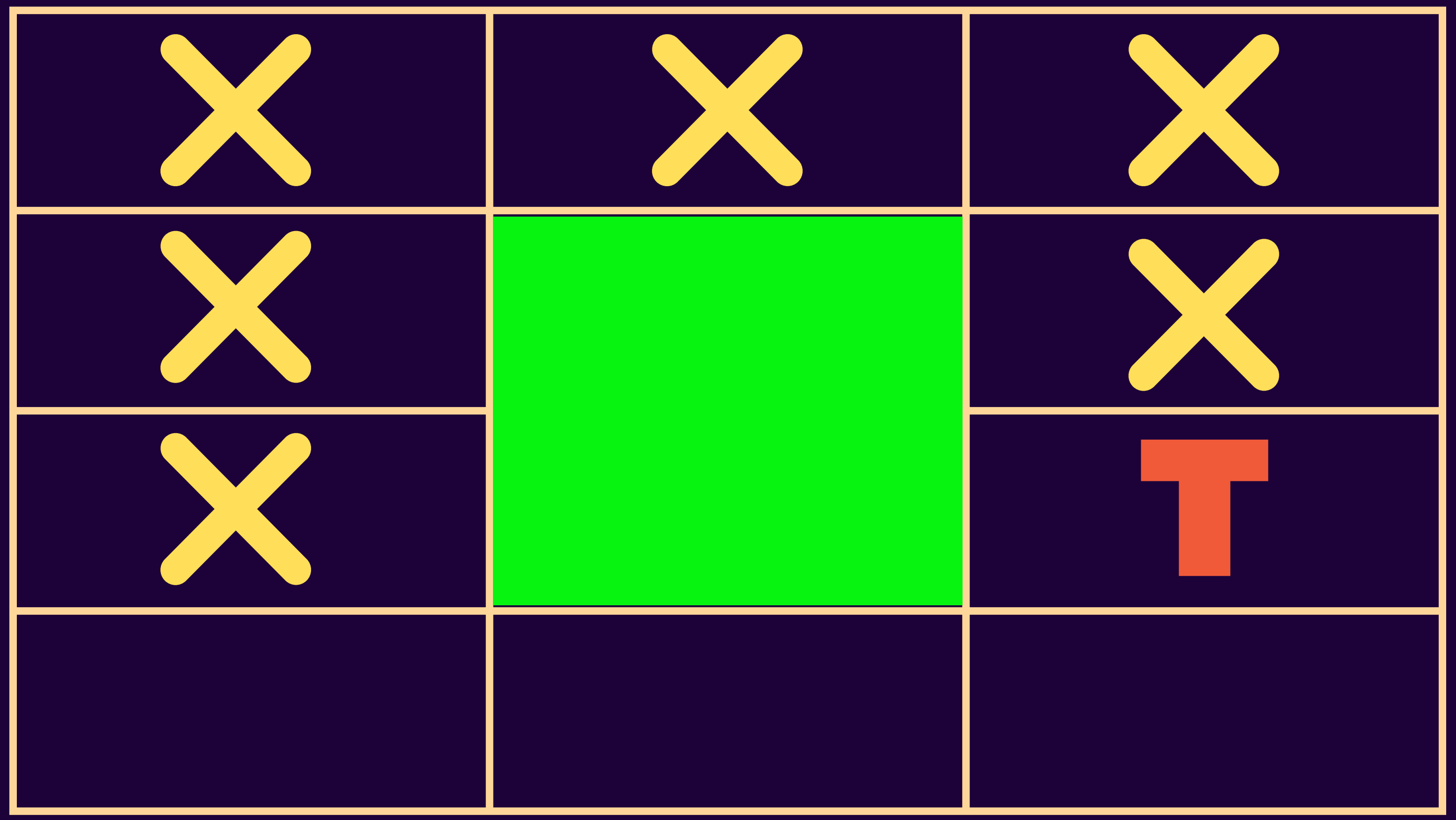
"DD", "RRD"

Check order:

Right , Up, Left , Down

PATHFINDER

Dequeue the element and walk corresponding way and check again the maze. Enqueue the new elements.



OLD PATH IS DD

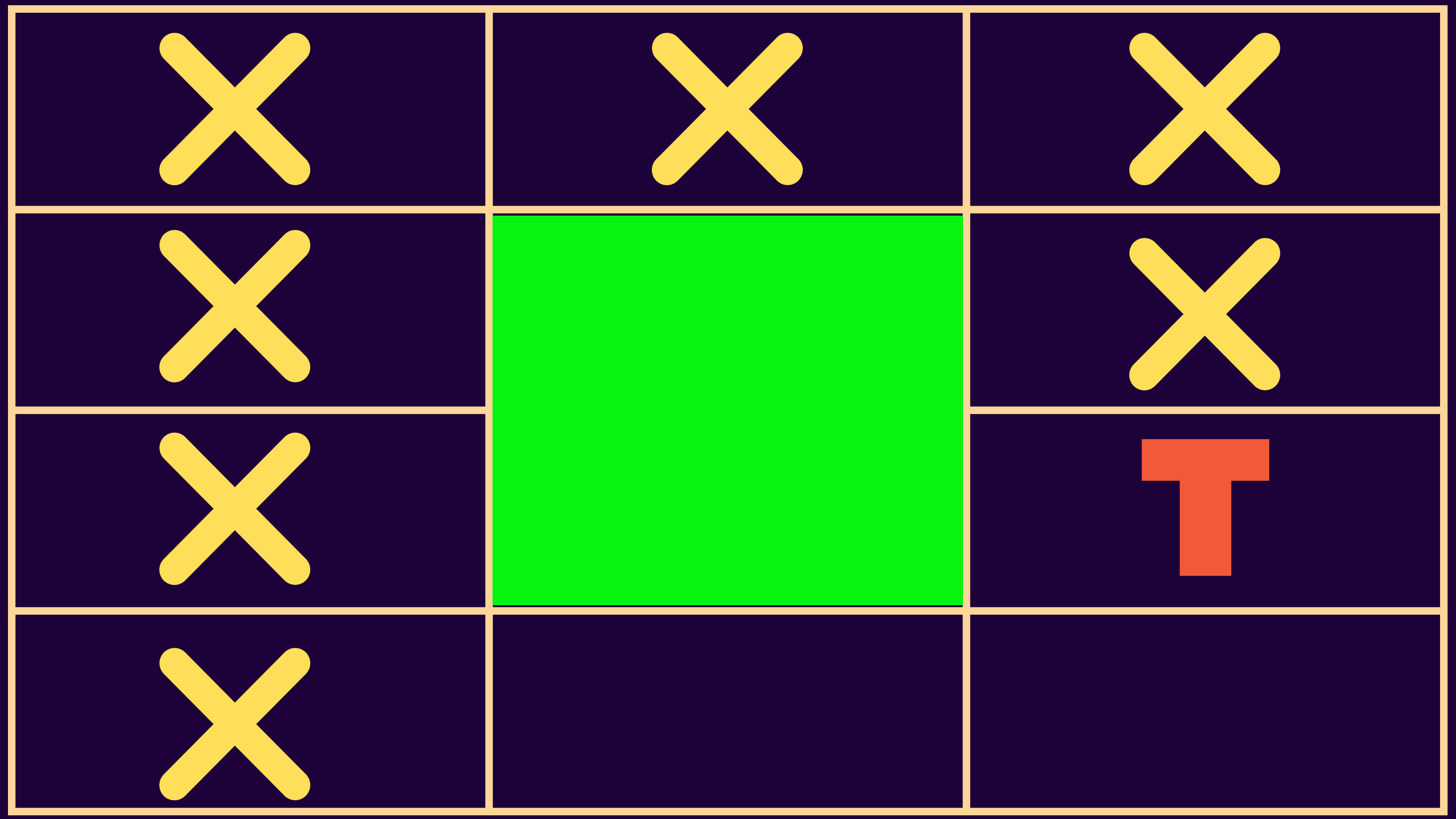
Queue

"RRD", "DDD"

Check order:
Right , Up, Left , Down

PATHFINDER

Dequeue the element and walk corresponding way and check again the maze. Enqueue the new elements.



OLD PATH IS RRD

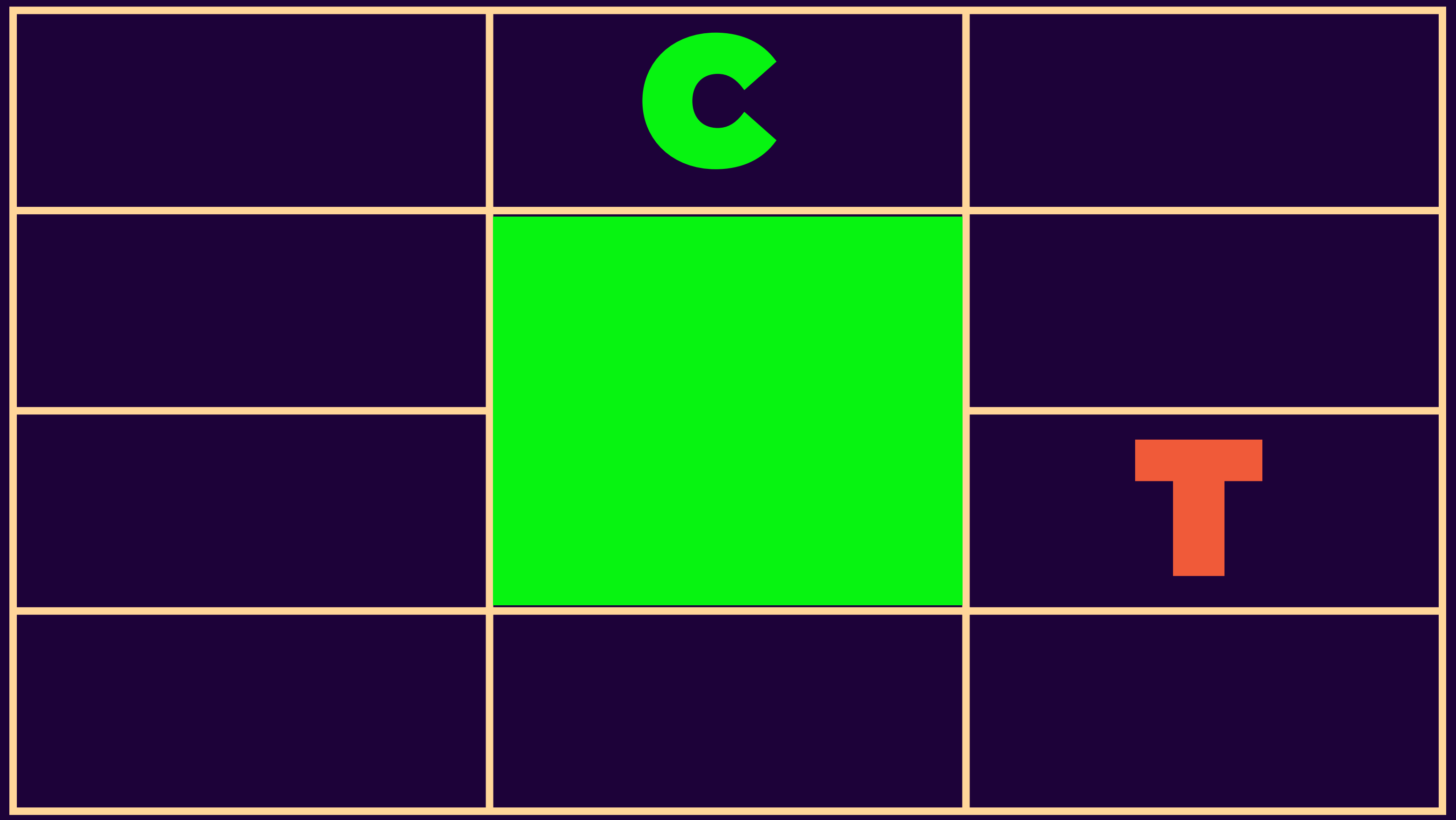
Queue

"DDD"

Bottom cell is target and the path is:
RRDD

PATHFINDER

C: computer T: Target

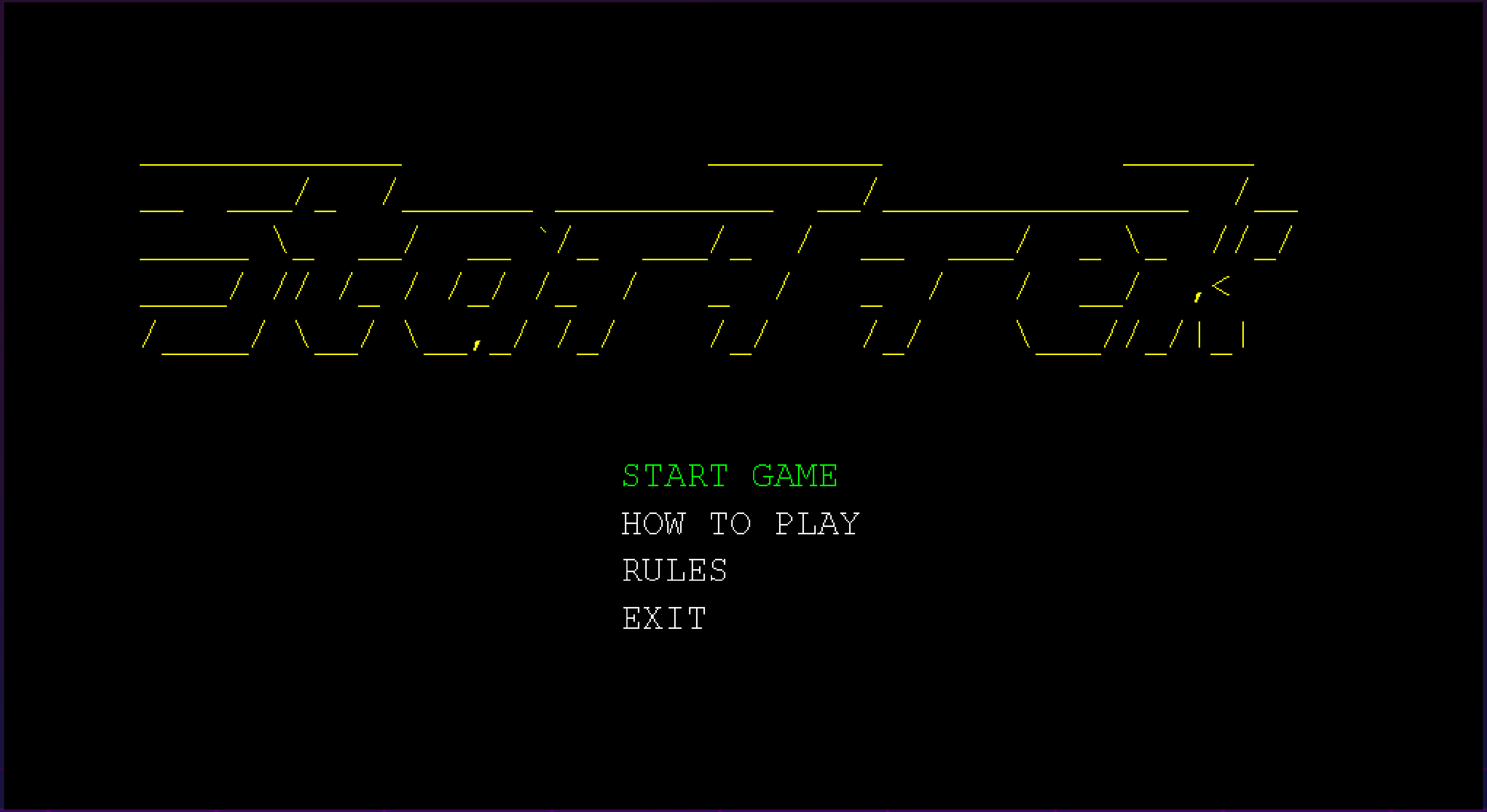


Queue

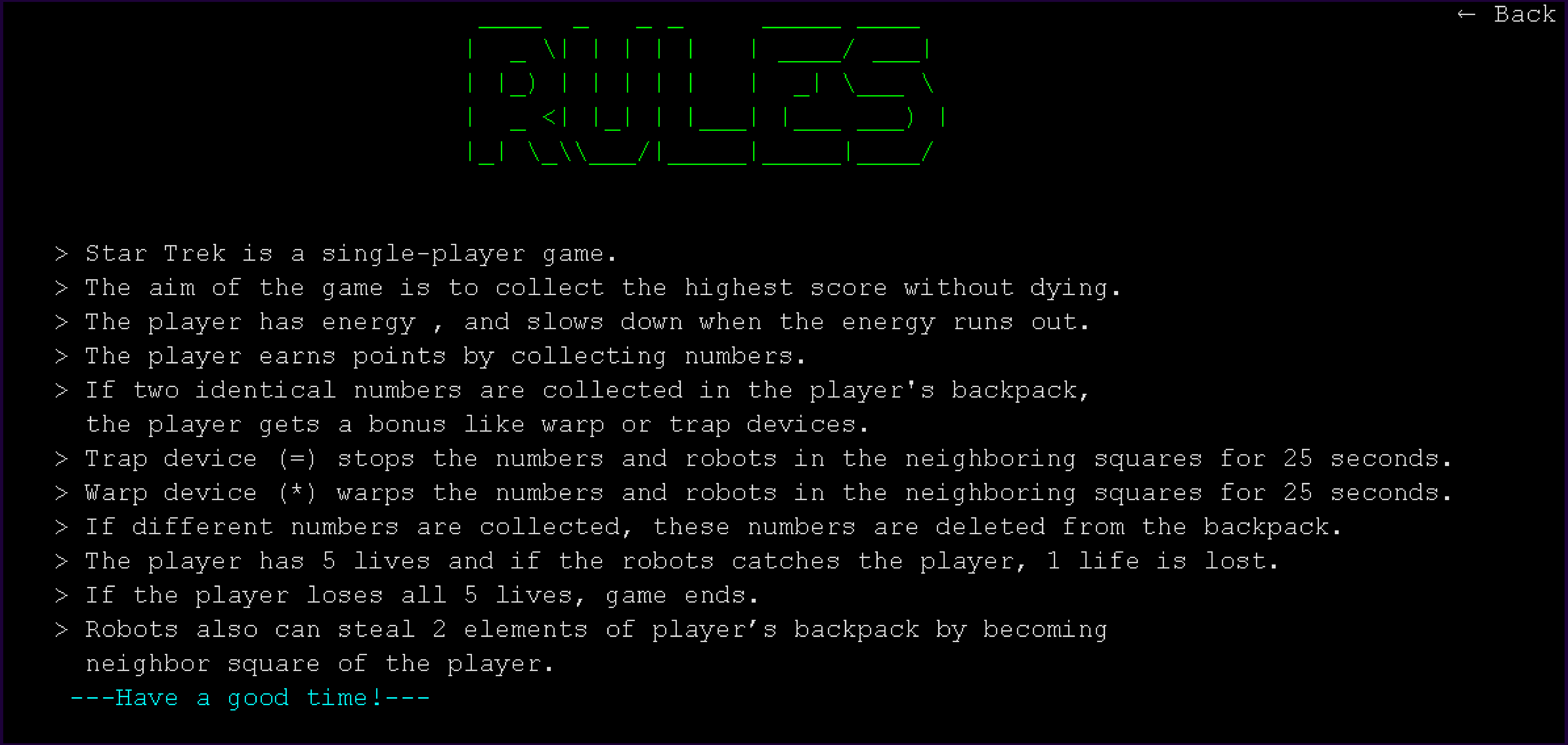
"R" , "D"

Therefore, computer goes right.

SCREENSHOTS

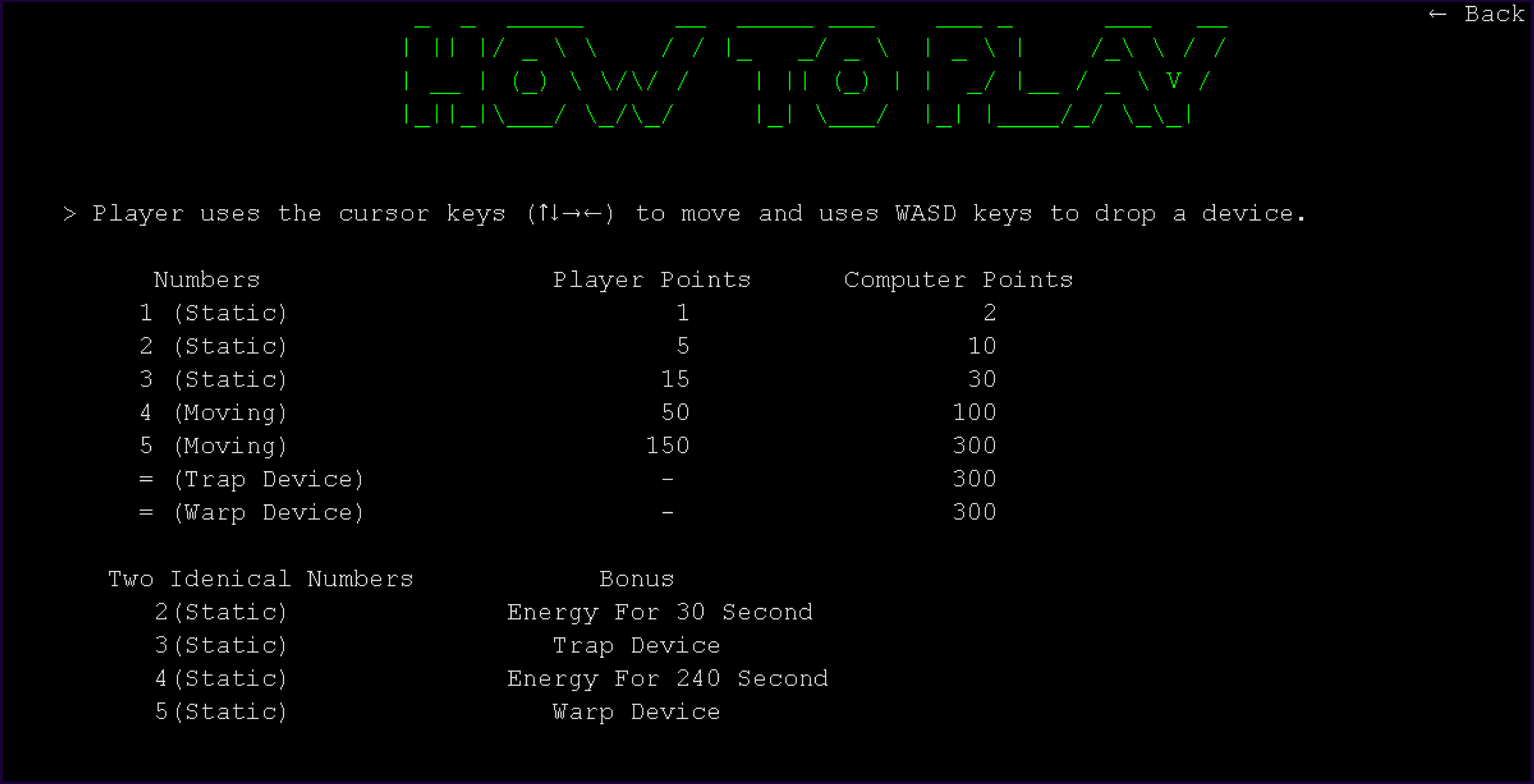


```
START GAME
HOW TO PLAY
RULES
EXIT
```



```
 _____  _    _ _      ______  _____
|  __ \| |  | | |    |  ____|/ ____|
| |__) | |  | | |    | |__  | (___
|  _  <| |  | | |    |  __|  \___ \
| | \ \| |__| | |____| |____ ____) |
|_|  \_\\____/|______|______|_____/
```

```
> Star Trek is a single-player game.
> The aim of the game is to collect the highest score without dying.
> The player has energy , and slows down when the energy runs out.
> The player earns points by collecting numbers.
> If two identical numbers are collected in the player's backpack,
  the player gets a bonus like warp or trap devices.
> Trap device (=) stops the numbers and robots in the neighboring squares for 25 seconds.
> Warp device (*) warps the numbers and robots in the neighboring squares for 25 seconds.
> If different numbers are collected, these numbers are deleted from the backpack.
> The player has 5 lives and if the robots catches the player, 1 life is lost.
> If the player loses all 5 lives, game ends.
> Robots also can steal 2 elements of player’s backpack by becoming
  neighbor square of the player.
 ---Have a good time!---
```

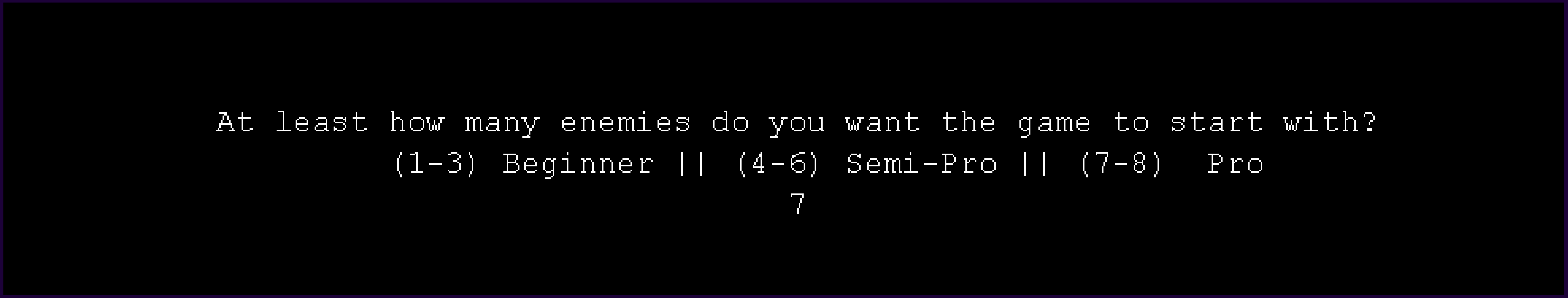


HOW TO PLAY

> Player uses the cursor keys (↑↓→←) to move and uses WASD keys to drop a device.

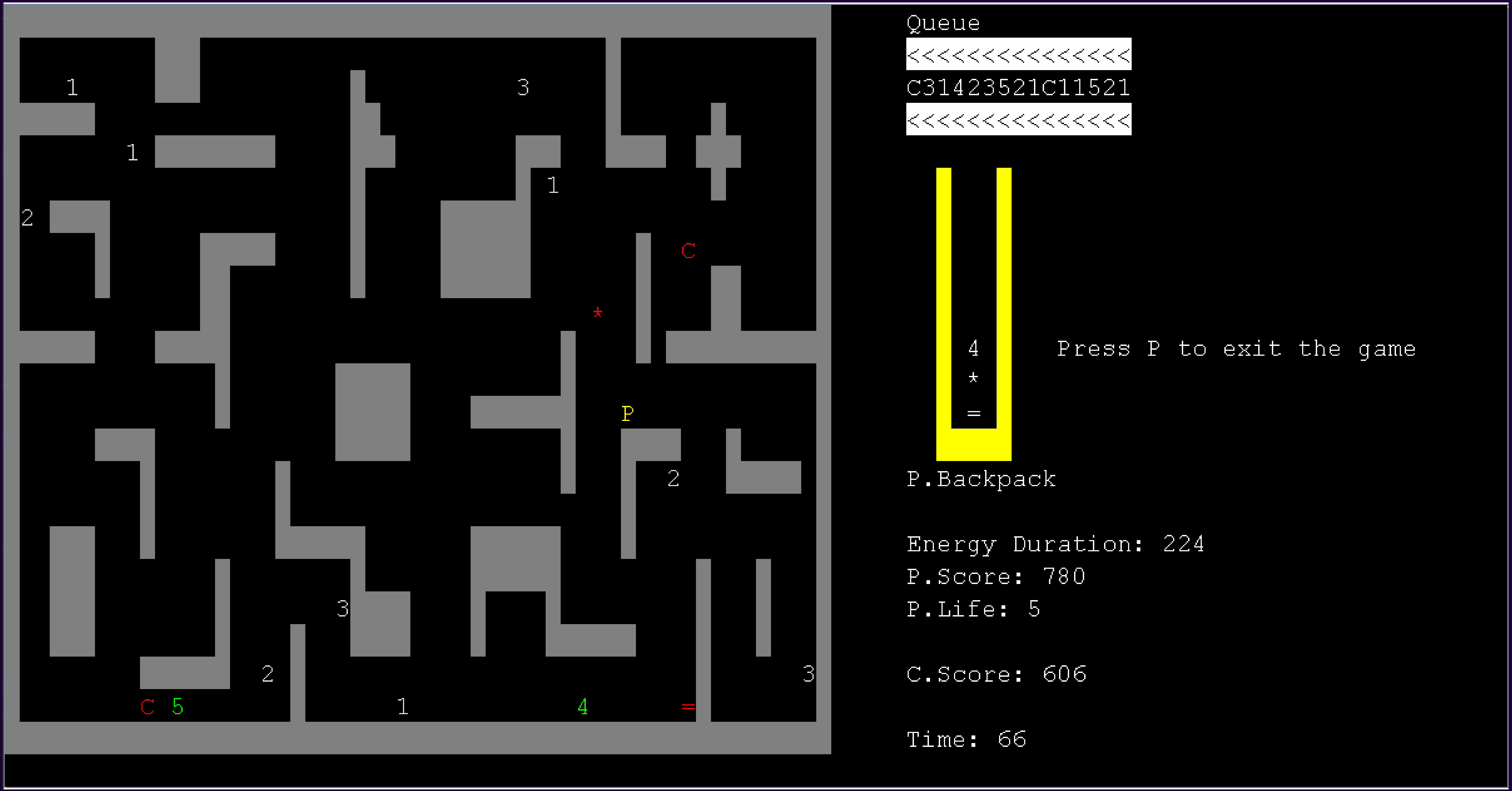
Numbers	Player Points	Computer Points
1 (Static)	1	2
2 (Static)	5	10
3 (Static)	15	30
4 (Moving)	50	100
5 (Moving)	150	300
= (Trap Device)	-	300
= (Warp Device)	-	300

Two Idenical Numbers	Bonus
2(Static)	Energy For 30 Second
3(Static)	Trap Device
4(Static)	Energy For 240 Second
5(Static)	Warp Device

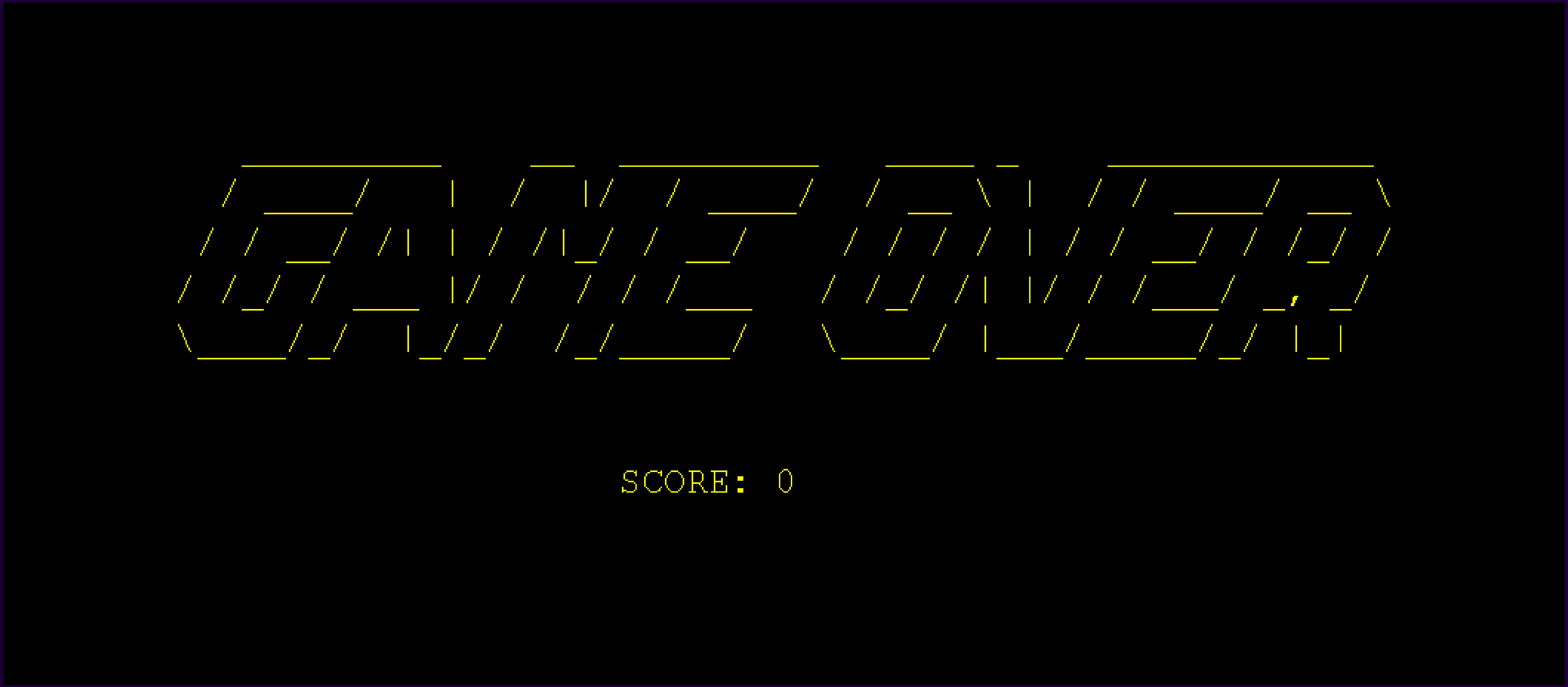


```
At least how many enemies do you want the game to start with?
      (1-3) Beginner || (4-6) Semi-Pro || (7-8)  Pro
                            7
```

Difficulty Screen



```
Queue
<<<<<<<<<<<<<<<
C31423521C11521
<<<<<<<<<<<<<<<

   4
   *
   =        Press P to exit the game

P.Backpack

Energy Duration: 224
P.Score: 780
P.Life: 5

C.Score: 606

Time: 66
```



```
   _________    __  _________   ____ _    ____________ 
  / ____/   |  /  |/  / ____/  / __ \ |  / / ____/ __ \
 / / __/ /| | / /|_/ / __/    / / / / | / / __/ / /_/ /
/ /_/ / ___ |/ /  / / /___   / /_/ /| |/ / /___/ _, _/ 
\____/_/  |_/_/  /_/_____/   \____/ |___/_____/_/ |_|  

                    SCORE: 0
```

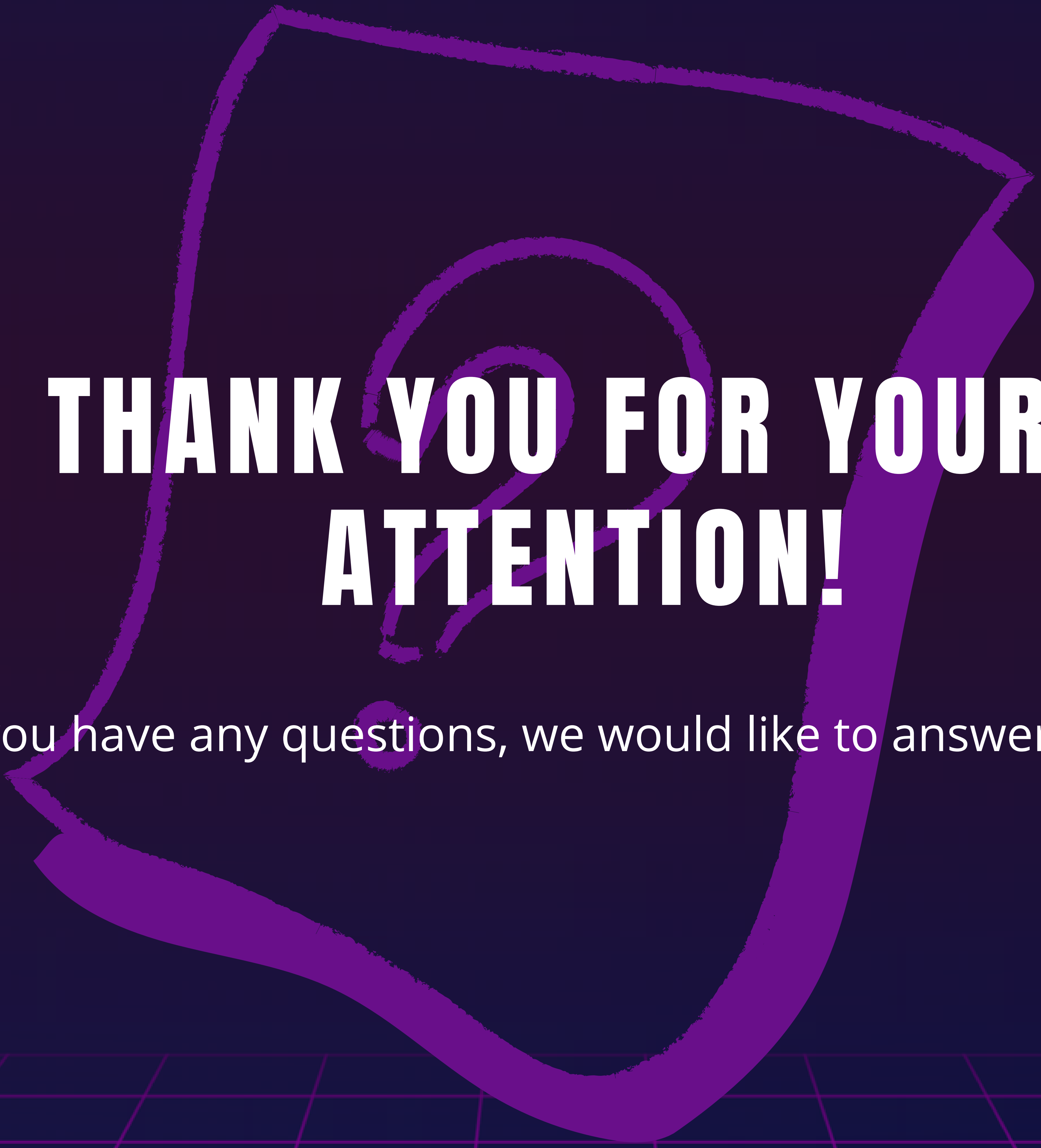
CONCLUSION

The project has been successfully completed.

The game was designed using object-oriented programming.

REFERENCES

1. Michael Burke and Konrad Zuse (1945). Breadth first search algorithm Retrieved April, 2022 from https://www.techwithtim.net/tutorials/breadth-first-search/
2. Java Pathfinder (n.d) Retrieved April , 2022 from https://www.geeksforgeeks.org/shortest-path-unweighted-graph/
3. E. Davis (1986). Ansi Art Retrieved April , 2022 from https://patorjk.com/software/taag/#p=display&f=Graffiti&t=Type%20Something%20



THANK YOU FOR YOUR ATTENTION!

If you have any questions, we would like to answer them.