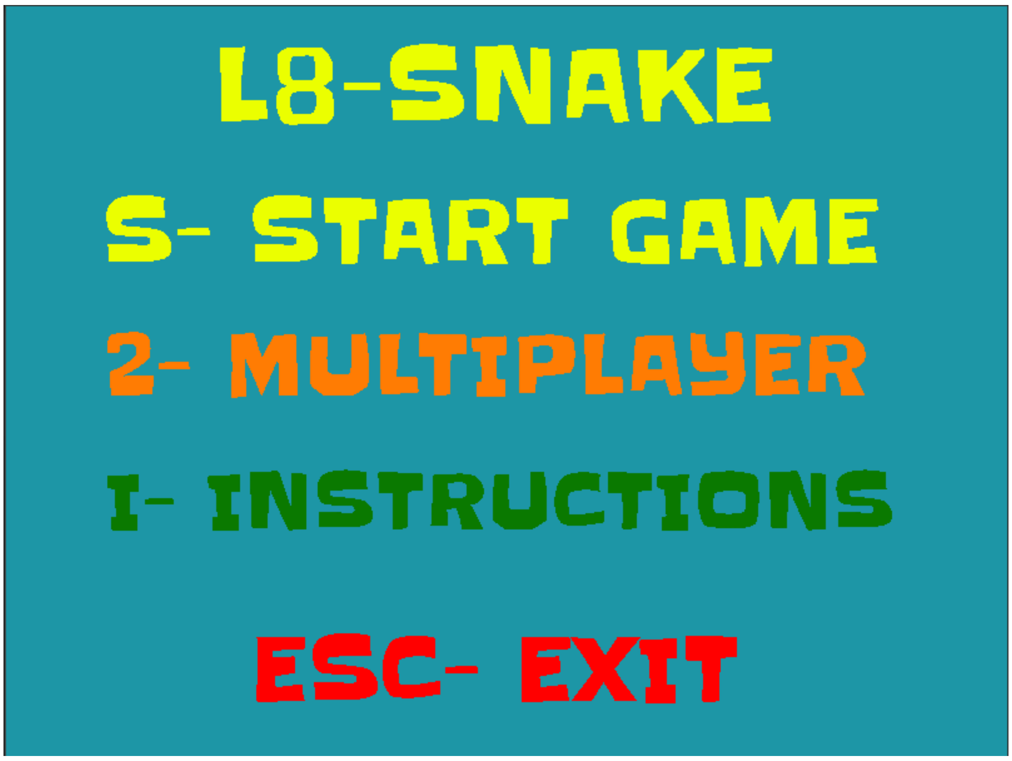
**Faculdade de Engenharia da Universidade do Porto**

# L8-Snake

Laboratório De Computadores



**Turma 8 Grupo 5**

João Cordeiro up202205682@fe.up.pt

Nuno Fernandes up202206289@fe.up.pt

Gonçalo Matos up202108761@fe.up.pt

Artur Candiani up201900839@fe.up.pt

## TABLE OF CONTENTS

1. Instructions
   1.1 Menu
   1.2 Game
      1.2.1 Concept
      1.2.2 Play

2. Project Status
   2.1 Video Card
   2.2 Keyboard
   2.3 Mouse
   2.4 RTC
   2.5 Collisions

3. Code Organitazion/Structure
   3.1 proj.c
   3.2 menu.c
   3.3 snakeController.c
   3.4 sprites.c
   3.5 keyboard.c
   3.6 timer.c
   3.7 video.c
   3.8 utils.c

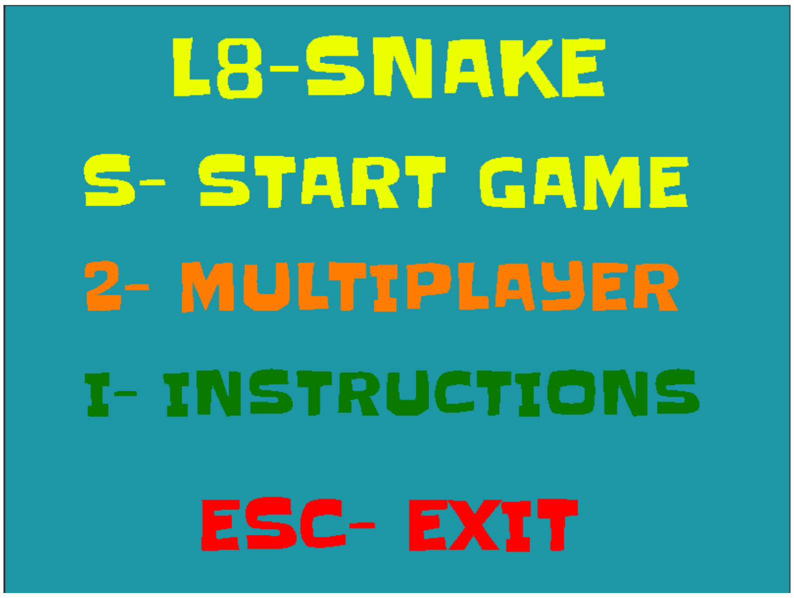
4. Implementation Details

5. Conclusion

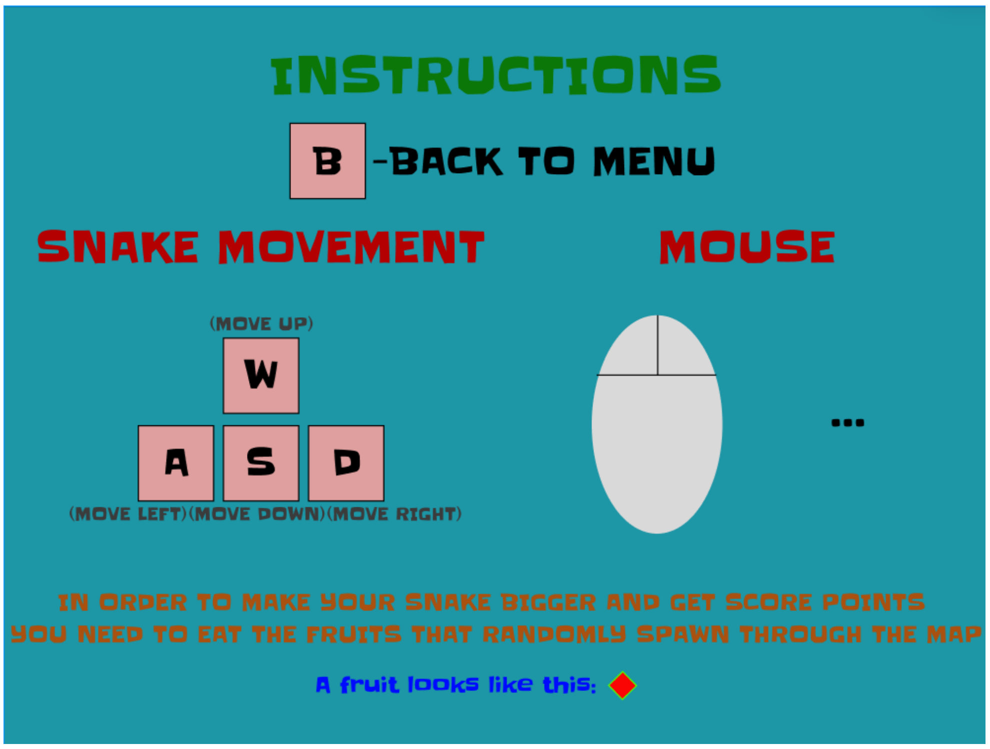
## 1. Instructions

### 1.1 Menu

When the game is launched, it shows the following menu. There, you can see you can see the name of the game (**L8-SNAKE**) and many options. In order to start the game, you need to press “S”. If you don’t know how to play, press “I” and it will show a display of the instructions. In order to exit the game, you need to press “ESQ”.



The instructions of L8-SNAKE are displayed as shown below. If you want to go back to the initial menu, just press “B”.



IN ORDER TO MAKE YOUR SNAKE BIGGER AND GET SCORE POINTS YOU NEED TO EAT THE FRUITS THAT RANDOMLY SPAWN THROUGH THE MAP

A fruit looks like this:

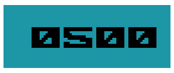
## 1.2 Game

### 1.2.1 Concept

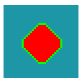
Our project was to creat a version of the very well-known “Snake” game. It is really similar to the one everyone knows. You can move your snake using the WASD keys and the goal of the game is to eat the most fruits in order to improve your score. Each fruit equals to 100 score points and they increase the size of your snake.

### 1.2.2 Play

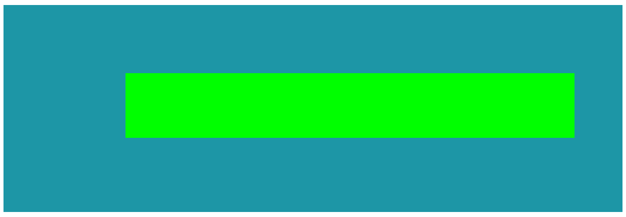
In the game, your score appears in the bottom right corner of the screen and it's represented like you can see in this picture (when the score is 500 points for example):



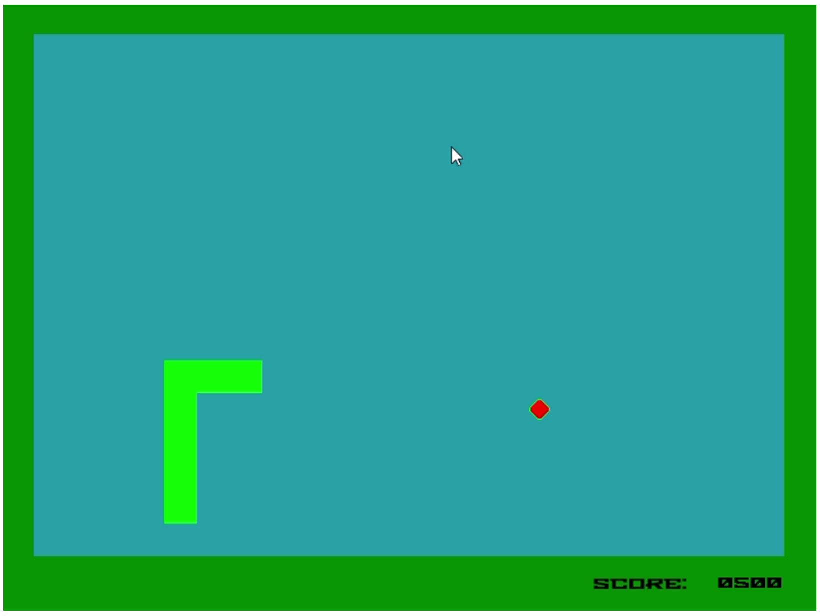
A fruit has this design and randomly spawns throughout the screen, so you need to move your snake to eat them:



The snake has this design and it becomes bigger and bigger the more fruits are eaten by the player:



Here is an example of the game screen from a random game and time of the game, for better understanding:



## 2. Project Status

### 2.1. I/O Devices Table

| Device | What for | Interruption |
|---|---|---|
| Timer | | Yes |
| Keyboard | -Move the snake<br>-Select menu options | Yes |
| Mouse | | Yes |
| Video Card | -Display Menu<br>-Display Game | No |

### 2.2 Timer

The timer 0 is used to render new frames whenever an interrupt occurs (timer 1 and 2 not implemented).

Interrupts are subscribed in the function **waiting_ESC_key(Snake* s, Fruit* f)** (proj.c)

Timer is configured to work at 30 frames per second by calling the function **timer_set_frequency(uint8_t timer, uint32_t freq) (**timer.c**).**

#### 2.2.1 Most relevant timer functions

proj.c:

- **waiting_ESC_key(Snake* S, Fruit* f)** - subscribes, checks and unsubscribes timer interrupts;

timer.c:

- **timer_set_frequency(uint8_t timer, uint32_t freq)** – sets a given frequency to the given timer;

### 2.3 Keyboard

The keyboard is used to get user input. It’s used to control the snake, navigate trough the menu and exit the game.
Its interrupts are exclusively subscribed and its configuration is restored when exiting the game.

#### 2.3.1 Most relevant keyboard functions

proj.c:

- **waiting_ESC_key(Snake* S, Fruit* f)** – subscribes, checks and unsubscribes keyboard interrupts. Calls interrupt handler;

keyboard.c:

- **IH()** – Keyboard Interrupt handler;
- **read_KBC_output** – assemble a scan code;

## 2.4 Mouse

In our project, we implemented the mouse so we can have 2 players playing (it is only available in multiplayer). The goal of the player with the mouse is to spawn the fruits wherever he wants in order to make it harder for the keyboard player to eat the fruits and get more score points.

#### 2.4.1 Most relevant mouse functions

proj.c:

- **void create_MPfrute(int x, int y)** – sets the position for the fruit to be spawned;

mouse.c:

- **void (mouse_ih)()** – mouse interrupt handler;

## 2.5 Video Card

Video card renders all objects and menus.
The game works in VBE mode 0x105 (800x600, indexed, 256 colors). (1) Double buffering, (2) moving objects (collision detection, animated sprites) and (3) fonts are implemented.

#### 2.5.1 Most relevant video card functions

proj.c:

- **createAndDrawScore()** – score positioning and drawing; (2)(3)
- **drawGameSprites()** – draws snake, fruits and playing background; (2)
- **drawMenuSprites()** – draws each menu's item; (3)
- **drawMultiplayerSprites()** – draws snake and playing background; (2)
- **drawInstructionsSprites()** – draws the instructions page; (3)

sprites.c:

- clearScreen() – clears the buffer; (1)

video.c:

- **swap_buffers()** – swaps buffers; (1)

snakeController.c:

- **snakeCollides(Position p)** – checks if the snake collides with any wall; (2)
- **snakeAtesItself(Snake* s) -** checks if the snake collides with itself; (2)

## 3. Code organization/structure

### 3.1 snakeController.c

This module is responsible for controlling the snake. It's where the functions that make the snake move are implemented. It also has the logic to eat fruits, and collision check. – 10% work

### 3.2 sprites.c (combined with head.xpm and fruit.xpm)

This module is responsible for loading/unloading sprites. It's also where the implementation of creating and drawing sprites is defined. All xpm's except for fruit are inside head.xpm and fruit is in fruit.xpm. – 20% work

### 3.3 proj.c

Where everything happens. This module is responsible for managing the game mode based on states. Here is where the interrupts are processed and where data from mouse and keyboard is read. It initializes sprites and draws them (menu, game, score), checks for exiting game and updates snake's and fruit's positions. – 80% work

## 4. Implementation details

This project gave us a better understanding of:

- State machine
- Sprite animation
- Frame generation

## 5. Conclusions

With this project we were able to test our knowledge acquired in lab classes, making it really remarkable.