**Final Report – LC Project**

**Snake Game Invasion**



**Group T02 G08**

André Morais – up202005303

Aníbal Ferreira – up202005429

Bárbara Rodrigues – up202007163

José Albano – up202008561

**Index**

1. User instructions
2. Project status
3. Code organization/structure
4. Implementation details
5. Conclusions
6. **User Instructions**

The project is a single player game inspired by the classic snake game released to Nokia phones in the 90’s. The game differs from the original, we play as a snake that needs to catch apples to grow but is also chased by alien enemies that try to kill it.

The game begins in the main menu where you can use the mouse to over the ‘play’ or ‘exit’ options and click to choose one of the options or you can use the arrow keys and press enter to select those options as well.

Uma imagem com texto

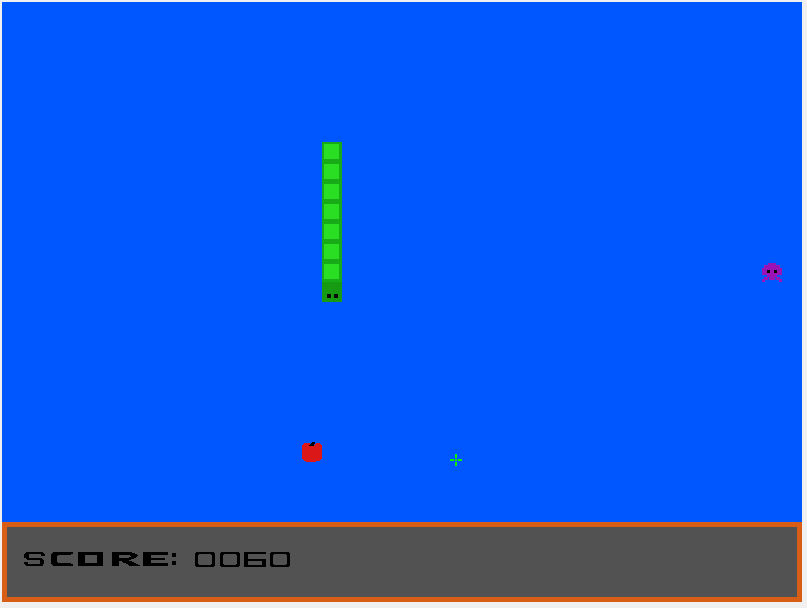
Descrição gerada automaticamente

At any time you can press ESC to leave the game.

Uma imagem com texto

Descrição gerada automaticamente

Inside the game, you can at any time press the ‘P’ key to pause the game and the game will be stopped where it was left. To return to the game you need to press ‘P’ again and the game will continue. In game, if you press the ESC key you leave to the main menu.



You can move the snake using W, A, S, D keys or the up, left, right and down key arrows to move the snake across the screen in those directions.

You can pick up apples to grow and earn 10 of score. You need to avoid the enemies by killing them clicking on them with the left mouse button, each kill grants you 5 of score.

If the enemy hits you, your score will decrease 50 points and your size will also be decreased by one unit. Also, if your score is less than 50 and the enemies hit you, then the game ends and your score will be 0.

If the snakes head hits itself the game ends.

Once you die the game over screen will appear showing you the score at the end of the game and the date and time of the death of the snake.

In the game over screen, you can press ‘Q’ to go back to the main menu.

Uma imagem com texto

Descrição gerada automaticamente

1. **Project status**

# **Implemented features:**

Devices Table

|  |  |  |
| --- | --- | --- |
| **Device** | **Usage** | **Interrupt/polling** |
| Timer | Updates the display and controls the game’s frame rate | Interrupts |
| Keyboard | Game navigation, pause button and player movement | Interrupts |
| Mouse | Menu navigation and player attacks in-game | Interrupts |
| Video Card | Menus and screen display | N/A |
| RTC | Date and time of the end of the game | Polling |

# Timer

Used to update the game screen and the game logic including player and enemy movement at a consistent frame rate.

**Functions:**

* menu\_loop() - Loop that controls the menu display
* game\_loop() - Updates game display and game logic
* gameOver\_loop() - Controls the game over screen display
* pause\_loop() - Pauses the game frame rate
* Functions in timer.c (from the labs)

# Keyboard

Used for player input, to select options in the menus, to exit the game and to control the snake movement in-game. The snake movement is limited so that you can’t go in the opposite direction you are heading to.

**Functions:**

* changeDirection() – receives keys pressed to change snakes movement
* Functions in keyboard.c (from the labs)

# Mouse

Used to control main menu options and to kill enemies during the game. The mouse packets are received, and the mouse position is checked to see if is above an interactable option in the menu or enemy in the game, so that, after the left click is pressed we can know if the click chooses an option of starting or exiting the game, or if kills an enemy in game.

**Functions:**

* updateMouse() - updates mouse packets in menu
* updateGameMouse() - updates mouse packets in game
* checkMouseEnemy() - checks if mouse is in an enemy position
* checkClickEnemy() - checks if mouse is clicked above enemy
* isInOption() - checks if mouse is in an option position
* Functions in mouse.c (from the labs)

# Video Card

Used to draw pixels on the screen and display all the game. Can display the game in the following video modes: 0x105, 0x110, 0x115, 0x11A and 0x14C who can be selected by commenting and uncommenting the code in the proj\_main\_loop().

In mode 0x105 we have a 1024x768 indexed color mode with 8 bits per pixel. In the other modes we have a direct color mode with different bits per pixel and resolutions all compatible thanks to the functions provided in graph.c. The game runs only with a linear framebuffer.

In game, there are moving objects, the snake and the enemies who spawn, one at the time. Also, there are apples that spawn in a random position one after another.

The enemy and the snake can collide with each other, and the snake’s body itself that is composed by the head and by many moving parts that can increase when the snake head contacts with an apple can also collide with the snake’s head. All this collision detection is achieved by storing all the current positions in structs and comparing the x and y coordinates after each frame.

We have fonts in form of pix maps for the mouse crosshair, the menu, the enemies, the apples, the score, and the game over screen that are drawn with the vg\_ultimate\_pixmap\_handler(). The snake is drawn with vg\_draw\_rectangles() from vbe.c from the labs.

The fonts are read from char arrays in sprite.h and they can present slightly different results depending in each graphics mode is selected.

**Functions:**

* vg\_ultimate\_pixmap\_handler() - receives all types of pix maps to draw pixel by pixel
* vg\_ultimate\_pixmap\_eraser() - receives all types of pix maps to erase pixel by pixel
* Functions in vbe.c similar to lab 5

# Real Time Clock (RTC)

Used to get the current date and time of when the snake dies in the game to display in the be displayed in the game over screen.

**Functions:**

* isRTCUpdating() - check if the data from the RTC is in binary-coded-decimal (BCD)
* isBCD() - converts the data in BCD to binary
* BCDtoBin() - reads the current date from the RTC
* getDate() - reads current time from RTC
* getHour() – auxiliar function to get time

1. **Code Organization/Structure**

# **Modules**

# Timer

* Functions from lab 2 to manage game’s frame rate.
* Developed by: All group members

# VBE

* Functions from lab 5 to change video mode and display it and other functions created to handle all the different pix maps used in the project.
* Data Structures used: Enumeration for the pixmaps
* Developed by: All group members

# Keyboard

* Functions from the lab 3 to read KBC scan codes
* Developed by: All group members

# Mouse

* Functions from the lab 4 to read the mouse packets
* Developed by: All group members

# RTC

* Implements the RTC device, allows to get the date and time information from the device and to convert it to values that can be displayed.
* Developed by: Bárbara Rodrigues

# Utils

* Auxiliary functions used in the project and in the labs to get important values such as the MSB and LSB of an address
* Developed by: All group members

# Assist

* Function to subscribe and unsubscribe all devices like the keyboard, timer, and mouse devices. Also functions to draw, verify position, and update mouse status.
* Developed by:

# Game

* Module that manages the game loop. Calls functions to display the game elements and begin the game. Handles the keyboard and mouse interrupts and calls functions to update game logic.
* Data Structures used: struct of mouse packets, struct for the snake, and struct for the enemy
* Developed by: Aníbal Ferreira e José Albano

# GameOver

* Game loop that gets all the information to display the game over menu and displays it. Calls functions to get the RTC data and receives keyboard input to go back to the main menu.
* Developed by:

# Menu

* Draws the menu and changes the menu according to the option selected by the mouse or keyboard. Reads the keyboard scancodes and mouse packets.
* Developed by:

# Pause

* Reads the keyboard scancodes and verifies if the pause key is pressed, if pressed pauses the game loop.
* Developed by: Aníbal Ferreira e José Albano

# Proj

* Subscribes all the devices and controls game logic by changing the game loops.
* Developed by:

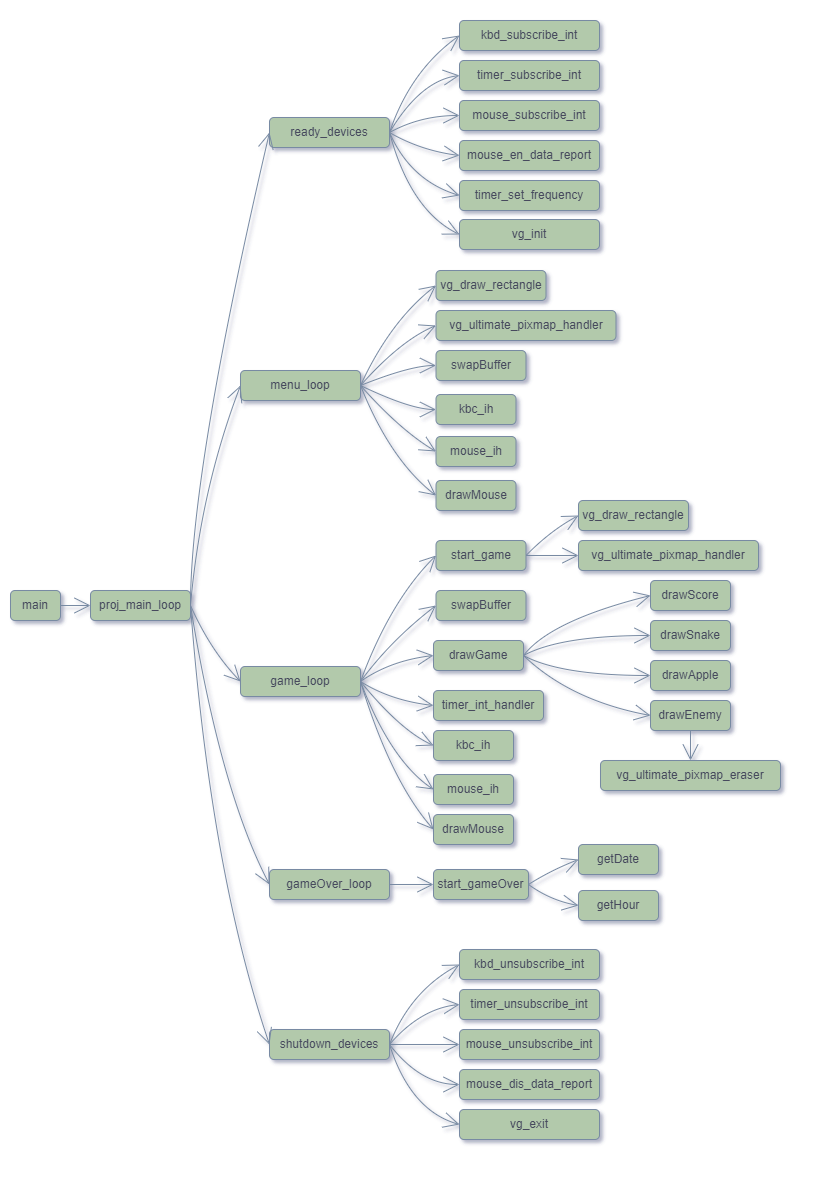
# Snake

* Most important module is responsible for all the game logic. Starts the game, initializes the snake, spawn apples and spawn enemies. Also manages all their interactions with each other. Checks if inputs are valid and collisions. Also updates score based in the events that happen in the game.
* Data Structures used: Struct snake, struct enemy, struct apple
* Developed by: Aníbal Ferreira

# Relative weight for the modules

|  |  |
| --- | --- |
| **Module** | **Relative Weight** |
| Timer |  |
| VBE |  |
| Keyboard |  |
| Mouse |  |
| RTC |  |
| Utils |  |
| Assist |  |
| Game |  |
| GameOver |  |
| Menu |  |
| Pause |  |
| Proj |  |
| Snake |  |

# Function Call Graph



1. **Implementation Details**

# RTC

The RTC needed a few research to put to work but it was simple to implement. (…)

1. **Conclusion**