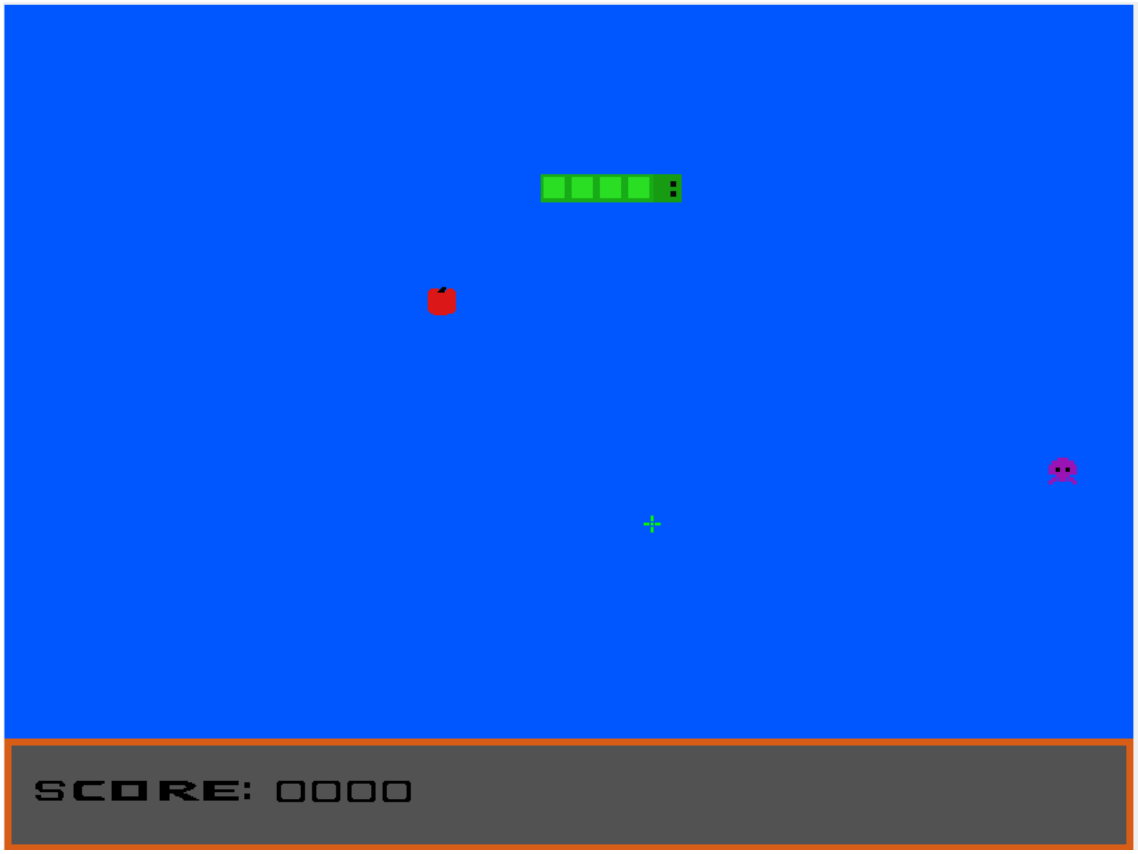
Final Report – LC Project

Snake Game Invasion



Group T02 G08

André Morais – up202005303

Aníbal Ferreira – up202005429

Bárbara Rodrigues – up202007163

José Gaspar – up202008561

Index

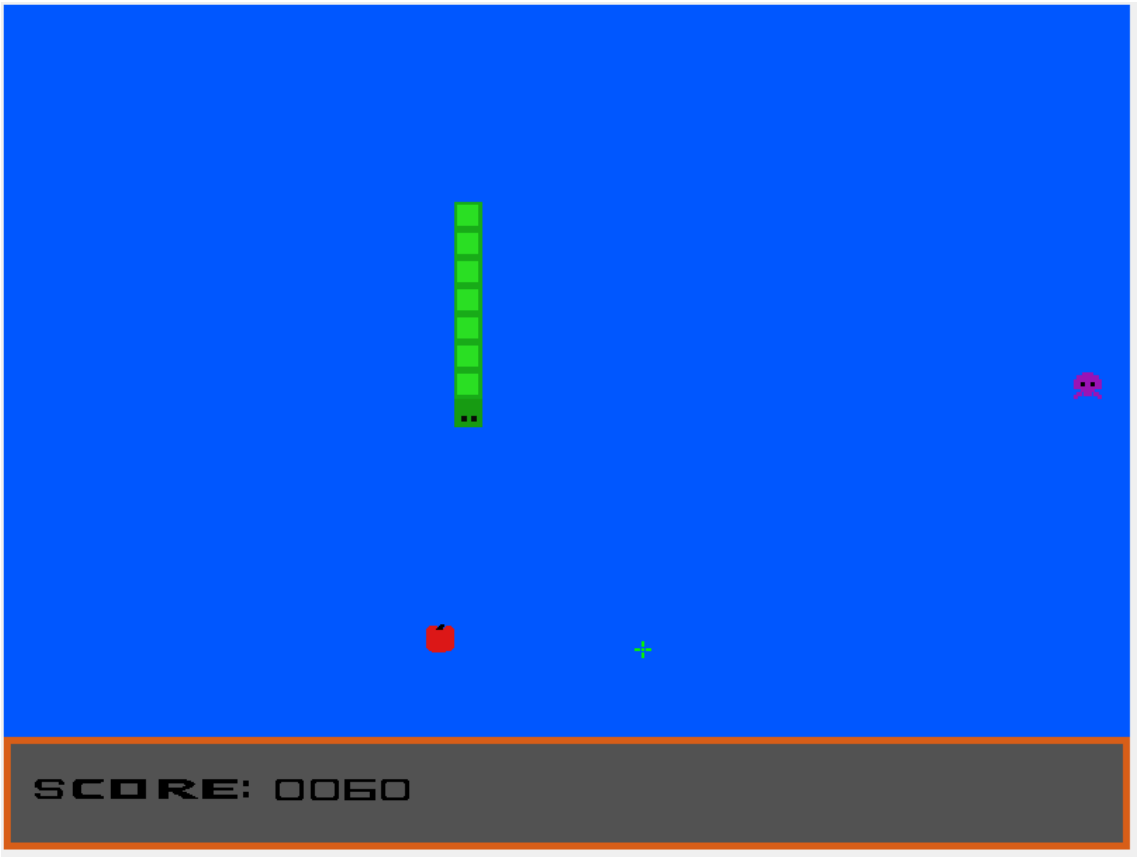
1. User Instructions

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 hover 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.

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

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.

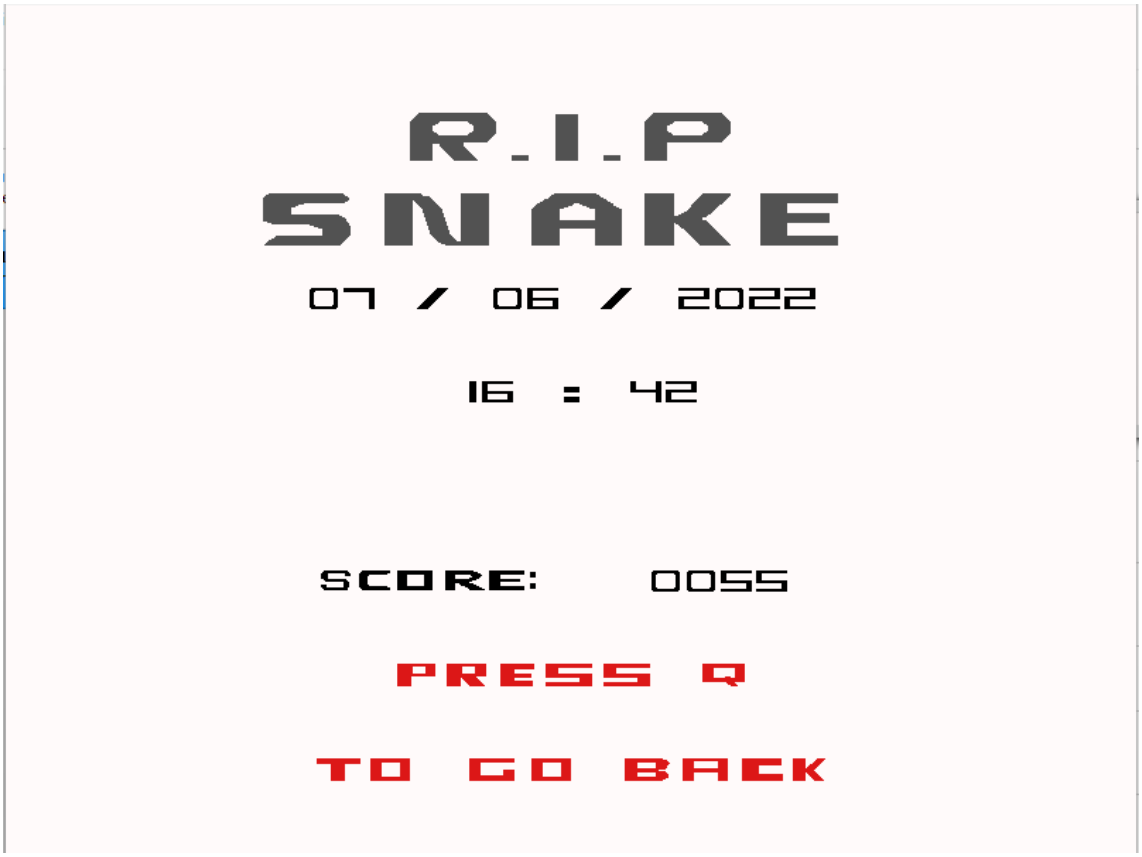
If the enemy hits you, your score will decrease 50 points and your size will be decreased by one unit. Also, if your score is less than 50 and the enemies hit you, then the game ends and your score will be 0.

If the snakes lefts the screen it with will appear in the opposite end of it.

If the snake hits itself the game ends.

Once you die, the game over screen will appear showing you the obtained score. Furthermore, the date and time of the death of the snake is shown as well.

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



2. 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	Gets the current date and time at 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
- 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 the 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 vbe.c. To improve the latency of the update functions the double buffer technique was implemented.

In the 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.

Besides the traditional collisions that the Snake game has (the collision between the snake's head and its own body and the collision between the snake´s head and the apples), the enemy and the snake can also collide with each other. 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 defined drawings for the mouse crosshair, the menu, the enemies, the apples, the score, and the game over screen that are represented as arrays of chars, replacing the conventional pixmaps. This measure was implemented to accomplish a better use of memory once that, to have compatibility with all graphic modes supported by the video card we would have to create different pixmaps, one for each mode. Those arrays are drawn with the vg_ultimate_pixmap_handler(). The snake is drawn with vg_draw_rectangles() from vbe.c from the labs.

The drawings can be found in the file sprite.h.

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 when the snake dies in the game, so it can 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

3. 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: André Morais

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 Gaspar

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: José Gaspar

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: José Gaspar and Aníbal Ferreira

Pause

- Reads the keyboard scancodes and verifies if the pause key is pressed, if pressed pauses the game loop.
- Developed by: José Gaspar

Proj

- Subscribes all the devices and controls game logic by changing the game loops.
- Developed by: All members of the group

Snake

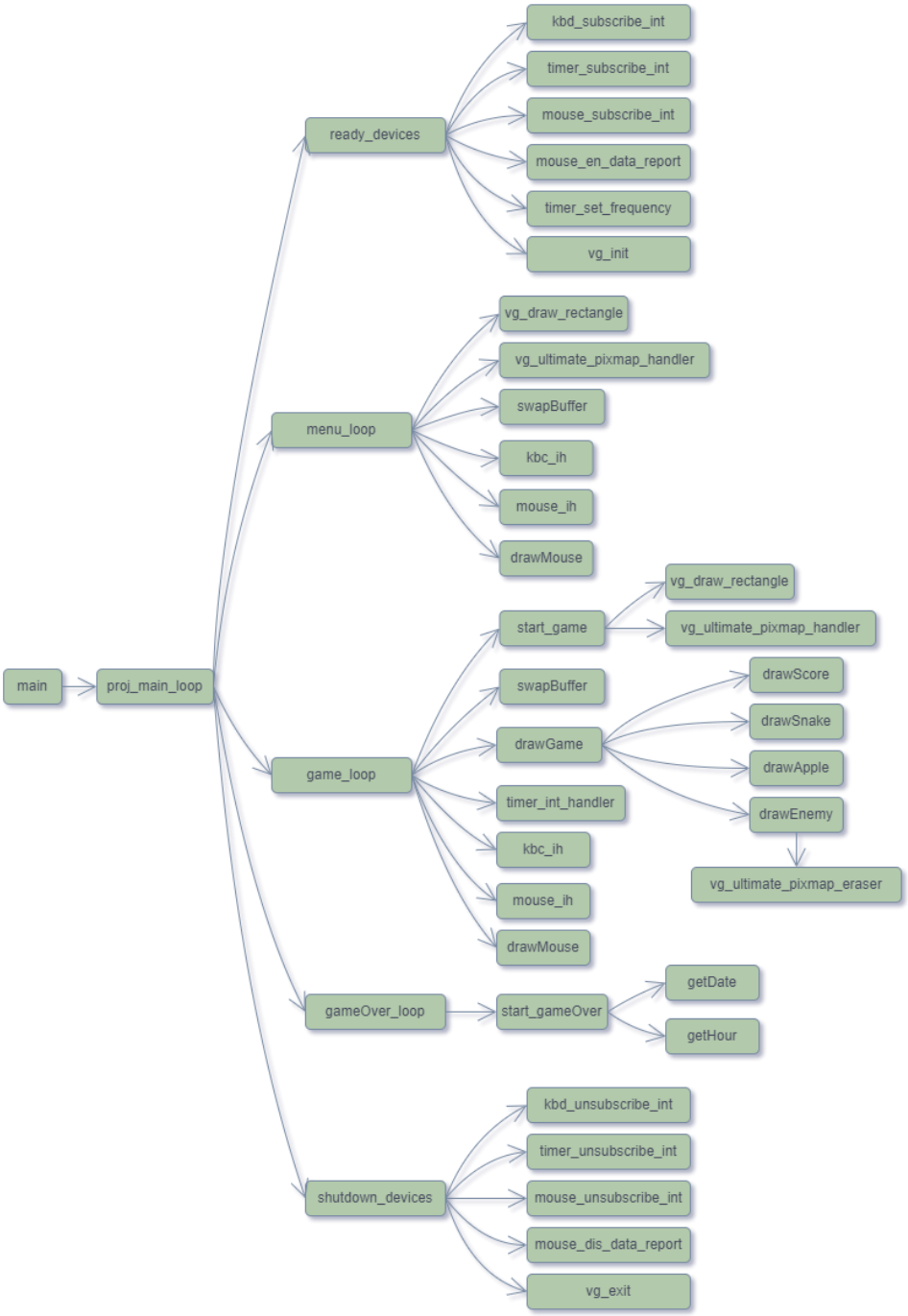
- Most important module is responsible for all the game logic. Starts the game, initializes the snake, spawn apples and spawn enemies. Also manages all their interactions with each other. Checks if inputs are valid and collisions. Also updates score based in the events that happen in the game.
- Data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

- Developed by: Aníbal Ferreira

Relative weight for the modules

Module	Relative Weight
Timer	5%
VBE	20%
Keyboard	5%
Mouse	5%
RTC	5%
Utils	1%
Assist	3%
Game	10%
GameOver	5%
Menu	7%
Pause	2%
Proj	7%
Snake	25%

Function Call Graph



4. Implementation Details

RTC

-The RTC needed a few research to put to work but it was simple to implement and a good feature to the game.

Double buffer

-A very useful feature that helped with the improvement of the game's performance.

Menu

- Receiving both mouse and keyboard input at the same time for the menu selection and detection of the ESC key to quit the game for more input options.

VBE

-The compatibility with all the video modes was simple but took a fair bit of time to implement and provides a lot more video compatibility options to the player.

Game

-The pause key was a great addition to the game that used the timer and keyboard to manipulate the game loop.

-The snake can't move in the opposite direction and can only change direction one frame at the time making it for a nice gameplay experience.

-The size increase with the collision with an apple or decrease with an enemy while keeping record and display of the score was a complicated to implement but it was important to the final game and deserves to be highlighted.

-The mouse input to kill enemies is an innovation to the original game and was well implemented.

-When the snake body lefts an end of the screen with will appear in the opposite end and doesn't disappear of the game.

GameOver

-In the game over screen we tried to use different scancodes and the game receives 'Q' as an input to leave the game.

5. Conclusion

The group thought that everything planned for this project was achieved. If we had more time, we could have improved our game by making it a multiplayer game with the use of the serial port.

The group divided and worked equally for this project and is proud of the different devices used and functionalities implemented.