Faculdade de Engenharia da Universidade do Porto

Snake Game

Computer Laboratory

Class 2LEIC18 Group 4

Duarte Marques (up202204973@up.pt)
Maria João Vieira (up202204802@up.pt)
Marta Cruz (up202205028@up.pt)

Index

1. Goal

1.1 Definition of the main objective

The main objective of this project was to design and implement an interactive Snake Game that runs in the LCOM/Minix environment.
The project had to use at least three hardware I/O devices studied during the semester: the Timer, the Keyboard, and the Graphics Card. The application aims to demonstrate a good understanding of low-level hardware interaction, real-time event handling, and structured programming practices within an operating system environment.

1.2 Description of the application context

The Snake Game is a well-known arcade game where the player controls a snake that grows in length each time it eats a piece of food.
The challenge lies in avoiding collisions with the walls or the snake's own body as it becomes longer. This project implements the game with three difficulty levels, offering increased speed and complexity. The game is played in a graphical interface rendered using the video card, with real-time input captured from the keyboard and time-based events controlled by the timer. This context allows the project to serve both as a game and as a demonstration of efficient use of low-level system programming in a constrained environment.

2. Game

2.1 Overview of the game concept

The game is a classic version of Snake, where the player navigates a snake around a bounded area in order to collect food items. Each time the snake eats a piece of food, it grows longer, and the player gains points. The game ends when the snake collides with the wall or with its own body. The project includes three levels of increasing difficulty, which differ in the speed of the snake and the size of the area. The game is designed to be simple and intuitive, but challenging as the levels progress.

2.2 User interaction and gameplay mechanics

The player interacts with the game using the W, A, S, and D keys to control the direction of the snake:

- W to move up
- A to move left
- S to move down
- D to move right

The snake moves continuously in the current direction, and the player must react in real time to avoid collisions and collect food. The gameplay loop is managed through timer interrupts, which regulate the snake's speed and the refresh rate of the game depending on the selected level. Graphics are rendered directly to the screen using the video card, providing smooth visual feedback for movement, food spawning, score updates, and the game over screen.

Before starting the game, the player is presented with a menu interface where they can select the difficulty level. Additionally, the game includes a help screen accessible from the main menu, which displays all the game instructions and controls, ensuring that the player understands how to interact with the game before playing.

3. Structure

3.1 General organization of the project

The project is organized in a modular and layered structure, with clear separation between core logic, input/output handling, and game-specific functionality. The directory and file layout is as follows:

Root directory

- **Makefile -** Contains build rules to compile the project in the Minix environment.
- **main.c -** Entry point of the application, contains initialization and the main loop.
- **utils.c -** Utility functions, including helpers like util_sys_inb.

src/controllers/

- **video/graphics.{c,h} -** Responsible for initializing the video mode and providing basic drawing functions.
- **keyboard/keyboard.{c,h} -** Handles keyboard input, including PS/2 scan code processing.
- **timer/timer.c -** Configures the Programmable Interval Timer (PIT) and handles timer interrupts.
- **menu/menu.{c,h} -** Manages the game's menu system, including drawing and user input.

src/game/

- **game.{c,h}, digits.{c,h} -** Implements the core Snake game logic and on-screen score display using graphical digits.

3.2 System architecture and components

The system architecture follows a modular event-driven approach, divided into several subsystems:

Initialization & Event Dispatching

- The program starts with lcf_start() and transitions to proj_main_loop(), where graphics are initialized and interrupts are subscribed.
- The LCF (LCOM Framework) is used to facilitate system call tracing (lcf_trace_calls) and logging (lcf_log_output).

Graphics Subsystem

- Initializes VBE graphics mode using BIOS calls via sys_int86.
- Provides drawing primitives such as vg_draw_rectangle and text rendering via utility functions.

Input Subsystem

- Subscribes to keyboard interrupts using kbd_subscribe_int.
- The interrupt service routine kbc_ih() decodes scan codes, which are then dispatched to either the menu system (menu_handle_key) or the game logic (handle_key) depending on the current state.

Timer Subsystem

- Configures the timer using timer_subscribe_int and timer_set_frequency.
- Periodic tick events are used to update the game state via update_game().

Menu Module

- Responsible for the main menu interface, including menu_init, menu_draw, and menu_handle_key.
- Renders a background, title, selectable options (e.g., level selection), and a help screen with gameplay instructions.

Game Module

- Contains core game functions such as game_start, draw_game_static, draw_game_dynamic, and update_game().
- Displays the score using graphical digits from digits.h.

3.3 Supporting architecture diagram

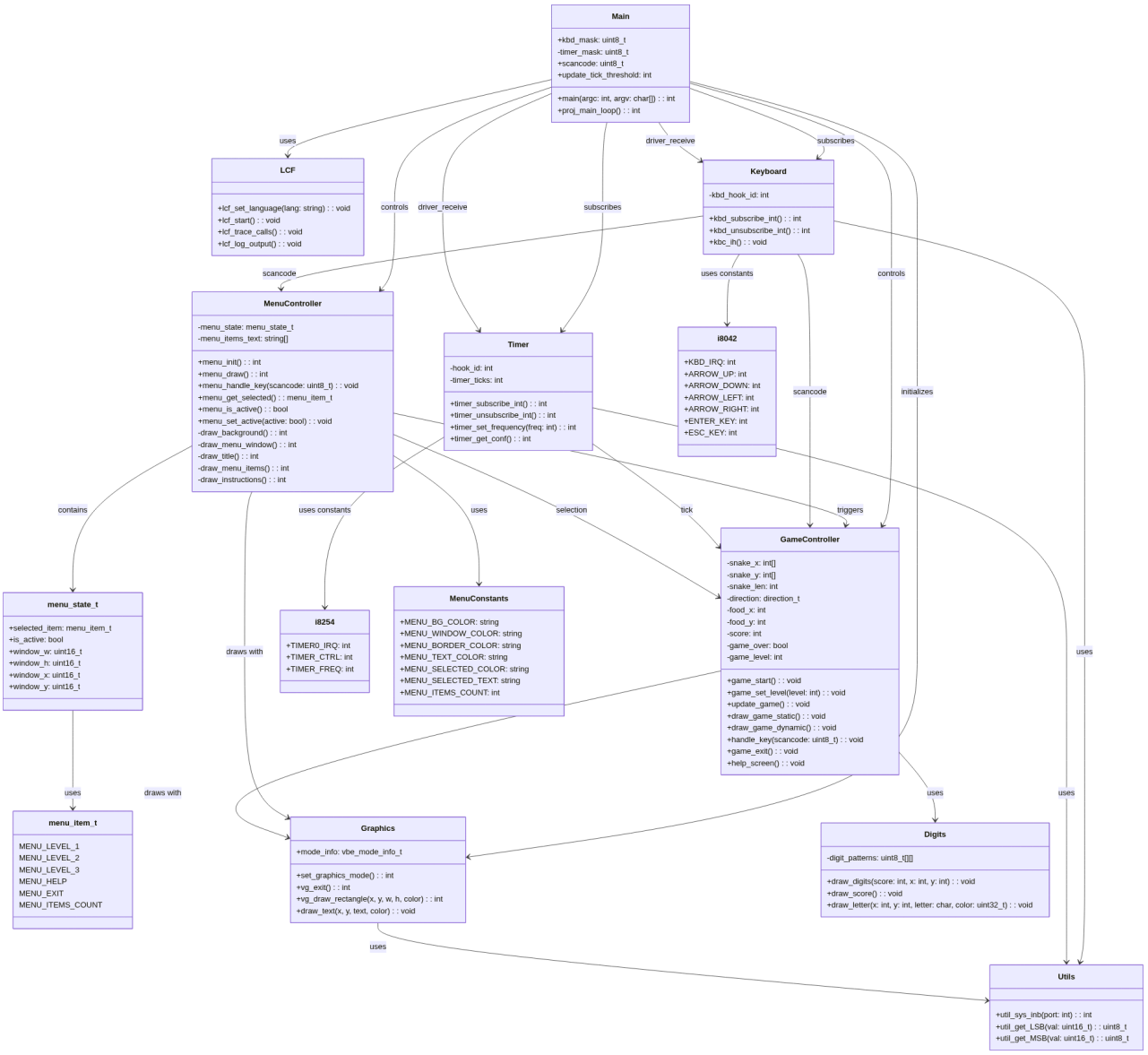

Figure 1: UML class diagram of the snake game architecture.

4. Devices

4.1 Identification of the devices employed

The project utilizes the following hardware components:

- **Keyboard** – PS/2 keyboard controller (i8042)
- **Timer** – System timer for timing control
- **Video/Graphics** – VGA graphics controller for display output

4.2 Functionality and purpose of each device

4.2.1 Keyboard Device

The keyboard device is implemented in the src/controllers/keyboard/ directory, with the main source files being keyboard.c, keyboard.h, and i8042.h, which define the core driver logic and hardware-specific constants. This module is responsible for detecting and handling keyboard input through interrupts, interpreting scancodes into meaningful actions, and managing key press and release events. It provides a critical interface for the user to interact with the system, allowing for menu navigation and in-game controls. The keyboard's functionality is fundamental to user interaction, enabling real-time command input and gameplay control.

4.2.2 Timer Device

Located in the src/controllers/timer/ directory, the timer module plays a vital role in ensuring the correct timing and pacing of the game. It provides precise control over the game loop's timing, helping to regulate the frame rate and support time-based mechanics such as animations and score updates. By generating periodic interrupts, the timer ensures consistent system timing and contributes to the overall fluidity of the game experience. This device is essential for synchronizing events and maintaining performance stability throughout gameplay.

4.2.3 Video/Graphics Device

The video or graphics device, located in the src/controllers/video/ directory, is responsible for initializing and managing the VGA graphics mode used by the game. It enables low-level access to video memory, allowing the drawing of pixels and rendering of shapes, sprites, and user interface components. Through techniques such as double buffering, it ensures smooth visual transitions and reduces flickering during animations. Additionally, it provides support for manipulating color palettes and handling the graphical representation of game elements. This component is fundamental to the player's visual experience, as it renders the menus, gameplay screen, animations, and all visual feedback presented by the system.

4.2.4 Additional Components

Beyond the core hardware devices, several software modules contribute essential functionality to the system. The menu system, implemented in src/controllers/menu/menu.c and menu.h provides an interface for user navigation, enabling the player to start the game, access help, or exit. The game logic, located in src/game/game.c and game.h, encapsulates the core gameplay rules, state transitions, and player interactions, forming the backbone of the interactive experience. Finally, the high score system, found in src/game/highscore.c and highscore.h, handles the tracking and storage of player performance, offering persistence across sessions and enhancing the game's replay value. Together, these components integrate seamlessly with the hardware to create a complete and interactive game system.

5. Differentiating features

5.1 Unique characteristics of the project

This project implements the classic Snake game in graphic mode with direct hardware interaction. It stands out for using keyboard and timer interrupts, an interactive menu with level selection, and direct video memory drawing without external libraries. The modular architecture and clear separation between menu and game logic are also distinguishing aspects.

5.2 Distinctive approaches or innovations

Innovative approaches include the use of interrupts for real-time game control, low-level graphics handling via VBE, and the implementation of difficulty levels. The project also follows a state-driven design, allowing smooth transitions between the menu, game, and help screen.