Tetris

Laboratório de Computadores 2023/2024

Licenciatura em Engenharia Informática e Computação

Turma 17 – Grupo 06

Eduardo Machado, up202105337

Gonçalo Sousa, up202207320

Simão Rodrigues, up202005700

Tiago Pires, up202008790

User Instructions

About

Our game is a classic implementation of Tetris, developed to run on the Minix operating system. The game was created as part of a project to demonstrate understanding and application of low-level programming concepts, including hardware interaction and graphics rendering. The objective of the game is to aligning falling tetromino pieces to form complete rows, which then disappear, making room for more pieces.

How to run it

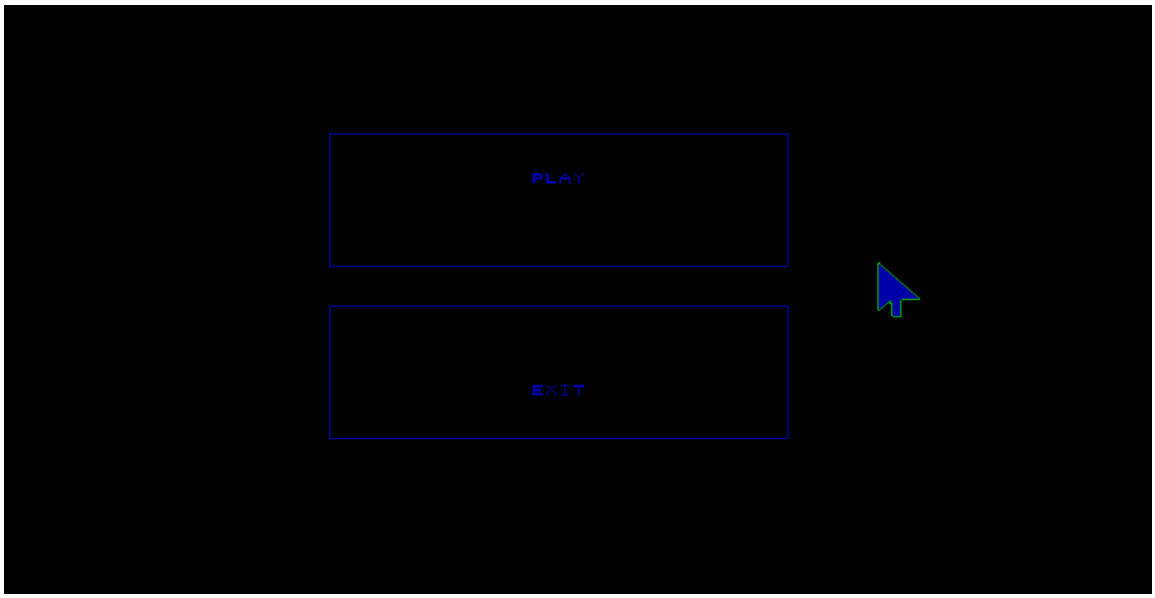
To play our Tetris game, follow these steps:

1. Navigate to the /src directory in your terminal.
2. Compile the game by running the make command.
3. Once compiled, start the game using the command lcom_run proj.

Menu

Upon starting the game, you will be presented with the main menu. The menu consists of the following options:

- PLAY: Start a new game of Tetris.
- EXIT: Exit the game and return to the command line interface.



Gameplay

Once you start the game by selecting "PLAY" from the main menu, you will enter the gameplay screen. The game begins with a random Tetris piece appearing at the top of the game board. The objective is to manipulate these pieces to form complete horizontal lines, which will then be cleared from the board. The controls for the game are as follows:

A: Move the piece left.

D: Move the piece right.

S: Move the piece down faster.

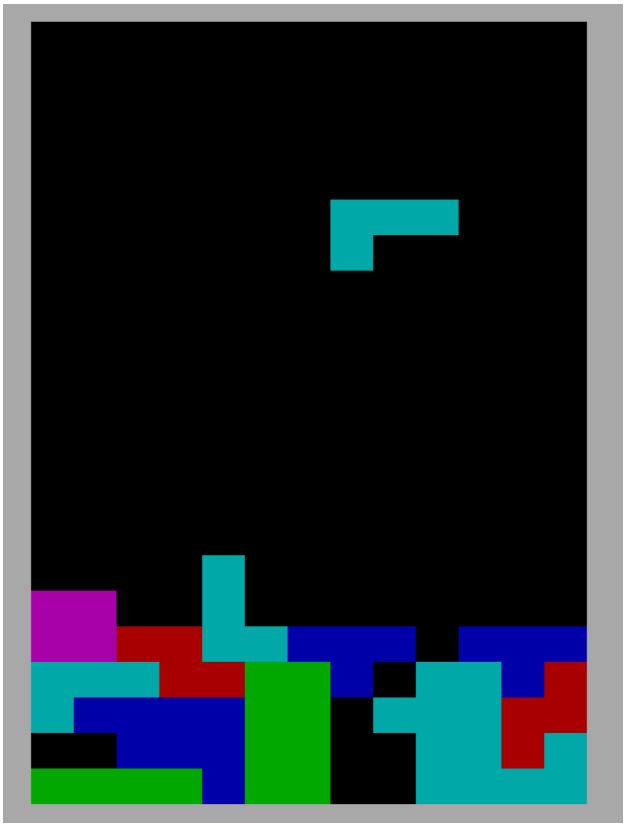
W: Rotate the piece.

SPACE: Drop the piece to the bottom.

Q: End the game.

ESC: Return to the terminal.

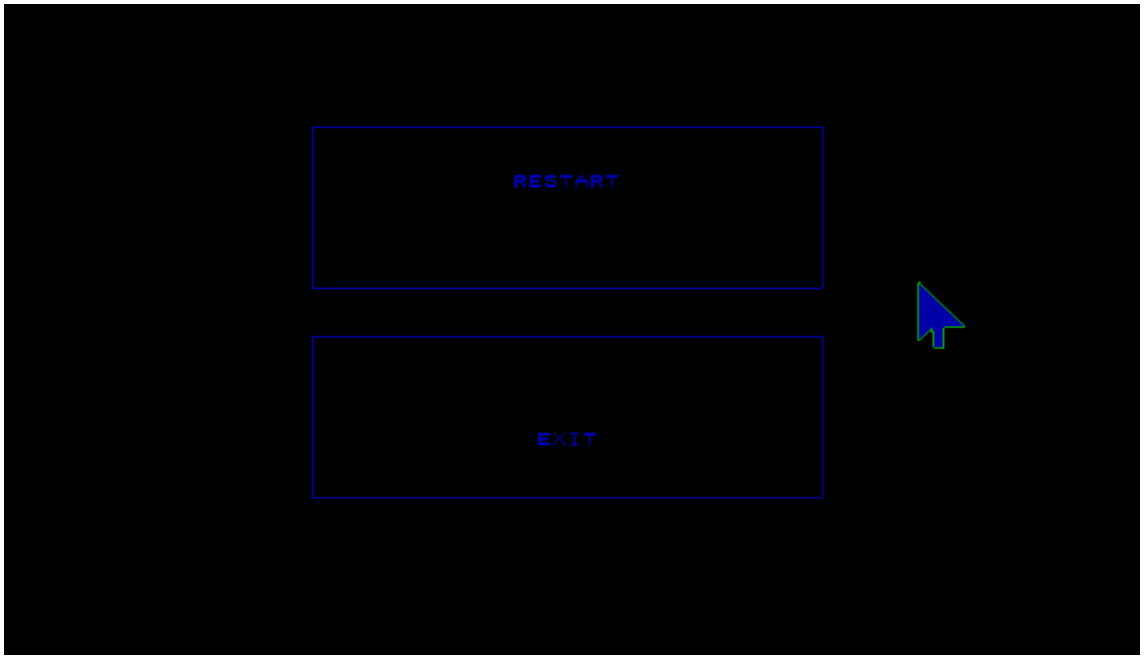
The game continues until the pieces stack up to the top of the screen, at which point the game is over. If the player manages to place 100 pieces on the screen, they win the game.



Game Over

When the game is over, either by reaching the top of the screen with stacked pieces, you will be taken to the game over screen. The game over menu provides the following options:

- RESTART: Start a new game with a fresh game board and a new sequence of pieces.
- EXIT: Exit the game and return to the command line interface.



Project status

Device	What for	Interrupt
Timer	Control of the frame rate	Y
Keyboard	Piece movement and rotation	Y
Mouse	Menu navigation	Y
Graphic Card	Rendering graphics and game elements	N

Timer

Refreshing the game and menu at a constant frame rate, ensuring smooth gameplay and consistent timing for piece movements, with the piece moving down every 60 ticks.

Code References:

timer_int_handler() – timer.c: Increments the counter to keep track of time.

proj_main_loop() – main.c: Uses the timer interrupt to update the game state and handle piece movements.

Keyboard

Controlling the movement (left, right, down) and rotation of the Tetris pieces, dropping the piece to the bottom immediately and exiting the game.

Code References:

kbc_ih_keyboard() – keyboard.c: Interrupt handler for reading scancodes from the keyboard.

escape_key() – keyboard.c: Detects if the ESC key is pressed to exit the game.

move_piece() – game.c: Moves the current piece based on keyboard input.

rotate_piece() – game.c: Rotates the current piece when the appropriate key is pressed.

Mouse

Navigating the game menus (Main Menu, Game Over screen) and selecting options like PLAY, RESTART and EXIT.

Code References:

mouse_ih() – mouse.c: Interrupt handler for processing mouse packets.

mouse_event_handler() – mouse.c: Updates the mouse position and checks for button presses.

proj_main_loop() – main.c: Handles mouse clicks for menu navigation.

Graphic Card

Rendering all game graphics including the game board, Tetris pieces, and menu interfaces.

Details:

Video Mode: 0x105, 1024x768 resolution with a 256-color palette.

Uses double buffering.

Code Organization

utils.c

The utils.c module include functions to get the least significant byte (LSB) and most significant byte (MSB) from a 16-bit value, and to read values from I/O ports. These utilities make it easier to interact with the hardware.

Weight of module: 5%

Main Data Structures: No complex data structures, just utility functions.

timer.c

The timer.c module manages the system timer. It includes functions to set the timer frequency, handle timer interrupts, and manage the timer's state.

Weight of module: 10%

Main Data Structures: Variables like hook_id and counter for timer management.

keyboard.c

The keyboard.c module handles keyboard input. It manages subscribing and unsubscribing from keyboard interrupts and processes key presses. This allows players to control the Tetris pieces using the keyboard.

Weight of module: 10%

Main Data Structures: Variables like keyboard_hook_id and scancode for keyboard handling.

mouse.c

The mouse.c module deals with mouse input. It includes functions to handle mouse interrupts, read mouse data, and process mouse events. This enables players to navigate the game menus using the mouse.

Weight of module: 10%

Main Data Structures: Variables like mouse_hook_id, count, pp (packet structure), and cursor positions.

video.c

The video.c module is responsible for drawing everything on the screen. It sets up the video mode, draws shapes and images, and manages the frame buffer to ensure smooth graphics. This module is crucial for displaying the game board, pieces, and menus.

Weight of module: 20%

Main Data Structures: Pointers to video memory (video_mem, double_buffer) and mode information (mode_info).

main.c

The main.c module is the heart of the game. It contains the main game loop, initializes the hardware, manages game states, and handles input from the timer, keyboard, and mouse. It ties everything together to run the game.

Weight of module: 20%

Main Data Structures: GameState enum for different game states and Arrays to manage Tetris pieces.

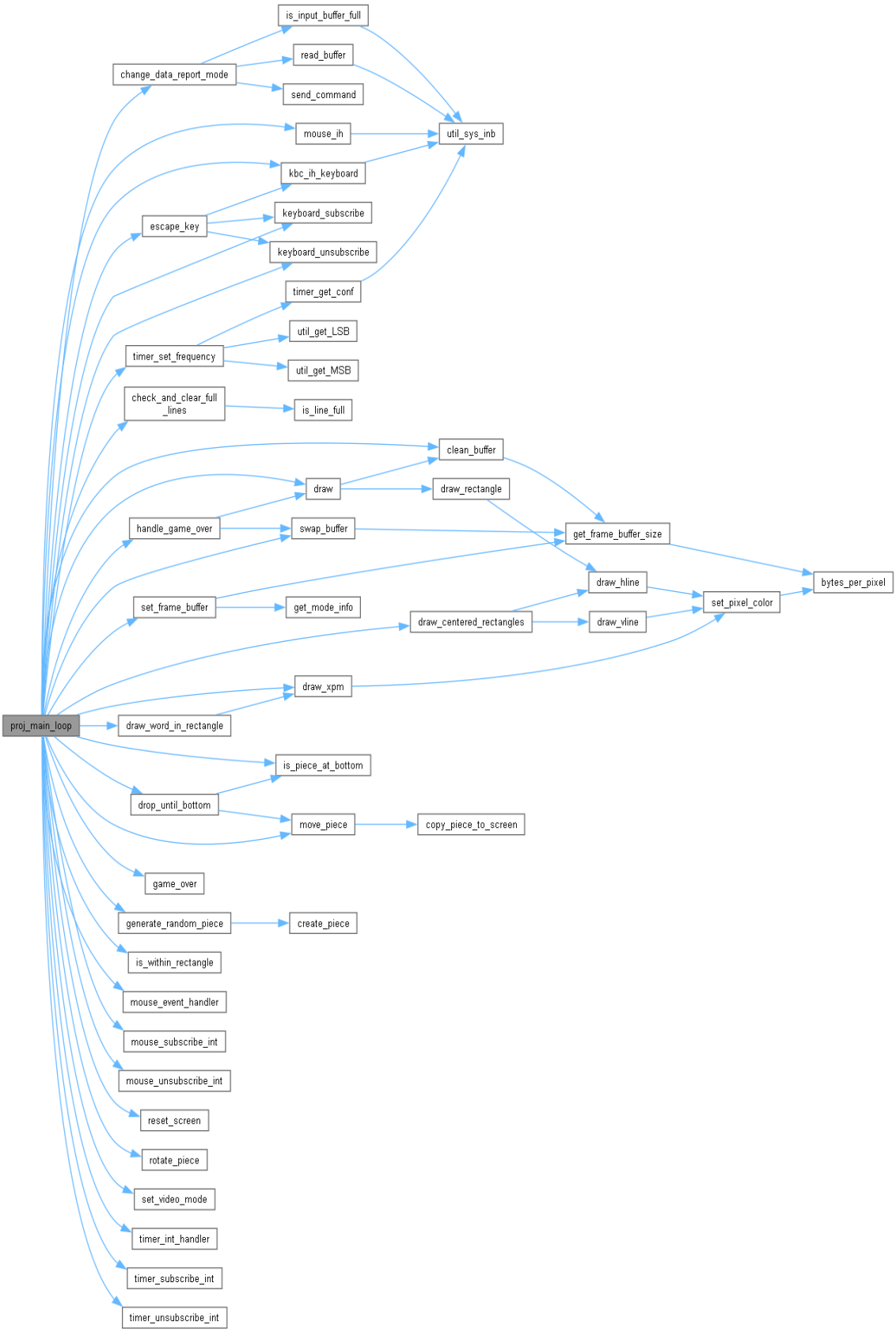
game.c

The game.c module contains the core game logic. It includes functions to create and manage Tetris pieces, check for collisions, clear lines, and update the game state. It handles how pieces move, rotate, and settle on the game board.

Weight of module: 25%

Main Data Structures: TetrisPiece struct to represent pieces and Arrays screen and colorScreen to represent the game board.

Function Call Graph



Implementation Details

Topics covered in the lectures:

State machines.

Topics not covered in the lectures:

Collisions.

Conclusions

Problems

- The game sometimes the piece moves down one step more that supposed before detecting the game over.

Like to haves

- Implementing RTC (Real-Time Clock) to display the current time.
- Introducing different difficulty levels.
- Leaderboard to compare scores with other players.

Main achievements

- Successfully implemented a classic version of Tetris on the Minix operating system.
- Integrated keyboard and mouse input for an enhanced user experience.

Lessons learned

- Improved skills in handling user inputs and rendering graphics.