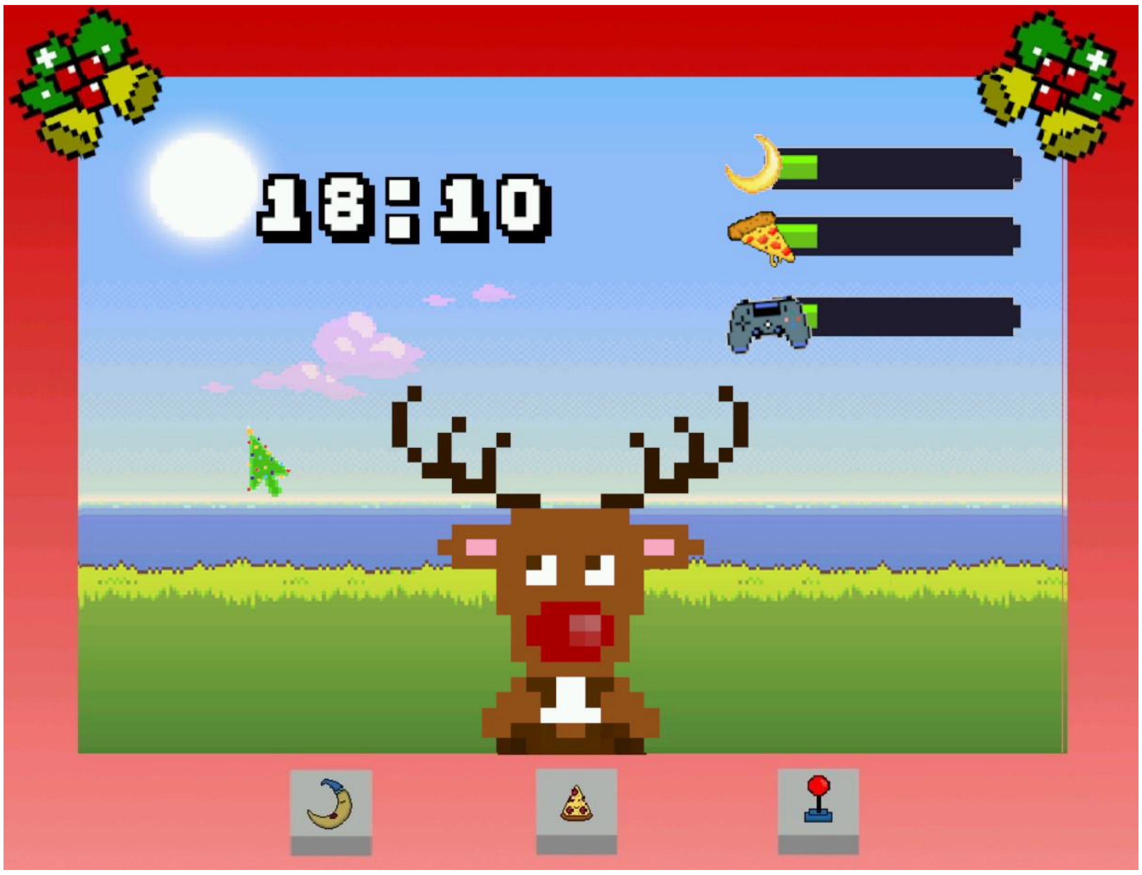
U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Christmas Tamagotchi



Project Report

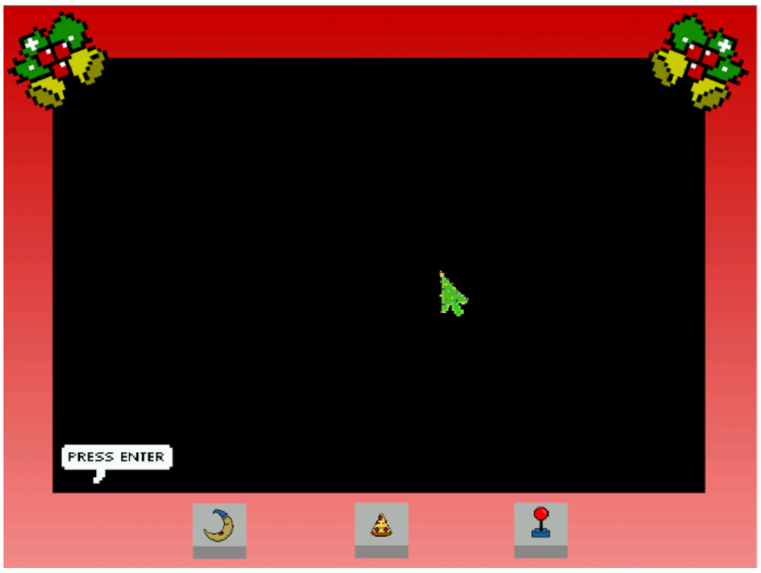
Adriana Cruz e Silva da Costa Gonçalves - up201808911

Beatriz Costa Silva Mendes - up201806551

Index

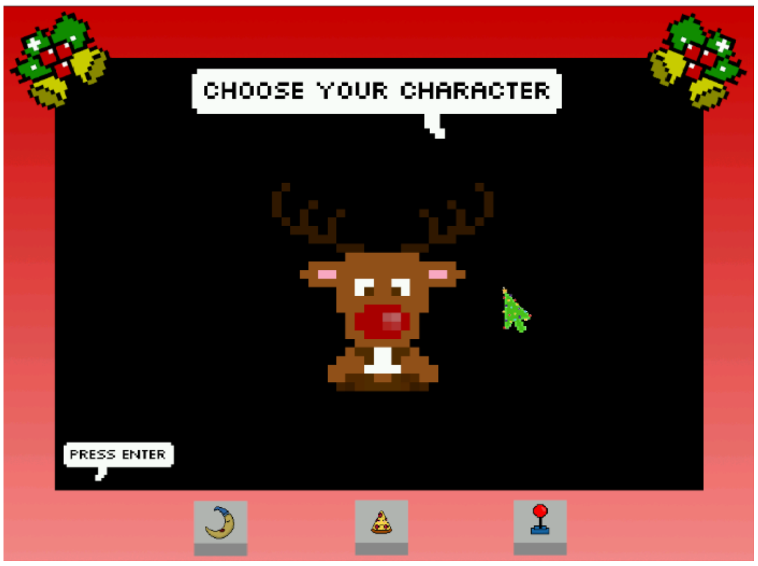
Press Enter menu

The first thing you see upon entering the game is a menu that says "Press Enter", which starts the game.



Choose your character menu

Next you're taken to a "Choose your character" menu in which you can use the mouse to pick which virtual pet you want to be your Tamagotchi. In this case you only have one character, Rudolph.



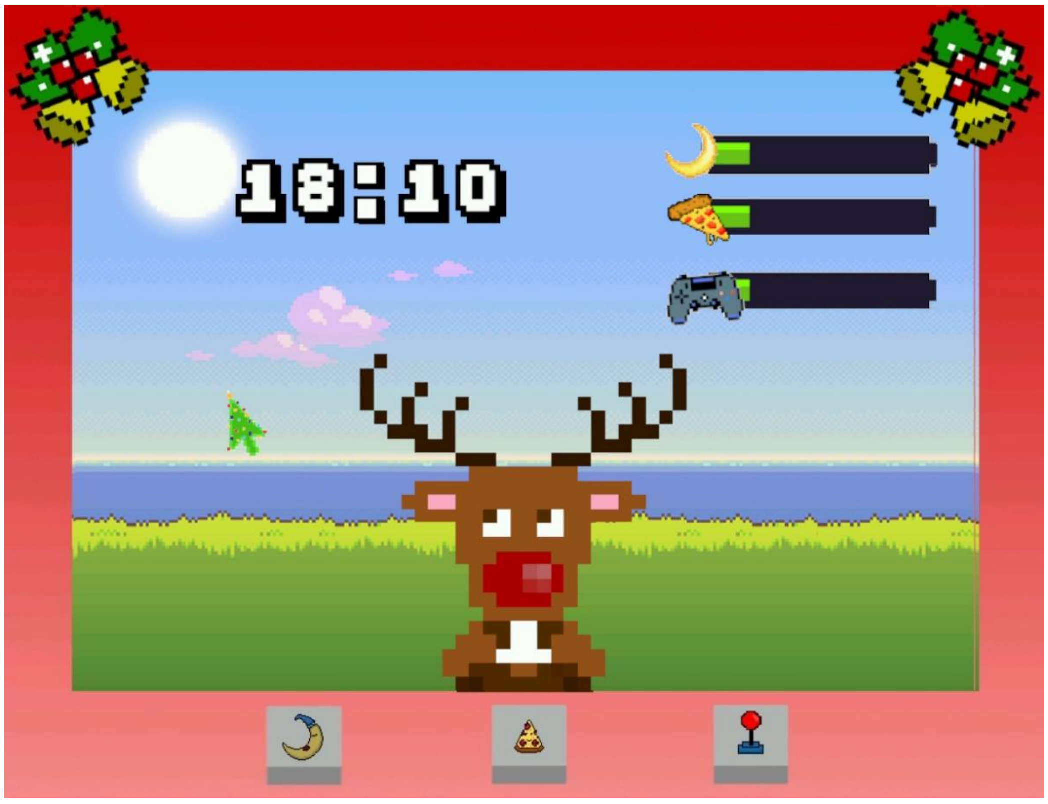
Main menu

After choosing your pet you're taken to the game's main menu, in which you can interact with your virtual pet in the following ways:

- Playing a game
- Feeding your pet
- Make your pet have some sleep

You can tell if your pet needs to do any of these things by looking at the bars on the top right corner. The level on these bars will go down with time, and you need to interact with your pet to make the bars go up again.

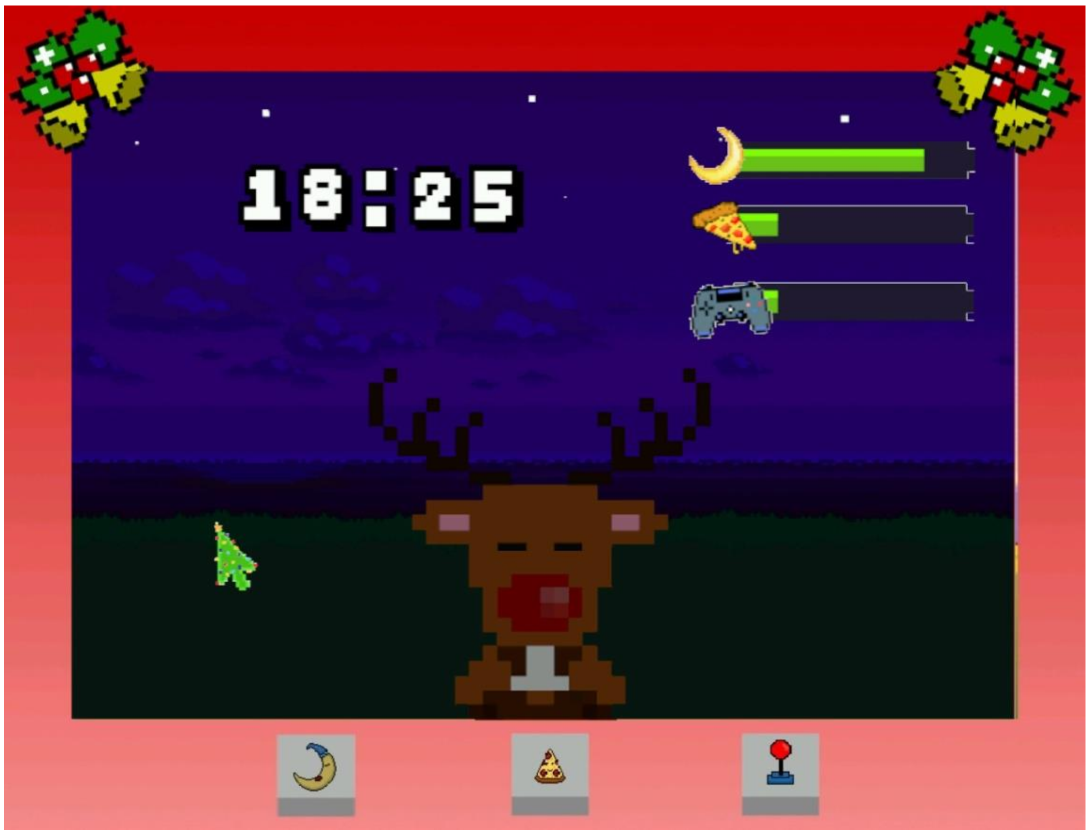
You can also interact with your pet by moving the mouse around the screen so your pet looks at it.



Sleep

To make sure your pet gets some rest, all you need to do is click on the sleep button – the button with the moon logo – and wait until the sleep bar is full.

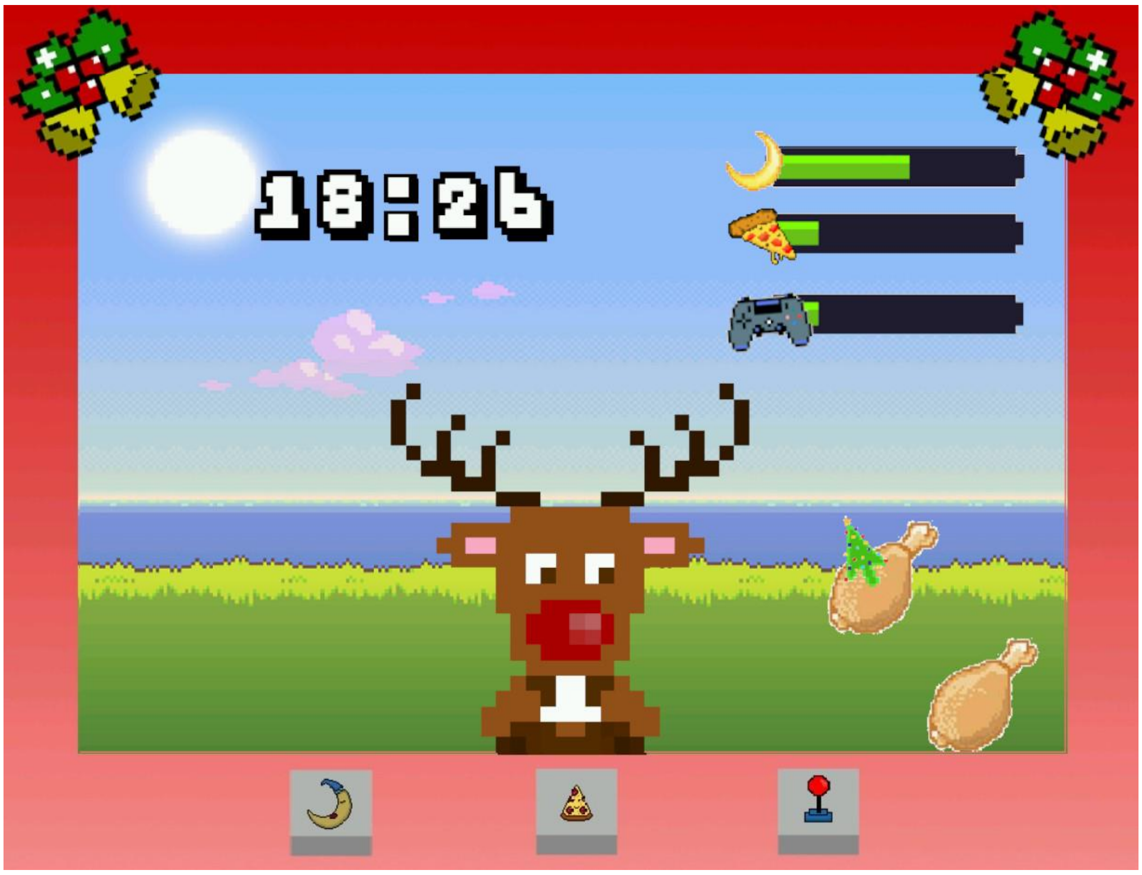
To wake your pet, press the backspace key.



Food

To feed your pet, just click on the food button below. When you click on it, a piece of food will appear on the screen. You then need to drag the food using the mouse and release it on your pet. Every time you do this the food bar above will go up a certain amount. To fill it completely, just feed your pet as many pieces of food as it needs.

To make the piece of food disappear from the screen, press the backspace key.

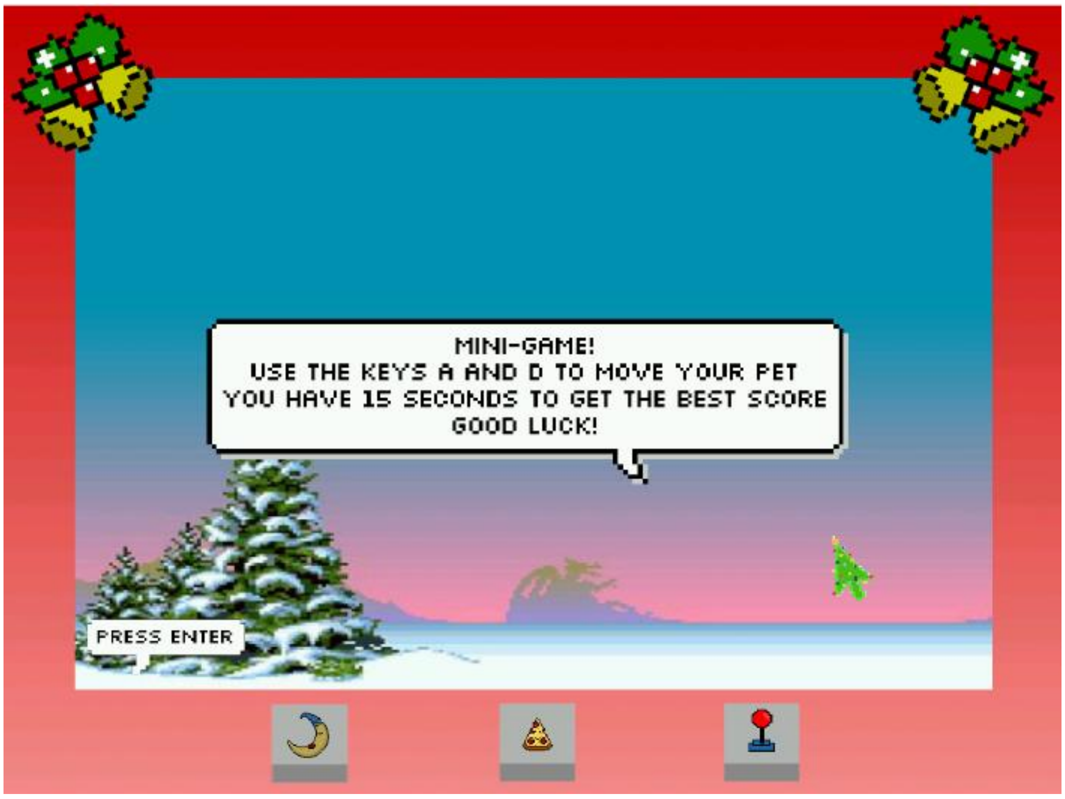


Game

If your pet needs to have some fun and play, click on the button with the controller below.

This will open the **Minigame menu.**

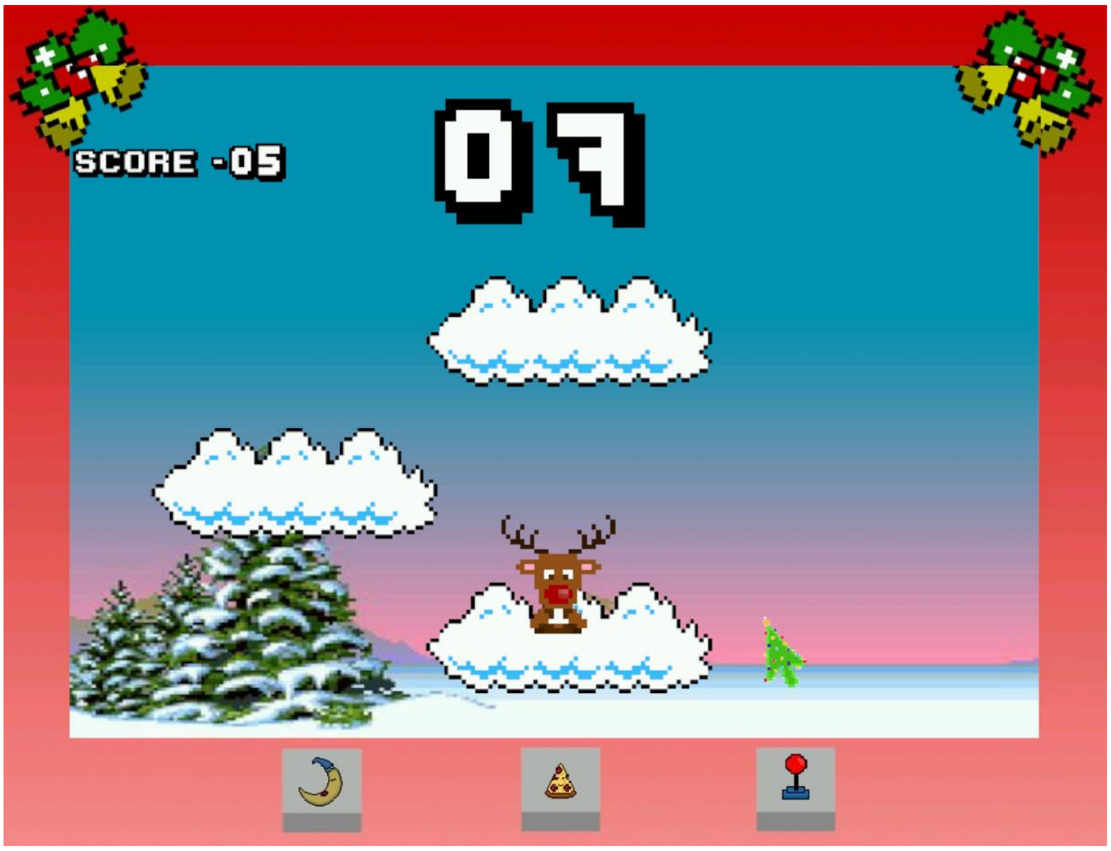
In this menu you have instructions as to how to play the game. Once you understand these instructions you just need to press enter to start the game.



MINI-GAME!
USE THE KEYS A AND D TO MOVE YOUR PET
YOU HAVE 15 SECONDS TO GET THE BEST SCORE
GOOD LUCK!

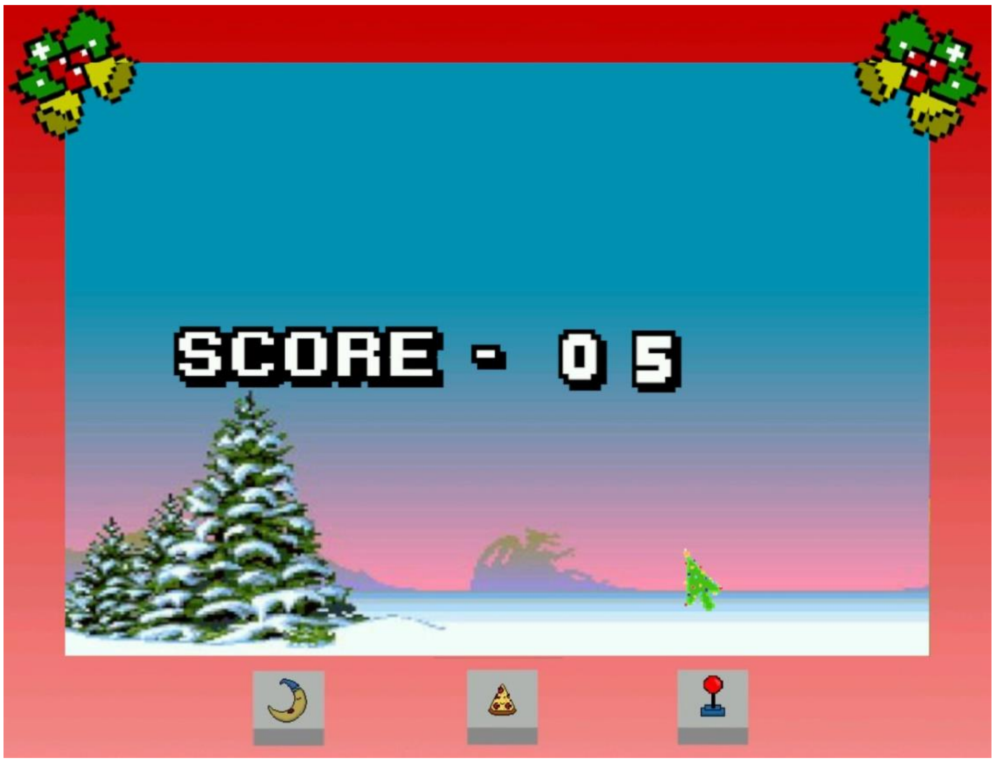
The game consists of a platformer in which you need to jump to as many clouds as possible before the time runs out. To jump you only need to use the A or D key. If the next cloud platform is at the right, press D. If you press A and the next cloud was at the right, you die.

While you're playing you can see the counter counting down the time you have left to play, and also your current score.



When you die or when the time runs out, you'll be taken to the **end of game menu.** This menu will tell you how much you scored.

To try again you just need to press Enter.



If you don't want to play again and want to go back to the main menu, press the backspace key.

Project Status

Devices Used

Devices	Use	Interrupt
Timer	Count-down in minigame All xpms are drawn at each timer interruption	Yes
Keyboard	Interface with main menus and minigame	Yes
Mouse	Interface with main menus	Yes
Video Card	Drawing all images presented	No
RTC	Displaying real time	Yes

Timer

Used to:

Count the time the pet has to jump on the platform in the mini game i.e. the time until it reaches zero.

Count the time passing to decrease the sleep, hunger and game bars.

Used to draw each xpm on screen i.e. all xpms are drawn at each timer interrupt.

Functions that use the timer:

All timer.c functions and timer_manager() in main_functions.c

Keyboard

The keyboard is used for game/application control.

Used to:

Allow your pet to jump left or right during the minigame using the A and D keys.

Press Enter to start Tamagotchi, to start minigame and to play again when you lose the minigame.

Press the Backspace key to return to previous menu.

Press Esc key to exit game.

Functions that use the keyboard:

All keyboard.c functions and keyboard_manager() in main_functions.c

Mouse

We use both the mouse’s position and the buttons.

Used to:

Click on a character to choose it (uses both position and buttons).

Click on the Tamagotchi’s buttons to make your pet sleep, feed your pet or play the mini game (uses both position and buttons).

Drag and drop the food on the screen onto your pet to feed it (uses both position and buttons).

Move the mouse around the screen so your pet looks at it (uses position).

Functions that use the keyboard:

All mouse.c functions and mouse_manager() in main_functions.c

Video Card

We are using:

XPM 5:6:5.

Video Mode 0x117 with a resolution of 1024 x 768 pixels.

Double buffering, used to make the game more fluid.

Sprite animation (food movement when you're feeding your pet).

Collision detection (feeding your pet only when you release the food on top of it).

Fonts to display the real time clock, the in game counter, the in game score and the final score.

Used to:

All the graphics of the Tamagotchi were done using the video card.

Functions that use the video card: all video_card.c functions and xpm_load() (given by the teachers).

RTC

Used to:

Used to read the time and display it on the Tamagotchi's screen.

Functions that use the RTC: all rtc.c functions and rtc_manager() in main_functions.c

Code Organization/Structure

Loading xpms

In this module we load all xpms used in the project, and we use various functions to decide which xpm should be loaded depending on other variables. This is one of the main modules of the project given that it decides what we see on the screen.

This module contains functions to:

Decide which bar to use (in the play, sleep bar and food bar)

Provides the correct xpm for Rudolph to change its eyes according to the mouse's position

Provides the correct xpms for the score shown during the minigame and when the minigame is over

Provides de correct xpms for the hours and minutes shown on the screen.

Module developed by: Adriana Gonçalves and Beatriz Mendes.

Main Functions

Along with loading xpms, this is one of the main modules in the project. In this module we have the timer_manager, the keyboard_manager and the mouse_manager which are the main functions that decide the game's logic. This modules also contains the main variables used in the game.

The timer_manager decides what to show on screen depending on certain variables, that can be altered by the mouse_manager or the keyboard_manager by mouse movement/clicks and pressing certain keys on the keyboard.

For example, when you click on the button to make your pet sleep, the mouse_manager will make the sleep variable true. The timer_manager will then analyze if the sleep variable is true and then display the xpms associated with the sleep menu.

The mouse_manager interprets the mouse's position, which is used to check where on screen you are clicking, and also to display the cursor.

Also in this module we have all the logic for the minigame. The minigame uses the keyboard_manager to check if the user is jumping to the right or left (meaning, pressing the D or A key) and if the user jumped to the correct side, which means it's the keyboard_manager that decides if the user lost or not. The counter and scores shown are, like before, displayed by the timer_manager. However, the clouds are displayed by another function called draw_clouds, which also counts the number of jumps.

Module developed by: Adriana Gonçalves and Beatriz Mendes.

Macros

This module contains all the macros used in the game to represent positions. These macros are mainly used in the Main Functions module to display the xpms in the correct position on screen.

In this module we also have the RTC Macros, used in the rtc.c functions.

Module developed by: Adriana Gonçalves and Beatriz Mendes

RTC

This module contains all the necessary function to access and read the value of the real time clock. In this module we update the value of the rtc_time struct with the real time hours and minutes, that are later used in the Main Functions module.

Module developed by: Adriana Gonçalves.

Types

This module contains a few enums that represent the state of the bar (play bar, sleep bar and food bar), the counter and the cloud position in the minigame that help with the games logic in the Main Functions module.

Module developed by: Adriana Gonçalves and Beatriz Mendes

Myutils

This module contains two functions, max() and min(), that return the maximum and minimum of 2 numbers. These functions are used in the mouse_manager.

Module developed by: Beatriz Mendes.

Timer

Code imported from the lab classes.

Mouse

Code imported from the lab classes.

Keyboard

Code imported from the lab classes.

Video Card

Code imported from the lab classes.

i8254

Macros for the timer, imported from the lab classes.

Mouse_macros

Macros for the mouse, imported from the lab classes.

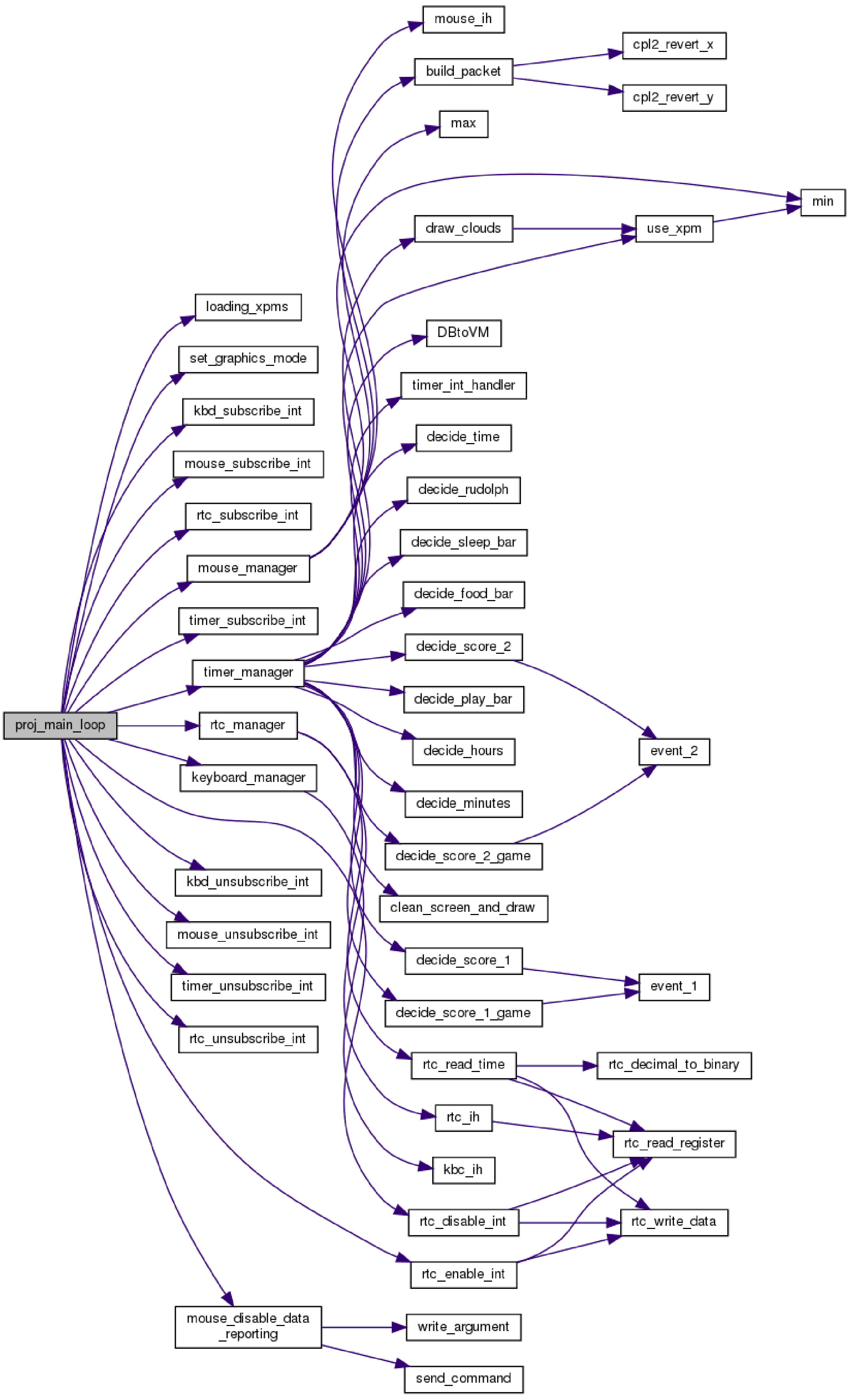
Keyboard_macros

Macros for the keyboard, imported from the lab classes.

Videocard_macros

Macros for the video card, imported from the lab classes.

Module	Relative weight of the module (%)
Loading xpms	25
Main Functions	30
Macros	2
RTC	6
Types	4
Myutils	1
Timer	6
Mouse	6
Keyboard	6
Video Card	6
i8254	2
Mouse_macros	2
Keyboard_macros	2
Videocard_macros	2


mouse_ih
build_packet
cpl2_revert_x
cpl2_revert_y
max
min
draw_clouds
use_xpm
loading_xpms
DBtoVM
set_graphics_mode
timer_int_handler
kbd_subscribe_int
decide_time
mouse_subscribe_int
decide_rudolph
rtc_subscribe_int
decide_sleep_bar
mouse_manager
decide_food_bar
timer_subscribe_int
decide_score_2
timer_manager
decide_play_bar
proj_main_loop
rtc_manager
decide_hours
event_2
keyboard_manager
decide_minutes
decide_score_2_game
kbd_unsubscribe_int
clean_screen_and_draw
mouse_unsubscribe_int
decide_score_1
timer_unsubscribe_int
event_1
decide_score_1_game
rtc_unsubscribe_int
rtc_read_time
rtc_decimal_to_binary
rtc_ih
rtc_read_register
kbc_ih
rtc_disable_int
rtc_write_data
rtc_enable_int
mouse_disable_data_reporting
write_argument
send_command

Implementation Details

As recommended we layered the code from the most basics functions that access register and manipulate that information to the more complex functions that represent the structure of our project, which prevented the code from becoming way too difficult and disorganized.

The xpms we used are loaded from the start, but appear only on screen when certain variables are true, which allowed us to create multiple menus and navigate between them with ease just by changing variables between true and false.

We implemented two state machines: one to make the level on the bars change when time passes or when certain actions are performed (for example feeding or sleeping).

The other state machine works for the minigame to check if the user has pressed the correct key and, if so, draw more clouds, creating the feeling of movement inside of the game.

We also implemented collision detection to make sure that you can only feed your pet when you drag the food onto it, and that it doesn't work when you drag it anywhere else on the screen.

To display the real time on the screen we decided to add every xpm that represents the minutes and hours into an array in a way that the xpm correspondent to a certain hour/minute is in the position of index equal to that hour/minute. This facilitates the code so that we don't need to check what's the value of the hours or minutes case by case and we can return the correspondent xpm right away.

We also feel like we should notice that the project available on Redmine doesn't have an incremental implementation due to problems we had upon committing our final version (many files would disappear, and when we would add them back, other different files would then disappear too) which meant we had to create a new folder for the project to be able to submit the final version.

Also during the making of the project we would sometimes have a server error when accessing Redmine, which the teachers couldn't resolve. Because of that we also worked a lot with Github, meaning not all commits can be seen on Redmine.

Conclusions

We both believe we gained a good understanding of the subjects we've worked with in the course of the unit.

Keeping up with the workload during the semester can be challenging, and we believe the workload can be excessive at times, especially given the other units we have during the semester.

Also for the labs we believe that at times the handouts can be more complicated than necessary, and that it would help to have it more summarized, especially given the time we have available not always being enough.

This final project was the most challenging part, and we both spent several days working on it to be able to meet the goals we set in the Project's Specification. We feel like those goals were met and we're very proud of our final project and the work put into it.