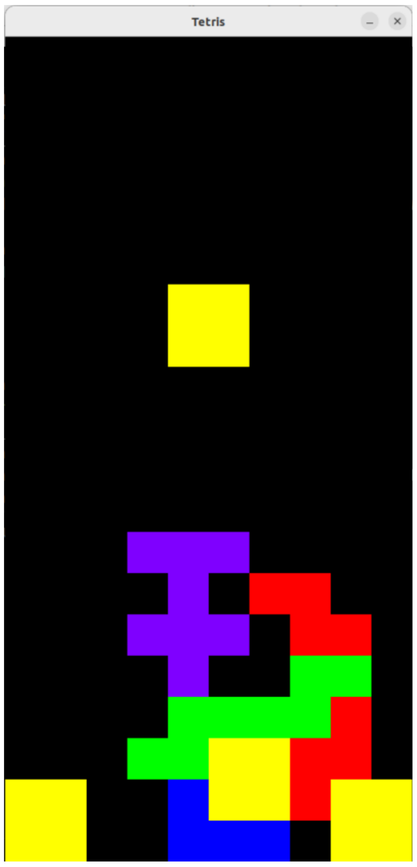
May 2023 - ENSTA Paris

Project Report

2D Tetris Game

IN104



Authors :

Edouard CLOCHERET

and

Louis PAILLONCY

Contents

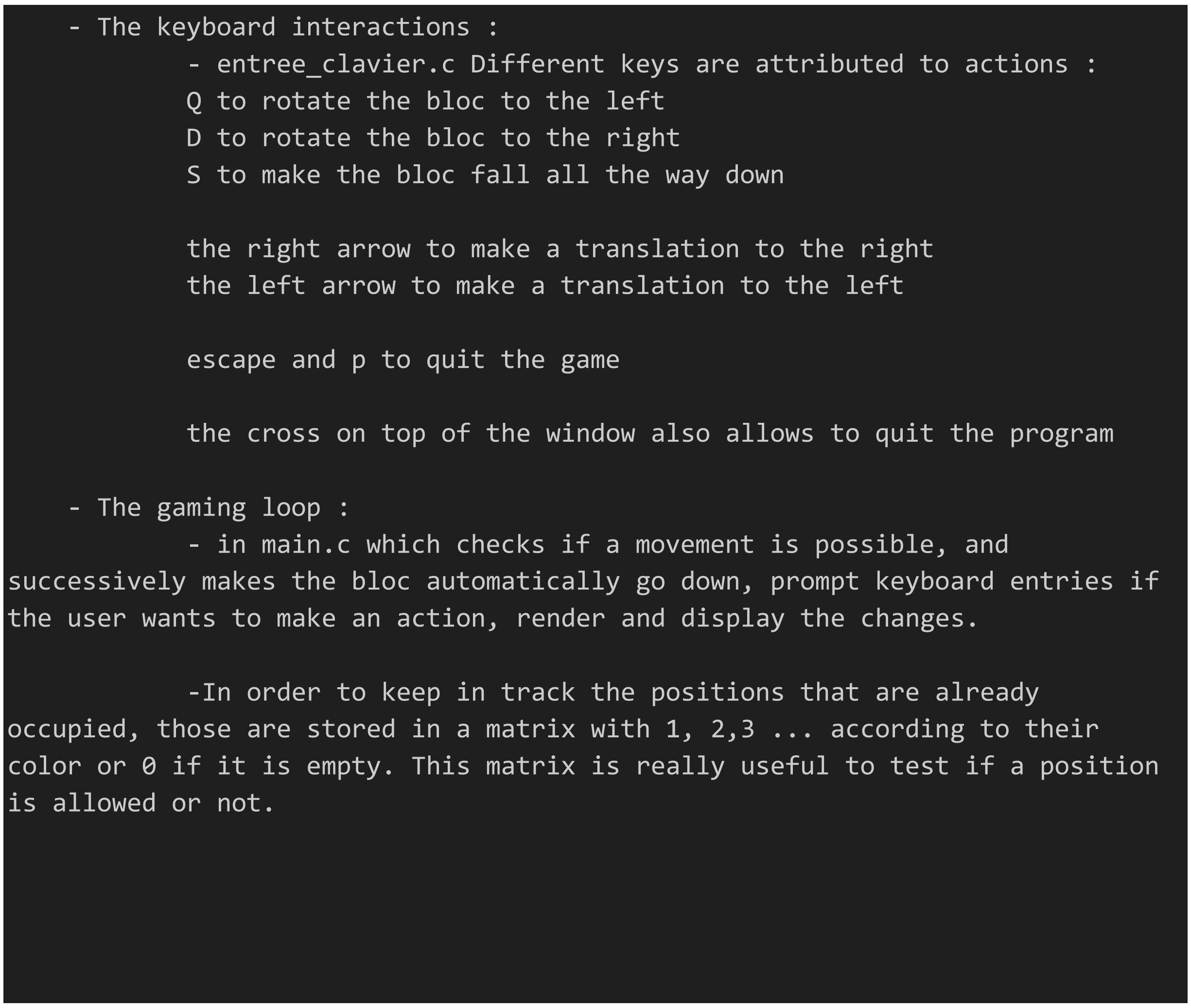
Introduction

This project is about the implementation of a Tetris 2D game in C, using the library SDL2. We decided to start from scratch and do not use any template, so that we familiarize with every aspect of the SDL library in detail and understand everything that we are doing. In this report, we will develop the main functions of our code, the coding paradigm and structure and the challenges that we faced.

Part 1 : README

Here is an extract of the readme file of the project. It presents all the most important functions that we used in our project. It is a good summary and gives a good summary of what we did.

```
-- 2D Tetris --

A project by Edouard Clocheret and Louis Pailloncy which is a 2D tetris game
using the SDL2 library. Have fun !


This git folder "in104_clocheret_pailloncy" contains :

    - a folder "codes" : codes made by the two students

    - the image bloc_names : it is our reference configuration for the 7
tetris blocks

How to use :

    - Compile with "make" command while being in the "codes" folder

    - Have fun playing with the ./game command (in the "codes" folder)

Description of the most important functions :

    -Definition of a bloc :
            - def_types_et_vars.h : The minimal definition of blocks is in
def_types_et_vars.h We refer to them with the enum "shape" for its shape and
the position of the higher left square, and the number of rotations from the
reference configuration.

    - The graphics :
            - initialisation.c initialises the SDL2 library

            - pre_render.c is used to display every bloc according to its
shape (if it is a I, J, L, Z, O bloc), orientation, and position

    - Updating the game state :
             - rotation.c updates the game when a rotation is needed
             - the other movements (suh as translations) are more simple and
just take a few lines in entree_clavier.c

             - collision.c is a very important function that says if the bloc
next position is allowed or not. It calls the function
translate_bloc_to_posistions

             - translate bloc_to_positions.c is a function that translate a
bloc defined minimally defined as described in def_types_et_vars.h (upper in
this README) to the 4 positions that need to be tested. The bloc is either
allowed or not to occupy the position, if the position is not empty.
```

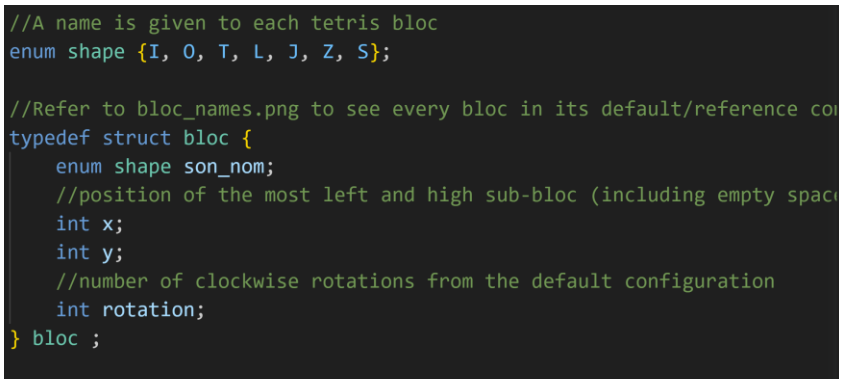


```
    - The keyboard interactions :
            - entree_clavier.c Different keys are attributed to actions :
            Q to rotate the bloc to the left
            D to rotate the bloc to the right
            S to make the bloc fall all the way down

            the right arrow to make a translation to the right
            the left arrow to make a translation to the left

            escape and p to quit the game

            the cross on top of the window also allows to quit the program

    - The gaming loop :
            - in main.c which checks if a movement is possible, and
successively makes the bloc automatically go down, prompt keyboard entries if
the user wants to make an action, render and display the changes.

            -In order to keep in track the positions that are already
occupied, those are stored in a matrix with 1, 2,3 ... according to their
color or 0 if it is empty. This matrix is really useful to test if a position
is allowed or not.
```

Figure 1 – README.md of the project

Part 2 : Coding Paradigm

As we started from scratch, we had to invent everything from the beginning. After a reflection on the code structure and how to represent a Tetris bloc, we came up with an idea. To represent a bloc, a minimal representation is to only store in a structure “bloc” an enum "shape" for its shape and the position of the higher left square, and the number of rotations from the reference configuration.



```
//A name is given to each tetris bloc
enum shape {I, O, T, L, J, Z, S};

//Refer to bloc_names.png to see every bloc in its default/reference co
typedef struct bloc {
    enum shape son_nom;
    //position of the most left and high sub-bloc (including empty spac
    int x;
    int y;
    //number of clockwise rotations from the default configuration
    int rotation;
} bloc ;
```

Figure 2 – Representation of the Tetris bloc by a structure

Hence, we were had to implement another function to complement that : translate_bloc_to_positions. It aims at returning the 4 positions (x and y coordinates) that are occupied by a bloc, given only the structure "bloc".

This is very useful when we try to change the position of the bloc, either because of "gravity" or because the player wants to rotate, or do a translation. This is a 200 lines function, to handle all case, if a bloc is a I, O , T, L, J, Z, S and in any state of rotation.

Collisions

In the main function, we implemented the gaming loop. To treat the collisions, we decided to in the background (ie not displayed), the bloc would undergo any transformation asked by the user, but as soon as it meets a forbidden spot, the action would be undone, and the bloc added to the positions of the still blocs.

Still Blocs

The still blocs are treated totally differently : they are represented as a number 1,2,3 etc according to their color in a matrix, that serves to keep track of the occupied spots. Their value is 0 if the position is empty, and it is what allows to test the empty positions.

To this, we added two functions : game_over to test if the grid is too filled and test_ligne_complete that detects if a line is full and deletes it and consequently updates the grid.

Display

So for the display, we made two functions : pre_render that draws the falling Tetris bloc according to its shape and draw_matrix that only draws the ones that are still. We went for a 16 ms delay to have an approximately 60 frames per seconds.

Part 3 : Challenges

This project taught us a lot about big programs and how manage the different files. We improved our knowledge on Makefiles and it allowed us to discover and appreciate the git files. It is really useful tool to work on big projects with other people.

The other challenge was setting up the development environment. Working on Linux, we had to adapt our habits which were to work on VS Code, but which does not allow to work with a graphic interface. This project was therefore an opportunity to familiarize ourselves with X servers like Xming and to explore other solutions such as virtual machines.

As for the pure code, it was very interesting to start from scratch and have to invent everything. It allowed us to work as a team, argue and make concrete decisions on the code paradigm for example.

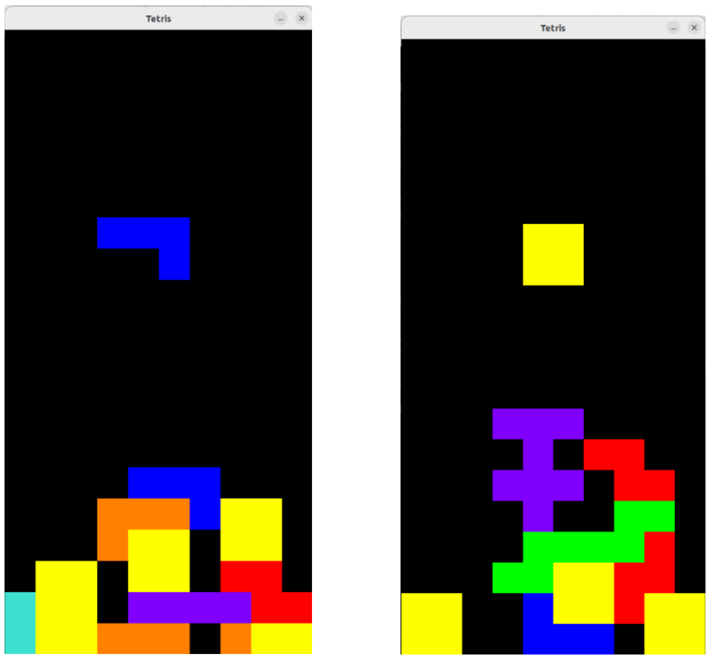


Figure 3 – Different gameplay screenshots

Conclusion

This project was really enriching for us. We have learned a lot, not only about algorithms but also about the development environment. It is very rewarding to get a concrete result with a graphical interface, and allowed us to get familiar with a new library.