COMP6047001 - Algorithm and Programming

Final Project Report

Submitted by: Dennis William (2702342211)

Submitted to: Jude Joseph Lamug Martinez MCS

Table of Contents

Project Specification

Introduction

Tetris is a puzzle video game developed by Alexey Pajitnov, a Soviet software engineer, in 1985. Blocks called 'Tetris' are heading down and your job is to rotate these blocks and try to clear the level by filling in the lines so that you accumulate enough points to progress to the next level.

Idea Inspiration

This project is inspired by my parents. One of my parents' childhood games is Tetris and I want to make this project in an attempt to show some progress in what I chose as my major. Thus, I have decided to remake the all-time classic as a way to start a progress in what I think could be my career. I believe that this project would also be served as a benchmark and a testament in my abilities.

Also, this project is heavily inspired by tutorials I found on YouTube. The links can be found on the reference list.

Problem

With the increasing number of games, Tetris has been slowly left in time. This is because access to old OG games require money (i.e Nintendo Entertainment Console [NES]) Thus, although Tetris will remain timeless for the next couple of decades, the accessibility of this game will be harder to get as time passes by.

Therefore, this project is made with the intention of making Tetris more accessible. With just a click of a button – 'Run' button, it allows users to play Tetris right away, without any unnecessary waiting time. The fact that this program can be run without internet means that this game that I remade can be played anywhere, anytime.

Thought Process

To start this project, I needed to find a reference in YouTube so that I can start somewhere. As mentioned, I started with this one YouTube video which I found that explains the code in detail (link can be found on references list).

I started by making the main class ‘App’, which will handle all the important things such as the main game loop, handling game events, updates, and etc. Next, the class ‘App’ has to be based off somewhere, thus I made three more separate files – ‘tetris.py’, ‘tetromino.py’, ‘settings,py’.

I started making the settings.py first, this is where I put all the constants such as frames per second (FPS), field length, background color, field color, shapes of tetrominoes, assets directory path, etc, and these constants are also based off the module ‘pygame’

Next, I made the file ‘tetromino.py’ and this is where I made two classes – ‘Block’ and ‘Tetromino’. Tetrominoes are basically just groups of blocks, thus I coded the ‘Block’ class first. I set up various methods such as the initialization, taking ‘tetromino’ and ‘pos’ as parameters. Then I coded some functions such as rectangle positions, detection system whether a block ‘is_alive’ (not cleared), and so much more. Then once the ‘Block’ class is complete, I coded the ‘Tetromino’ class, in which I used sprite group since, as mentioned, Tetromino is basically just a group of blocks, and these groups will move as a whole, instead of individual blocks moving.

Next, I made the file ‘tetris.py’ and this is where the scores, array are coded. This class would be operating the ‘Tetromino’ class as whole and would make controls much easier, improving code reusability.

Finally, in the ‘tetris.py’ file, I made a separate class called ‘Text’ and this is where all the text elements (color, background, font, position, etc.) are handled. Furthermore, I also used custom fonts that I have downloaded beforehand.

Thankfully, with the help of online resources, I managed to go through few troubleshoots before my tetris program finally worked.

Solution Design

As discussed in my thought process of this project, this project contains the following:

- main.py
- tetris.py
- tetrominoes.py
- settings.py
- assets (folder)
    - font
        - modern-tetris (TrueType font file)
    - music
        - tetris (OGG file)
    - sprites
        - 0
        - 1
        - 2
        - 3

main.py – This is the main file in which the game will be run on. This file contains the music setup configuration, along with the main class 'App' which is what will be run on this file. The modules imported for main.py are *settings*, *tetris*, *sys*, *pathlib*.

At the end of this file, this code is written:

```
if __name__ == '__main__':
    app = App()
    app.run()
```

This is to ensure that this file main.py is meant to be run on, instead of being imported as a module, which will cause the program to not work as intended.

tetris.py – This file contains two classes – 'Text' and 'Tetris'. The modules *settings, math, tetromino, pygame.freetype* were imported

- In the 'Text' class, this is where the texts are written and displayed.
- In the 'Tetris' class, the following is done:
    - Initialization of game field array
    - Creates a sprite group
    - Sets up initial tetromino and final tetromino
    - Score incrementation
    - Detection system for full lines
    - Adding current tetromino blocks in the array

- Creating and returning empty 2D array
- Detection system for when the game is over
- Detection system for when the tetromino has landed
- Controls for playing the game (i.e using the keyboard W, A, S, D or the arrow keys)
- Drawing the grid using black squares
- Updating game events for the blocks

tetromino.py – This file contains two classes – 'Block' and 'Tetromino'. The modules *typing, settings, random* were imported.

- In the 'Block' class – this is where the configuration for each individual block is made. The following is done:
  - Initialization of Block object, taking '*tetromino*' and '*pos*' as parameters.
  - Visual effects
  - Detection system for when the block is still alive (not cleared)
  - Rotation of block around a specific pivot point
  - Detection system for collision with game field

- In the 'Tetromino' class – Groups of blocks form shapes called tetrominoes. This group of blocks is controlled through this class. The following is done:
  - Initialization of Tetromino object, taking '*tetris*' and '*current*' as parameters.
  - A 'random' module was implemented to determine the tetromino color and shape
  - Rotation of tetrominoes
  - Checks for collision when a rotation is attempted
  - Moves the tetrominoes in a specified direction
  - Updating tetrominoes as it moves down

settings.py – This file contains all the settings and global configuration for the games (i.e frames per second, grid length, etc.) This file contains:

- Frames per second (FPS)
- Field Color
- Background Color
- Sprites Directory Path
- Assets Directory Path
- Animation Time Interval (Normal and Fast)
- Tile Size
- Field Size
- Field Resolution

- Field Scale Width
- Field Scale Height
- Windows Resolution
- Initial Position Offset
- Final Position Offset
- Move Directions
- Shape of Tetrominoes

Discussion

Algorithms

I used Object-Oriented Programming (OOP) to recreate Tetris. Thus, the majority of this code is filled with classes and functions, which will all eventually run through the main game loop, which is the following code:

```
    def run(self):
        while True:
            self.check_events()
            self.update()
            self.draw()
```

The algorithm used on this code is known as the game loop design pattern. It constantly checks for user input, game events, and redraws. This is the core structure of many interactive applications, especially games.

One of the main benefits of using OOP is its troubleshooting mechanism. Suppose you find an error in your code. With OOP, you would find out right away where the error lies in the code, compared to the standard programming method, where troubleshooting could be harder.

Code Reusability is another benefit of OOP due to the concept of inheritance. With this benefit, there is no need to write any more code when a certain part of the code is being reused if it has the same attributes of a class. This allows your code to be more efficient and productive, while also making your code more readable. (Sriram, 2023)

Solution Scheme

A solution scheme in Python programming refers to finding the numerical solution of a problem. In python, there are several libraries that can be used to provide numerical tools, such as *numpy*, *pathlib*, etc. This is the solution scheme that I used for this code:

1. Initialization
- Necessary modules are imported into the code (i.e 'settings', 'Tetris', 'Text', etc.)
- Setting up Pygame, which includes displaying the caption ('Tetris'), screen, clock, custom font, etc.
- Initialize the main class ('App'), which includes creating instances of 'Tetris' and 'Text' classes.

2. Game Loop ('run' method)
- Continuously loops until user quits the game
- Within each iteration of the loop:
    - It checks Pygame events through the 'check_events' method of the 'App' class.
    - Update the game state using the 'update' method of the 'App' class.
    - Draw the game elements on the screen using the 'draw' method of the 'App' class.

3. Text Class ('Text')
- Render various text elements on the screen, including the word 'next'. 'tetris', 'score' and displaying the current user score.
- Utilize the *freetype* module for rendering text with a custom font
- Defines the positions, color, and sizes of each text element

4. Tetris Class ('Tetris')
- Initializes the game state, which includes:
    - Game field
    - Current and next tetrominoes
    - Scoring variables
- Define methods for updating the score, checking for full lines, adding tetromino blocks to the game field, and handling game over conditions.
- Implement controls for moving, rotating, and speeding up the falling tetromino.
- Draw the grid on the game field and update the sprite group to render game elements.
- Update the game state in the update method, including checking for animation triggers and handling tetromino movement and landing

5. Tetromino Class ('Tetromino')
- Represent individual tetrominoes and their behavior, including movement, rotation, and landing.

- Define methods for updating the tetromino state and drawing the tetromino on the screen.
- Handle collision detection and manage the state of individual blocks within a tetromino.

Overall Flow:

- The game loop continuously checks for user input, updates the game state, and redraws the screen.
- Tetrominoes fall, and the player can control their movement and rotation.
- When a line is completed, it is cleared, and the player's score is updated.
- The game continues until the player quits.

Summary:

The solution scheme follows a typical game loop structure, as well as implementing various aspects of the game, such as text rendering, tetromino behavior (i.e moving down, right, left, rotating, etc.), user input handling, within respective classes and methods.

Data Structure

1. Classes
- ‘App’
- ‘Text’
- ‘Tetris’
- ‘Tetromino’

2. Data Structure within Classes
- ‘field_array’ (inside ‘Tetris’)
- ‘blocks’ (inside ‘Tetromino’)

3. Lists and Dictionaries
- ‘points_per_lines’ – Dictionary

```
self.points_per_lines = {0: 0, 1: 100, 2: 300, 3: 700, 4: 1500}
```

- ‘files’ (inside ‘App’) – List Comprehension

```
files = [item for item in pathlib.Path(SPRITE_DIR_PATH).rglob('*.png') if item.is_file()]
```

4. Pygame Sprite Groups
- ‘sprite_group’ - render game elements

5. Pygame Surface Objects
- ‘self.screen’ (inside ‘App’) – representing game window

6. Custom Classes and Functions
- ‘vec’ – representing 2D vector

7. Pygame Events and Constants
- Handle user input
- Set up Timers
- Control the game loop

8. Constants

Some examples include:

- Frames Per Second (FPS)
- Field Color (FIELD_COLOR)
- Background Color (BG_COLOR)

9. Tetromino shapes

Listed in a dictionary, the shapes of the tetrominoes is determined by the following using lists of relative positions (x, y):

```
TETROMINOES = {
    '[': [(0, 0), (1, 0), (0, 1), (0, 2), (1, 2)],
    'P': [(0, 0), (0, 1), (0, 2), (1, 1), (1, 2)],
    'T': [(0, 0), (-1, 0), (1, 0), (0, -1)],
    'O': [(0, 0), (0, -1), (1, 0), (1, -1)],
    'L': [(0, 0), (1, 0), (0, -1), (0, -2)],
    'I': [(0, 0), (0, 1), (0, -1), (0, -2)],
    'Z': [(0, 0), (1, 0), (0, -1), (-1, -1)],
}
```

10. Vector Representation – used to represent 2D vectors which also represents positions and movement directions.

Screenshots

The following screenshots is taken to prove that the program is working. When newly launched or run, tetrominoes will start falling and your job is to rotate accordingly, and accumulate points by clearing lines (Fig 1.1). Once a line is cleared, the lines will be cleared and then the score will be incremented (Fig 1.2)

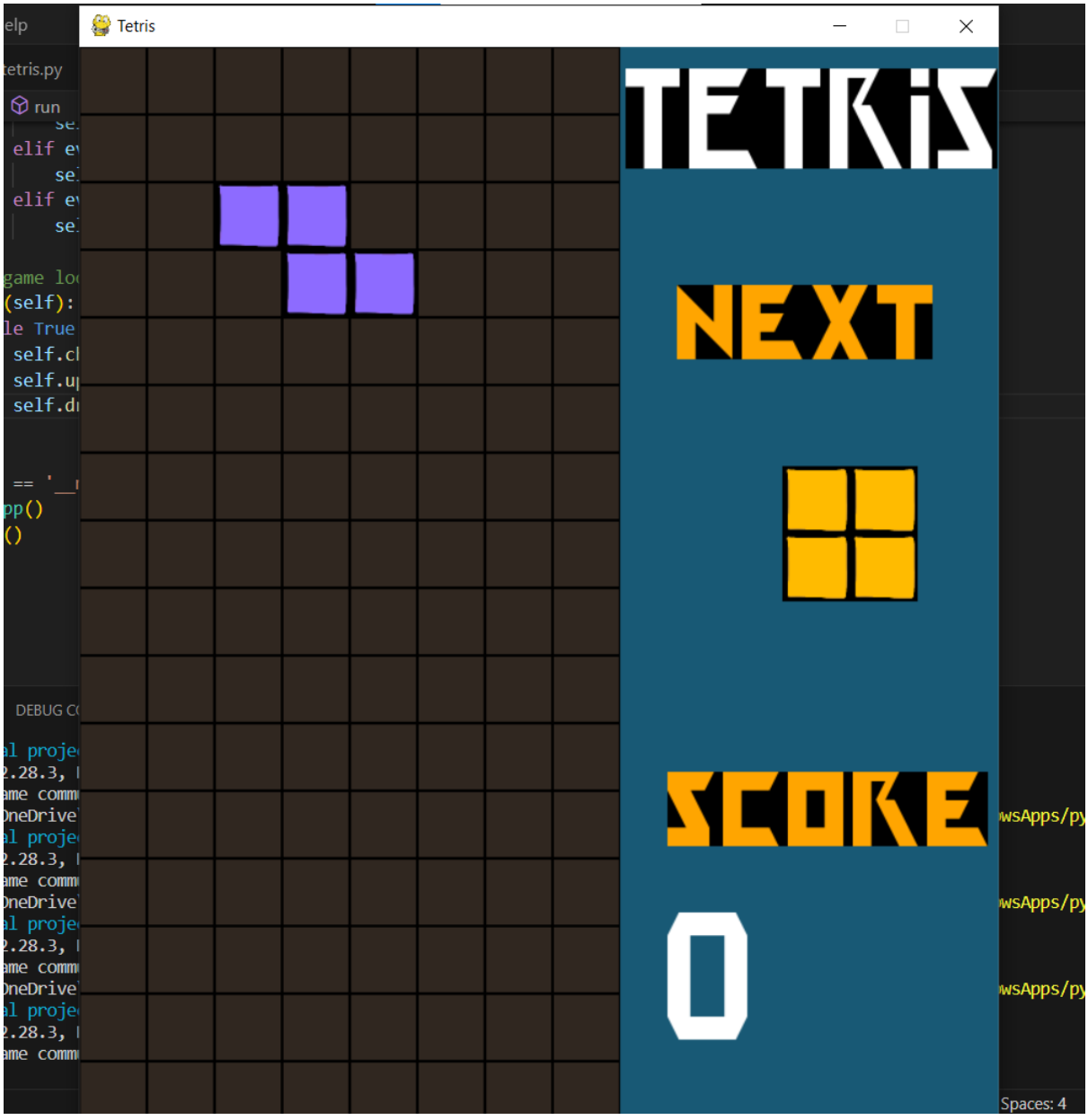

Fig 1.1 – The program after launch

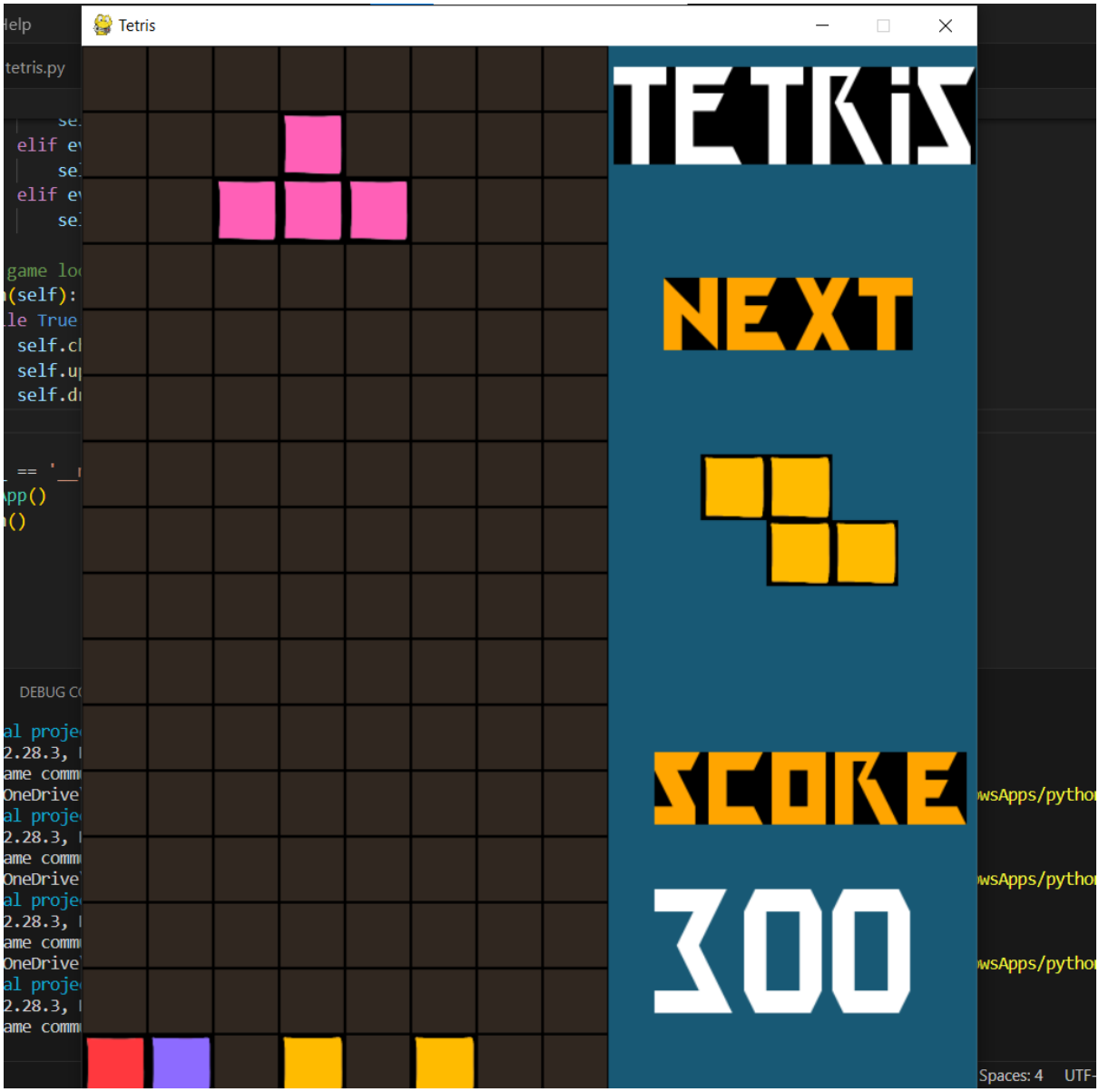

Fig 1.2 – Implementation of the scoring system after lines are cleared

GitHub link & Demo

GitHub link:

https://github.com/DryCrade/algopro-sem1-final-project

Demo link:
https://drive.google.com/file/d/1cZZSiRlYxtLAdHgYubQNOxpqvKhG7_Q4/view?usp=sharing

References

Encyclopædia Britannica, inc. (2024, January 4). *Tetris*. Encyclopædia Britannica. https://www.britannica.com/topic/Tetris

What are the Advantages of Object-Oriented Programming? | upGrad blog. (n.d.). *What are the advantages of object-oriented programming?*. upGrad blog. https://www.upgrad.com/blog/what-are-the-advantages-of-object-oriented-programming/

Numerical methods using Python (scipy)¶. 16-scipy. (n.d.). https://www.southampton.ac.uk/~fangohr/teaching/python/book/html/16-scipy.html

YouTube tutorial:

Detailed Tetris Tutorial in Python

[Creating Tetris in Python with pygame - Beginner Tutorial (OOP)]