Dr. Asieh Parsania
Institut für Mathematik
Universität Zürich

MAT101: Programming

Group Project: Tetris Game

Loreno Heer

Deadline: December 13th, 2019 at 18:00

1 Objectives

In this project you will program a clone of the game "Tetris". We will use the python module pygame to help us implement the game. The main tasks will consist of implementing graphics, collision detection, game physics a highscore system and input handling. A template will be provided in which the code can be implemented. Additionally you are provided a file which specifies the rules of the game. You do not need to follow the rules too closely, but the game has to work in the end and look and feel like regular tetris.

If you have completed the main tasks you can come up with some exta functionality of your own choosing.

2 Prerequisites

Every member of the group needs to know the basic of the python pygame library and should read the game design tutorial: `https://dr0id.bitbucket.io/legacy/pygame_tutorials.html`.

Every member should document what they programm, also indicate which part of the code has been written by whom. Also the code itself should be well documented using proper docstrings.

To use the pygame module you will have to install it by executing the following command from the terminal:

```
python3 -m pip install -U pygame --user
```

You can test that it is working by executing the following command:

```
python3 -m pygame.examples.aliens
```

You can look at the source code of the example programs. They should be in the following directory:

```
~/.local/lib/python3.5/site-packages/pygame/examples
```

You can change to this directory by typing the command:

```
cd ~/.local/lib/python3.5/site-packages/pygame/examples
```

3 Guidelines

The following tasks are to be implemented in the file called template.py:

Task 1 (Graphics)
In this task one handles the loading of graphics and the drawing of graphics on the screen. The `draw_board()` function should take the board as input and draw each tile on the screen. You can use graphics from the internet for the blocks. There is another similar function `draw_preview()` this function should draw a preview of the next piece that falls down. Write a text with the current number of points somewhere.

Use pygame to create a gui and draw the graphics there.

An example implementation using the command line is provided.

Task 2 (Objects)
Have a look at the Board and Piece classes in the template. They define the game board and the moving pieces. Not all methods are implemented. Complete them and make yourself familiar with how those classes work. We use an 2 dimensional integer array internally to represent the board. Any non zero value indicates a block of a piece. Different integers represent different colors. 0 indicates that the position is empty.

```
class Piece:
    def __init__(self, blocks):
        size = (len(blocks), len(blocks[0]))
        # Coordinates of the top left corner of the block
        # They are calculated so that the piece appears centered on top
        self.x = (BOARD_SIZE[0] // 2) - (size[0] // 2) - 1
        self.y = 0

        self.blocks = blocks

    """ Returns a copy of the piece rotated clockwise
        Note that in your game implementation you have to check
        if the rotation collides with the pieces on the board.
        It should not be possible to rotate if doing so results in
        a collision.
    """
    def rotated(self):
        ...

    """ Change the coordinates of the Piece by the indicated amount.
        For example move(1,0) moves it one block to the right.
        move(0,-1) moves it one block down.
    """
    def move(self, x, y):
        self.x = self.x + x
        self.y = self.y + y


class Board:

    def __init__(self):
        self.size = BOARD_SIZE
        self.blocks  = [[0 for _ in range(self.size[1])] for _ in range(self.size[0])]

    """ Returns a Board object where the piece has been added to it.
    """
    def __add__(self, other : Piece):
        ...

    """ Checks if the piece could fit at the current spot in the board
    or if it may cause a collision. Returns false if it fits,
    true otherwise.
    """
    def __or__(self, other : Piece) -> bool:
```

```
        ...
```

Have a look at the list shapes. This looks like this in the template:

```
SHAPES = [
    [[0,1],
     [1,1],
     [0,1]],

    [[2,2,2],
     [2,2,2],
     [2,2,2]]
]
```

Here we defined two pieces corresponding to the shape T and the block shape. Have a look at the specification of tetris to define the other pieces. There should be 7 different pieces in total (different up to rotation).

Task 3 (Main Loop)
In your main loop implement the basic game logic: If no piece is falling, create a new piece (random.sample). Store the next piece in a variable `next_piece` (this is used for the preview window). If a piece is falling, in each iteration move the piece down the y position by 1. Check for collisions and check if the end of the board is reached. If the piece can not move further, add it to the board permanently: `board = board + piece`

Task 4 (Input)
In the main loop of your program (an example is given in the template), implement input handling. At each iteration you should check if the user pressed any key (left, right, down, space, q). Then implement the corresponding action: move the block left right or down, or rotate (Space). Quit the program if q is pressed. Make sure that only valid moves are possible (ie. a block should not be moved outside the screen or through other blocks) (See also collision detection below)

Task 5 (Collision Detection)
Write a function that takes as input a Board object and a Piece object. The board describes the playing board with the pieces that are already fixed. The piece is the currently moving piece. Each piece object has coordinates attached to it. Check if you can add the piece at the current coordinates to the board without causing any collision. Return true or false depending on the outcome.

Task 6 (Line detection)
Write a function which analyses the board and detects if there are any filled out full lines. If there are any, remove those from the board and move the pieces that were above the full lines down accordingly. Return the number of lines removed or the points gained.

Task 7 (Highscore system)
In this task one should load and save the highscores and usernames. Before playing a player should enter his username which will then be stored with the score. The score should only be updated in the file if it is greater than the previous score or if the user did not play before. The 10 best scores should be stored in the file. Other scores should be discarded. The highscore table should be displayed whenever one player quits or looses a game.

Task 8 (Extra Task)
Think about some extra(s) to implement in your game. You can take a look at the official tetris specification for ideas.

Task 9 (Optional Task)
You can use the game code to create a different game of your own design. (Depending on what you want to do, this can be easy or hard).

Feel free to ask me any questions you have while writing the project.

4 References

Make sure you read the documentation of the pygame module. (you dont need to understand everything but you should understand the basics and especially all the parts that are used in the code). You can also read some of the tutorials to understand how games are programmed.

- Pygame documentation: `https://www.pygame.org/docs/`
- Pygame tutorials: `https://www.pygame.org/wiki/tutorials`
- Free game assets (graphics and sounds): `https://itch.io/game-assets/free`
- More free game assets: `https://www.garagegames.com/community/resources/view/23092`
- The Spyder Python IDE: `https://docs.spyder-ide.org/installation.html`
- The PyCharm Python IDE: `https://www.jetbrains.com/pycharm/`
- The Eric Pytho IDE: `https://eric-ide.python-projects.org/`
- Emacs, the best editor (but difficult to use)[1]: `https://www.gnu.org/software/emacs/`
- Vim, the second best editor (also difficult): `https://www.vim.org/`

5 General Notes

- The goal of this project is to experience programming in a group. Discuss the project as a group and then divide the tasks among yourselves.
- You should discuss your progress with the supervisor of the project. Whenever you have questions about your project, feel free to ask them during the exercise class or post them in the forum.
- Once you have written your code you should briefly describe your results. You should include interesting examples and illustrations (if they are part of your project).
- To hand in your project, just send an email to the supervisor of your project. Make sure that it is clear who is responsible for which task.
- It is important that you understand the entire code of your group, not just the part that you have written yourself. In particular, you should be familiar with the prerequisites.
- During the last week of the semester, every group will have a 20-minute presentation of their project. Each member of the group should prepare a 4-minute presentation of their own code and be ready to answer questions about the entire project.
- For the project you will be graded as a group, but for the presentation you will be graded individually. Together the project and the presentation account for 30 percent of your final grade.
- The presentations will take place during the last week of the semester.

[1] Emacs and Vim heavily rely on keyboard use and do not have much mouse functionality