Tetris Assignment Report

Student Name: Yuxiang Feng

Student ID: 32431813

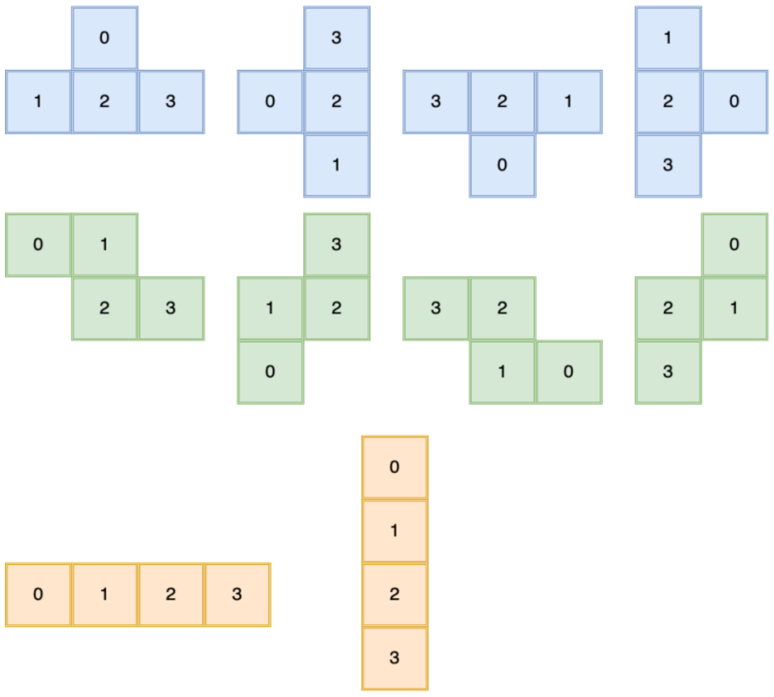
1. Design Decisions

1.1 Overlapping Checking

I use a class to manage different-shaped blocks. Only this class manages block positions and retrieves their information. I implemented double dispatch in the design. The game state calls the class's function, passing itself as a parameter. The class evaluates and decides whether to update the game state, returning a new copy to the tick function. Double dispatch enhances code flexibility by allowing dynamic interactions based on shape.

1.2 Rotation System

For the same reason as overlapping checking, I use classes to manage blocks of different shapes. Therefore, the class represented by each shape will be rotated and managed. The rotation method I designed is that each shape in each shape Each small square will get a unique rotationID. The square with rotationID equal to 2 is the pivot selected by the rotation system. Each rotation rotates the entire shape 90 degrees around the square with rotationID equal to 2.



1.3 MVC

The entire game code is designed based on MVC. The automatically running Observable flow and the player's operation flow together form the controller. The render function is responsible for rendering the view. The pure game state management system I built forms the model.

2. State Management System

The state management system controls the game's state. I design for a single state for clarity and code purity. The tick function in the state.ts file handles state maintenance, making decisions based on the given parameters. Instead of altering the game state, it generates a new copy with the expected changes, ensuring code and data purity.

The game's restart depend this state system. I compare the game's final score with the historical high and decide whether to update it. Then, I create a state based on the initialState, update the high score, and pass it through the pipeline.

Using one function and state for game management simplifies the game logic and ensures the state's uniqueness in the pipeline.

3. FRP Style

In order to follow the FRP style, I did the following in this assignment:

- Disable for-loop
- The modification of the array will be based on functions such as reduce(), map() and filter() that conform to the FRP style
- Make full use of higher order functions
- Use an Observable stream to generate random numbers instead of using Math.random()
- The control of the main game process is completely put into the pipeline of source$

4. Observable and rxjs

4.1 Usage

The use of Observable focuses on the "control" part of the whole game.

Besides the interval runs of the game itself, I have three interesting uses:

- Use fromEvent to monitor the user's keyboard input event
  - By listening to user input, the game can perform movement and rotation operations
- Use fromEvent to listen to the user's mouse click event
  - By listening to user click, the game will respond to the button at the specified location to restart the game
- Use Observable<T> with seed to generate a random number generator

– By combining higher-order functions and Observable, the function returns a random number generator that can be continuously subscribed.

4.2 Advanced

One of the core applications is that I merge multiple Observable streams together to form an Observable. And use my custom type to genericize their data. Merge is necessary in the game I design, because the game consists of a fixed interval running, player input and random number generation. They are all asynchronous and discrete on the time axis, so I need to recombine them to update the game state uniformly.

5. Extension Feature

I designed three extensions for the game

- The first extended function is Star-Block with power-up effect. This block is a small block of size 1x1, and its color is white. When the row it is on is eliminated, the rows above and below it are also removed.
- The second extension function is Bomb-Block with debuff effect. This block is a small block with a size of 1x1, and its color is brown. It will explode when it hits the ground, and five squares including it and its four directions (up, down, left, and right) will be "blown up."
- The third extended function is that players can click the button on the game page to directly start a new game.