FIT 2102 Assignment 1- Tetris

Shannon Johnson – Student #33119864

Introduction

This report will provide an overview of the implementation of Functional Reactive Programming principles and a general higher-level overview of the tetris game assignment. This project aimed to replicate the popular game tetris using FRP and making use of observables for event handling and state management.

Parts of the project:

1. Observable event handling

- Utilised observables extensively for handling user inputs by providing updates when a certain input was processed. These keyboard inputs were used to control parts of the game. The keyboard inputs are mapped different directions which are applied to parts of the game.
- These observables were then used to manage how the game reacts to certain events caused by user input.

2. State Management

- The state is managed purely through an immutable data structure. This ensures that the state changes are predictable and do not contain any unforeseen side effects. Any time the state changes, a new state will be created instead of mutating the previous.
- By using FRP we could represent the game state as an observable stream, with each output from the stream representing a new state. This allowed for successive states to be built on previous ones and allowed me to develop more complex game concepts.
- Each state contains a canvas which is a 2D matrix that contains all information of blocks positions in our game. A matrix was chosen here as its elements can be easily converted to SVG elements to be displayed to the user.
- The state also contained other game information such as the current score. This was important to keep track of and change were necessary.
- The game does not get progressively harder as the levels increase; the level simply just effects a multiplier on the score given to the user.

3. Movement and rotation

- Observables were created for handling specific movement and rotation inputs from the user. These observables were merged into our source observable to simplify the code.
- Movement and rotation are handled by their respective functions which perform the appropriate action on the current state and return a new state with the actions performed.
- Rotation was implemented using the Sega rotation system and making use of the rotation matrix. I chose this system for rotation as the canvas representation of an array made it very easy to apply a rotation matrix to a matrix containing the tetromino I wanted to rotate.
- All movement and rotation actions are confirmed to not be interfering with other blocks to ensure that blocks are not overlapping or replacing one another when they shouldn't be.

4. Other game updates

- Other updates in the game occur on a regular interval. This is processed with the tick$ observable which provides a sequence of numbers every 500ms.
- The updates here include:
    - Checking if the game should be over.
    - Checking if there is a row of blocks which could be removed.
    - Applying gravity on the currently active blocks.
    - Spawning new blocks if there is a need to
- All these updates are performed in a way that maintains purity.
- These other updates could have been improved further by changing how they are called. If I had moved away from simply calling them every tick, then I would have been able to change the interval at which they are updated which would of added to the gameplay.