Rohan Sivam

FIT2102

05 September 2023

FIT2102 ASSIGNMENT 1 REPORT

This report provides an overview of a Tetris game implemented using Functional Reactive Programming using the RxJs Library. The aim of this code was to create a fully functional game with a focus on functional programming principles.

User Input

The first key aspect of this part of code is to listen to keyboard events using RxJS's "fromEvent". It captures key presses for movement, rotation and restarting the game.

Constants

The code defines constants for the game's viewport, tick rates, grid dimensions, and block properties to maintain readability and reusability.

Utility Functions

This code is used to update the grid with new cubes, check for collisions, clear lines and block manipulation.

- setCubeAtCoordinates: updates the grid with cubes at specific coordinates.
- checkAgainstExistingBlocks: checks if a block collides with existing blocks on the grid.
- createGrid: initializes an empty grid.

- removeFullRows: removes full rows from the grid.
- adjustYPositions: adjusts the Y positions of blocks after removing full rows.
- calculateTickRate: calculates the tick rate based on the player's level.

Random Number Generator

This code uses RNG to generate random block shapes and colours. It uses hash based RNG with scaling to generate random values which are used to select from predefined shapes and colours

State Management

The game’s state is managed through an observable stream . The key aspects include the initial state which is the state in which the game starts from or restarts to. Tick represents the game’s main loop updating the state at each tick. The fallingBlock observable manages the falling blocks' collisions, movements and interaction with the grid. It also updates the state with scores, level and calls for new blocks to be generated each time a block is dropped.

Rendering

This code uses RNG to generate random block shapes and colours. It uses hash based RNG with scaling to generate random values which are used to select from predefined shapes and colours

Observables

There are a few observables created to manage the games state changes and events. These include

- key$: captures keyboard events.
- movement$, tickdown$, and movementTickDown$: manage movement and tick events.
- getRandomBlockShape$: selects random block shapes.
- spaceBarPress$: handles spacebar presses.
- fallingBlock$: manages the falling block's behavior and interactions with the grid.
- source$: represents the overall game state over time.

Functional Reactive Programming

This code follows the functional programming principles by using pure functions, immutable data structures and observables to update game data and states. It uses RxJS to handle game events and state changes reactively. For example with keydown events.

Design Decisions

In this code, I feel like I have made some interesting design decisions, firstly, would be representing block shapes as a 2D array of 1s and 0s instead of relative coordinates like what most people would do. I felt that this way of representing my block shapes made it easier to update my grid and render out block shapes. I also used a 2D array to represent my grid as it allowed for easy updating of the state by creating shallow copies and generating new grids with updated rows and columns based on the cubes placed. I used the W key as my rotation button as it is usually used as the forward movement key so most people will be used to pressing it.

CONCLUSION

This Tetris game code successfully implements a functional and interactive game using RxJS and FRP principles. It provides a clear and organised structure, making it easy to understand and extend. As I have stuck to the functional programming style, it maintains immutability and readability.