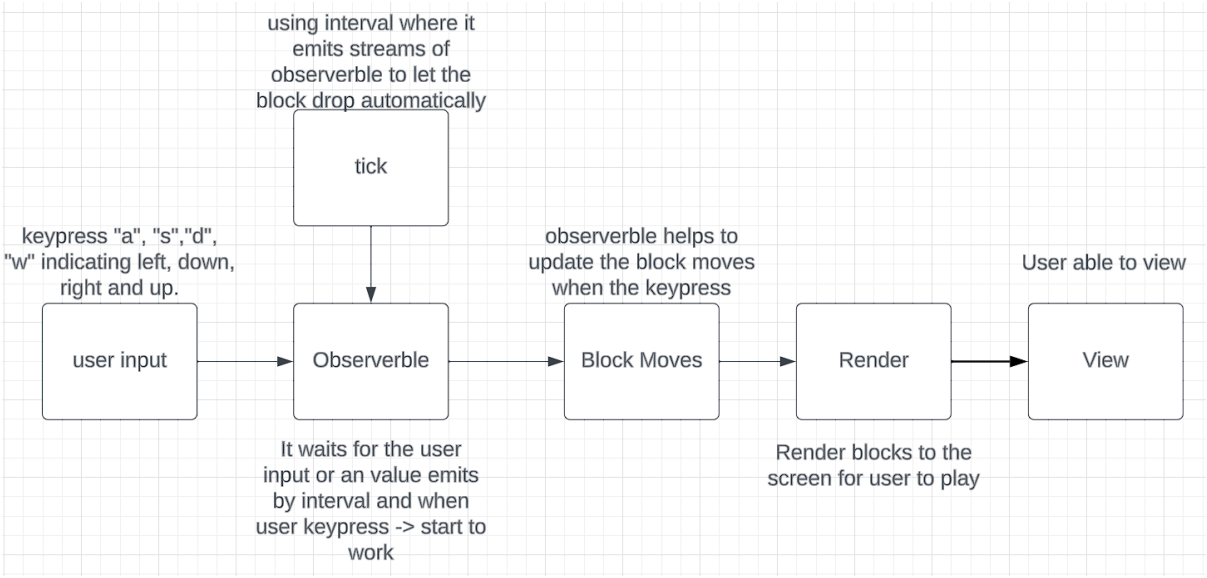
Name: Teh Jia Xuan	Student ID: 32844700
Assignment 1: Functional Reactive Programming	



Creating Tetrominos:

- Create all shapes made of 4 cubes
- Using merge and scan to change the x and y value
- Insert x and y to function that creates the shape by adding Block.Width and Block.Height.
- It returns an array of shape's coordinates that adds up the current x and y coordinate.
- All shapes have different colours except for the moving one
- The moving one always will be white -> differentiate from the others

Design Decision (DD):

- Merge multiple observables into a stream of observable
- The merged observable comes in order
- Scan to accumulate values over time

Render:

- Utilising the coordinate the shape function.
- Loop through every cube's coordinate to create an Svg element
- Append the cube to update the canvas

Block moves:

- Subscribe to observable
- call render to modify the x and y
- a keypress, the block moves by updating the position

Descending blocks

- Use interval create observable to decrement the block by timestep. Where decrement the block by a fixed size of 10.

DD:

- Using interval it emits value automatically certain amount of time
- Easy to implement

Collided blocks

- Implement a 2D Array with boolean values
- Whenever the block drops, check whether the block below is occupied in 2D Array
- If it is, then there is a collision
- Add to block list

DD:

- 2D Array helps me to keep track of which grid is occupied

Side Collision

- Using 2D Array to check side blocks
- Return original position if there is

Score & HighScore

- Loop through 2D Array
- which row is fully occupied
- Delete and add a row on the top array
- Add to player's score
- Implement highscore by comparing the previous round of score

Rotation system (Nintendo rotation system)

- "w" Merge with other observables
- Hard coded all forms of rotation of the tetrominoes
- Store it in list
- Whenever "w"is pressed it rotates 90 degrees clockwise

DD:

- Move and rotate at the same time
- Asynchronized stream
- easy to implement
- Instead of thinking for few hours about doing math

Next shape preview & Upcoming shape

- Randomly choose a shape using hash
- Store in the preview
- When the current block collides
- preview block assigned to current block
- Get another random block for preview and upcoming shape

Restart game

- Creates a function to set everything to initial state
- This impure code is wrapped under a function
- It stays at that particular function
- Maintain purity -> clear boundary of impure and pure code -> easier to maintain
- Add an event listener
- When gameEnd and “r” is pressed it resets the state and hides gameover

DD:

- Convenient way to restart

Increase difficulty

- Level increases by 1, for every 500 points obtained by player
- The tick rate divided by level
- lower tick rate faster speed
- Speed increases with the level increase

DD:

- Observable stream, emit values every timestep
- Automatically increase the speed when level increases
- convenient

game finishes

- When a block collides and the block y position is 0
- It is at the top
- Set game continue to false where it terminates render and show gameOver

FRP style

- Uses observables to represent asynchronous data streams
- Apply merge to merge multiple observables and In arriving order
- Apply scan to accumulates values over time
- Separates observable, application of operators and subscription
- Using higher-order function

Interesting usage observable

- Able to handle user interaction
- Chain variety of operators using pipe
- Only emits data when someone subscribes to
- Asynchronise stream

State management

- Almost all of the function takes state as parameter and it returns new state Instead of modifying it
- States have readonly properties for purity
- Scan ,concat etc is used for modifying and updating state-> they maintain purity
- Reuse state that return by other function