FIT2102 Assignment 1

Student name: Lee Chi Kit (Aiken)
Student ID: 35423056

How to Play

Since I don't have a UI telling user how to play it. I would add it here.

Flap	Space Key
Pause/Resume	P Key
Restart	R Key, only works when you win/died

State Management

The core of the game is a single immutable State object, which encapsulates properties like birdY, birdVelocity, gameTime, etc. We can see this as a singleton design. By doing this, we can avoid creating global variables and side effects, making it easy to predict. Updates occur through pure reducer functions that return new state objects without mutating the original.

The game loop is similar to the Unity Update() method. But instead of setting the Update related to the game's fps, we set it with an interval observable emitting every 20ms. This handles physics (gravity, velocity), pipe movement, collision detection, and scoring in a single pass. ((To integrate pipe spawning with the game's internal clock (respecting pauses), I replaced real-time timer observables with a pipeSpawn$ derived from tick$, checking gameTime against each pipe's spawn time. This ensures spawning only occurs when the game is active, avoiding desynchronization during pauses.))

User input is captured via fromEvent observables for key presses. Flaps, pause toggles, and restart are merged into the main state stream using functional operators (map, merge, scan). No imperative listeners directly mutate the DOM or state.

Collision and Bounce Mechanics

Collision detection uses pure functions. I've split the type of collision into 3 parts. Top, Bottom, Pipes. Identifying where we hit and therefore choosing that if we should bounce up or down. The bird bounces with a "random" velocity generated via a seeded RNG (from the applied class). Which ensures that we would get the same result with the same seed.

Pipe Handling and CSV Integration

At startup, the CSV file is parsed into an immutable array of pipe descriptors. Pipes are spawned into the state only when their scheduled time matches gameTime. Once they scroll off-screen, they are removed with .filter. The score is gained after we pass the pipe by calculating the pipe x and width and compare it to the bird X. Then we will mark the pipe to prevent adding the same pipe score multiple times.

Handling Time and Events

A key design choice was to base gameplay around an internal game clock (gameTime) rather than real-time. If we use the actual time function. If we pause the game, the time is actually still moving and when we return to the game, every pipes will generate at the same time due to them all meeting the required time to spawn. By changing this to a custom time logic, we allowed pausing and invulnerability to integrate smoothly, because advancing or freezing the clock directly controls all time-based mechanics.

Pipe spawning, for example, is derived from this clock. A pipeSpawn$ stream checks scheduled times from the CSV against the current gameTime. This ensures pipes only appear when the game is active and remain synchronized even after pausing.

Functional Programming Style

Throughout the code, I applied principles from functional programming. However, there are parts that is not pure functional by its nature. For example, state transitions are always pure, but rendering introduces side effects. It’s something to do with the svg part.

Additional Features

Pause

I implemented a Pause feature to allow the player to temporarily stop the game by pressing P. This works by extending the state with a paused boolean. The tick stream then check this flag: if paused is true, we simply returns the same state without advancing time or spawning pipes. The keydown handler toggles pause. And it is being merge and scan.

Invulnerability

I also implemented Invulnerability. When the bird hits anything and lost a life, it will have a 2 second cooldown on losing another life and will not bounce when it hits something. This is done using the gameTime that we implement (Our own gametime instead of the time function, because we want the pause to actually stop the time). We also set the birdImg to flashing to indicate that.