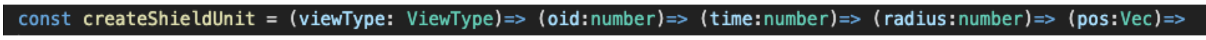
1. Summary of the code:

The tries remain loyal to the main promise of FRP, which is linearising and simplifying the flow of control. The main body of the code is divided into 5 parts:

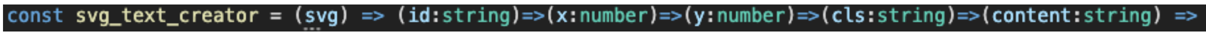
1. ***Body Generator:*** functions for creating bodies/objects that are involved in the game
2. ***Controller:*** Creation of the observable streams
3. ***State:*** Continuously updates the state of the game which could be altered by either the user or time.
4. ***View:*** Updates the game display based on the game state. The only non-pure section of the code.

2. Principles:

- **Functional:** Where possible, all the functions are defined in a curried way to remain loyal to functional paradigm and potentially alleviate function reusability.
    - **Example:**

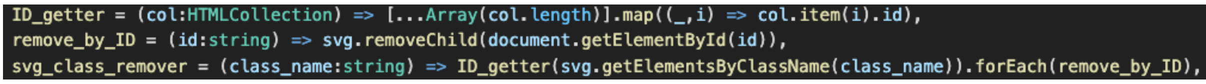


```
const createShieldUnit = (viewType: ViewType)=> (oid:number)=> (time:number)=> (radius:number)=> (pos:Vec)=>
```



```
const svg_text_creator = (svg) => (id:string)=>(x:number)=>(y:number)=>(cls:string)=>(content:string) =>
```

- **Reusability:** Where possible, the functions are defined in a hierarchical, compact manner, so that new functions can be derived through composition of smaller functions.
    - **Example:**
        - The third line combine the functions in the first and second line.



```
ID_getter = (col:HTMLCollection) => [...Array(col.length)].map((_,i) => col.item(i).id),
remove_by_ID = (id:string) => svg.removeChild(document.getElementById(id)),
svg_class_remover = (class_name:string) => ID_getter(svg.getElementsByClassName(class_name)).forEach(remove_by_ID),
```

- **Minimal Streams:** Since every subscription creates a new stream, and at the end of every subscription mutation occurs, there is a single subscription for the entire game.

3. Notable Challenges:

- **Alien Movement:** To manage aliens' movement, I have specified a special function for them which recognises the "right-est" and "left-est" alive alien at any time and orders the aliens to "bounce and descend" whenever these edge aliens hit the boundaries of the screen.

- **Alien Shooting:** To manage alien attacks, I have defined a function that finds the "frontier" aliens and randomly picks one to order shooting at a given interval.

4. Features:

- **Lives:** The player is given 3 lives for every level. The number of remaining lives is indicated in the top right of the screen.

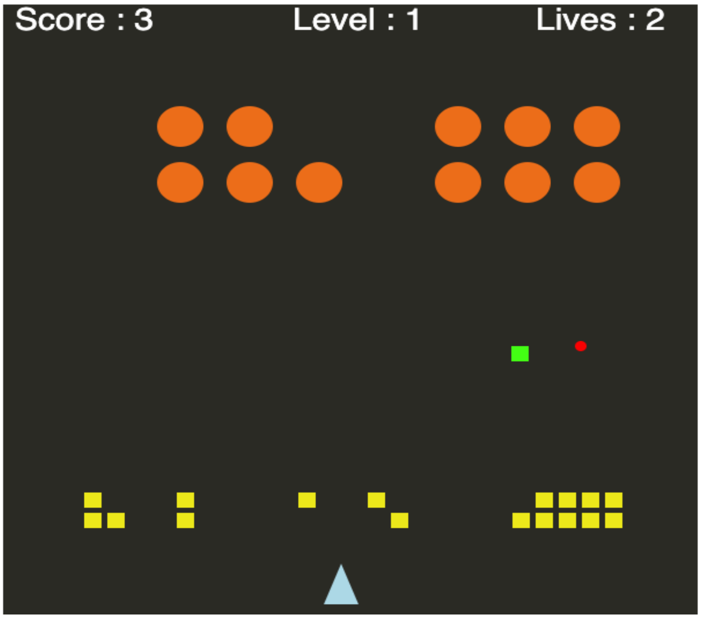
- **Restart:** Players can restart the game in the middle of a game or at the end of a game. The restart takes back the player to the *first level*. No second chance is given at any level.

- **Difficulty/Levels:** After winning a game, the player can proceed to the next level by pressing “F2”, as it is indicated on the screen. As the level goes up the game gets more and more difficult. Namely, the number of aliens increases, the speed of aliens increases, and aliens shoot more often.

- **Shields:** The shields are made up of smaller blocks (shield units) and can be destroyed by both aliens and the ship. (Just like the original game)

- **Power-Ups:**

    - **Shields Rebuild:** Looks like a green shield unit, and if the player catches it they can renew their shields instantly.


Score : 3
Level : 1
Lives : 2