# Space Invader Report

**Aw Sheng Xiang | 31259030**

---

Type | Objects:

| | | |
|---|---|---|
| <Body> | Alien | Spaceship |
| | Shields | Bullets |
| <Stats> | stats | |
| <State> | Game state | |

**Overview:**

*“Main framework of game flow is constructed using observable stream and models any possible event as a flow of observable”*

Interval function emits an observable at a constant interval to simulate the passage of time in game. Interval is then merged with other functions that emit an observable stream upon firing of events such as keyboard input from users such as keydown or key up event. A map function is then used to map these streams of observables to return objects of different class types so that it can be used in the flow control in the reduce function.

A scan operator function is then used to capture the observable streams and a reduce function is used to apply necessary transformations to the elements in the accumulated state of the gameplay and return a new state object with transformed elements. This is inline with the principle of *Functional Reactive Programming* style. The type of object passed into the scan operator dictates the type of transformation that will be applied to the state object with the initial state as the starting value of the reduce function

The transformation functions that are applied to the state/body objects are pure functions such that they do not alter any values in the object directly or any variables be it inside or outside of the scope. Rather a new object is returned with values resulting from the transformation assigned to the element in the object. This is also true for the transformation of elements in an array where the array used is always declared as a readonly array to prevent the elements from within to be mutated directly. Instead the map, filter and reduce functions are used to deal with the elements which results in a new array with updated changes applied to the elements being returned. Variables used in the functions are also declared as constant and therefore not mutable. Transformation function might call another function from within its scope, however as the functions called are also pure the side effects are effectively contained.

The subscribe function is then used to apply the updateCharacters function where the objects are added, moved or removed from the svg canvas visually. This is the only place where an imperative style function with side effects is allowed. This helps in limiting the

possible side effects to a singular place in the whole code which is in the subscribe function allowing debugging to be carried out more easily.

**Start of game:**
A state object called initialState is created with the elements within initialised with starting values

- ➔ Characters that need to appear in the display at the start of game are created by calling functions that creates a certain count of the characters as object of type Body
- ➔ Objects such as bullets that are not needed yet are not created and therefore assigned as an empty array

**During gameplay:**
Events that happen throughout the gameplay is wrapped in the tick function which is called at every interval of the observable stream if no keyboard event is detected:

- ➔ Alien moves left/right then down

Aliens are given an initial velocity either to the left or right when created to start moving immediately. They are also given a maximum distance that it can move before it has to reverse its velocity and also move down
The maxDistFilter function is used filter the alien array for aliens that has moved maximum distance so that they can be mapped with a new velocity which is reversed horizontally and has a vertical velocity
The aliens array is then mapped using the moveObj function that adds the velocity to the coordinates of every alien at every call of tick function

- ➔ Alien shooting at random

An interesting aspect of this is the need to update the aliens that will be shooting if one of them is removed due to collision. To achieve this, I have chosen to include a rank in the alien object to easily identify and grab them in the alien array and maintain an array of aliens that are currently shooting by storing their rank. The array is maintained by cross checking the array with the damaged aliens array and matching them using their rank. If an alien in the shooting array is found in the damaged array, the task is passed to another alien 1 row above in the same column. However, there might be no alien found, therefore, to solve this problem of finding a successive alien, a recursive function validPos is used to traverse each row before until either an alien is found in the particular column or when it reaches the last row and no alien is found. In the case where no alien is found, the column in the array of aliens shooting will be updated with a negative value which will be filtered out when the array is processed for the aliens to shoot

The frequency aspect of the shooting is controlled by a prob() function which takes in the probability of an alien creating a bullet and also firingInterval() function which ensures that the alien is only able to have a chance at shooting every interval.

- ➔ Spaceship action

Spaceship actions are mainly carried out by the input of keyboard depending on the key that is assigned to a particular action and an object of the action type will be returned and fed into reduce function as input

Movement of spaceship is controlled by giving the spaceship object a velocity of value of [1,-1,0] where by 0 is given to stop the spaceship, the movement is carried out by the moveObj function which adds the velocity to the coordinates of the spaceship at every call of tick function

- ➔ Collision

Collisions between alien-bullet, shield-bullet are handled in the same fashion by flatMapping the 2 arrays so that their output is an array of all combinations of the elements as a pair and filtered using the objectCollide function to check for the distance between 2 objects that returns a truth value
The filtered array is then separated into 2 arrays again so that the collided objects can be removed independently of each other. However, for shield-bullet collision, as the shield has the ability to take one collision multiple times before being completely destroyed, it is not removed immediately upon collision, rather the damage element of the shield object is updated to reflect the total damage it has taken thus far. However, if the damage threshold is reached, then the shield object will be removed from the shield array
As for space-ship bullet collision, the bullet array only has to be filtered using the objectCollide function with the spaceship as the second input

**Game Over:**

- ➔ There is 2 possible scenarios when the game ends:
  - ◆ Level is cleared which means that all the aliens are shot down
    - The levelCleared function is used to check if the current score equals to the total score gained by shooting down all the aliens at every interval. The value returned by the function is then assigned as a switch to the gameover element in the state object
  - ◆ Spaceship has collided with a bullet from the alien and the player lost
    - The collision of ship and bullet is checked by filtering the bullet array for bullets that collided with ship and checking if the length of the filtered array is more than 0 indicating that at least 1 collision happened between ship and bullet. The value returned from the check is also assigned to the gameover element in the state object
    - Restart mechanism
      - The restarting of the game is implemented by using an observable stream of keyboard event whereby when the ‘r’ key is pressed, a stream of object type Restart will be emitted which will be used by the reduce function to identify that the game has to be restarted and thus clearing the canvas of any existing objects and returning the initial state object so that the accumulator of scan function will start with the initial state of the game again effectively resetting the game to a new gameplay as the state object is what defines the state of the game it is in.

- ➔ In the updateCharacters function, the movement of other objects are no longer updated resulting in a display freeze and a text indicating player is lost or a level is cleared will be appended to the svg depending on the truth value of gameover element and levelCleared function

# References

Alien.png from
*https://www.pikpng.com/pngvi/JoTxmx_space-invaders-alien-png-image-background-8-bit-space-invaders-png-clipart/*

Shield images from
*https://www.raspberrypi.org/blog/coding-space-invaders-disintegrating-shields-wireframe-9/*

Spaceship image from
*https://steamcommunity.com/sharedfiles/filedetails/?id=277472506*

Font from *https://www.fontspace.com/minecraft-ten-font-f40317*

Code structure adapted from *https://tgdwyer.github.io/asteroids/*