31158145 YI JIE NG

FIT 2102 Assignment 1 REPORT

GAME DESCRIPTION and RULE

1. **Control**
   Arrowleft: Space ship move to left side until it reached the boundary.
   ArrowRight: Space ship move to right side until it reached the boundary.
   ArrowDown: Space ship stop moving and stay at its current position.
   Space: Shooting action.

2. **Rule**
   - Player took a single shot or the aliens reached the baseline, game lose.
   - Score is gained per bullet hit with alien.
   - Total 5 stages, vertical movement speed and bullet shooting rate of aliens increased stage by stage.
   - 3 phases per stage
     - First phase (Alien > 15): Normal mode
     - Second phase (Alien >1 && <= 15): Alien bullet rate increased, player gain a double bullet number buff.
     - Third phase (Alien==1): Alien movement speed, firing rate and bullet number increased significantly. Player gain a buff of triple bullet.
   - Stage and phase will be reset back as 1 if player dead. High score is recorded.

GAME DECISION MADE TO IMPROVE THE GAMEPLAY

1. **Space Ship movement**

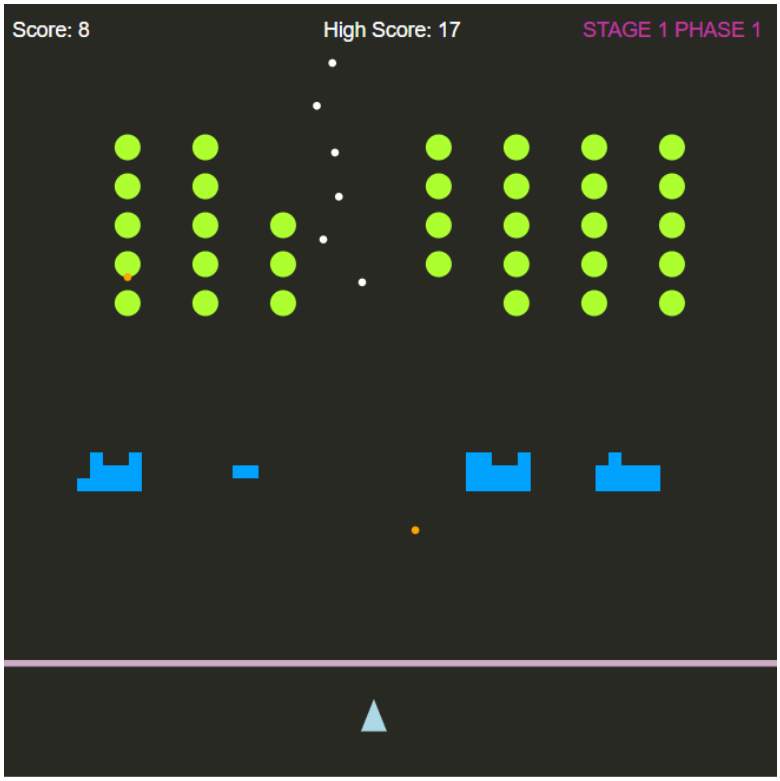
After assigning html code of the space ship which declare its polygon size and color, we know that we can move the spaceship by changing its transform attribute which include x and y. Thus, I have created an observable stream for keyboard event which filter out the keys I needed for further implementation. Initially I had decided to design the movement with press and hold the ArrowRight key will result the spaceship to keep moving within boundary until the keyUp event was triggered by the same key same as ArrowLeft which control left motion. After some justification, I realized that the spaceship will not response for a while if the player keeps holding ArrowRight key, hold ArrowLeft key at the same time and release the ArrowRight key while keep holding the ArrowLeft key. This result the movement of spaceship not smooth and fluent due to the keyboard event we declared. Hence, I decided to remove the keyup event of ArrowLeft and ArrowRight, making them keep moving with only one press and added a stopMove observeKey for the ship to stop moving.

```
288  // Observable stream for keyboard event
289  const observeKey = <T>(e:Event, k:Key, result:()=>T)=>
290  fromEvent<KeyboardEvent>(document,e).pipe(
291    filter(({code})=>code === k),
292    filter(({repeat})=>!repeat),
293    map(result)
294  )
295
296  // Movement available for the ship
297  const moveLeft = observeKey('keydown', 'ArrowLeft', ()=> new Move(-1)),
298  moveRight = observeKey('keydown', 'ArrowRight', ()=> new Move(1)),
299  // stopLeft = observeKey('keyup', 'ArrowLeft', ()=> new Move(0)),
300  // stopRight = observeKey('keyup', 'ArrowRight', ()=> new Move(0)),
301  stopMove = observeKey('keydown', 'ArrowDown', ()=> new Move(0)),
302  shoot = observeKey('keydown','Space', ()=>new Shoot())
```

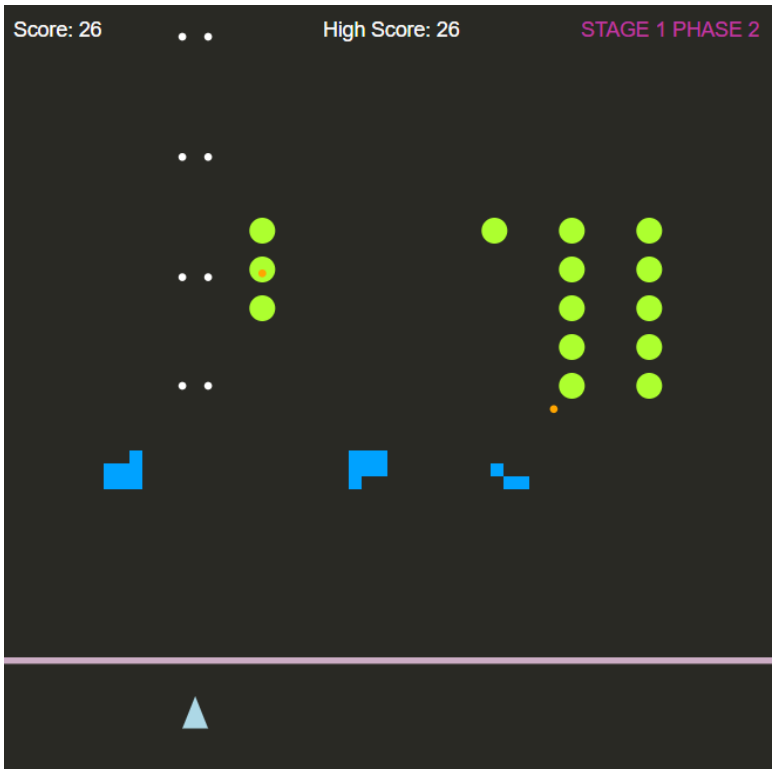
By setting the boundary at the reduceState, we can now move our spaceship by changing its x value adding with xDirection according to the keyboard event.

2. Game Phase

In order to make a game more interesting, distinct exciting and interesting game scenarios should be carried out within a stage. Instead of the aliens keep moving left right and down, either shoot the player or shot by player or reach the baseline with constant speed, I have decided to separate and divide the game to 3 phases. For phase 1 (number of aliens > 15), all of the aliens and player are in normal mode.



During phase 2 (number of aliens <=15 && > 1), the bullet rate and speed of the aliens will increase. Player will be buff by increase bullet number from 1 to 2 and their bullet speed increased significantly as well.



When there is left with only one alien, the game is then entered to phase 3, where the alien will now move rapidly and shoot 5 bullets per attack, trying to reach you baseline or kill you faster.

Such implementation made the game more fun and challenging to play. To declare the phase, we can always check the alien size to define the current phase. We can make use of elapsed of the state to control the bullet shooting rate by using mod. The larger the mod value, the slower the alien or player can shoot. For the bullet speed, we can control it by changing the yDirection of the bullet body. In order to create rows of bullets, I have created extra function which help to declare the extra bullets interval and information, concatenate all the bullets using concat built in function to create a list of bullets body.

3. Game Stage

It will be pretty boring if the game contained only a stage with constant phases without a doubt. Thus, I decided to design 5 stages each with 3 phases respectively to let the game harder and interesting. After analyzing the code, I realized that I can increase the bullet rate and vertical movement speed of the aliens to define the stage difficulty. I added stage attribute inside the state, increasing the bullet rate and vertical movement speed of the aliens by including the stage value in the calculation.

4. Restart of the game

It will be pretty annoying if the player is required to restart the game manually by pressing the refresh button or Ctrl + r button every time they win or lose. To resolve the issue and improve quality of the game, I added 4 new attributes including gameWin, gameOver, victory and highScore to the state of the game. gameWin indicates when all the aliens within the current stage are killed and the game will then enter to next stage when its value become true at this condition. When entering next stage, all of the state attributes will become the same as initialState except the highScore and score attribute. For gameOver, for every stage, when the player took a shot by a single bullet, it will become true and resulting the same as gameWin, except the score will be reset as well. victory attribute value will only be true when the stage is currently 5 and all of the aliens are killed. When player wins the game, the game stream is unsubscribed and a congratulation message will pop up. User can restart the whole game by pressing Enter key at this moment. highScore attribute will never reset since it is the attribute to record down the player highest score for the game.

STATE/GAME CONTROL (Following FRP style)

1. Initial State

When the game first start, all of the state attributes are set as the same as const initialState. We declared const intialState by referring to a list of immutable variables in Constants. We can control the overall game play behavior by changing the value of the Constants. Once we run the game, they are then all immutable and fix and all of the body will follow the default value set at Constant until we stop running the game. Hence, it is act like a control center for the game.

```
// Immutable variable (Act like control center)
const
  Constant = {
    CANVAS_SIZE: 600,
    BULLET_EXPIRED_TIME: 500,
    BULLET_RADIUS: 3,
    ALIEN_RADIUS: 10,
    ALIEN_COLUMNS: 5,
    ALIENS_ROWS: 8,
    ALIEN_PHASE1_BULLET_RATE: 120,
    ALIEN_PHASE2_BULLET_RATE: 80,
    ALIEN_PHASE3_BULLET_RATE: 65,
    ALIEN_NORMAL_BULLET_SPEED: 1.5,
    ALIEN_RAPID_BULLET_SPEED: 3,
    SHIP_PHASE1_BULLET_SPEED: 1,
    SHIP_PHASE2_BULLET_SPEED: 3,
    SHIP_PHASE3_BULLET_SPEED: 5,
    DEFAULT_QUANTITY: 1,
    SHIP_BULLETS_INTERVAL: 10,
    ALIEN_BULLETS_INTERVAL: 15,
    SHIELD_X: 50,
    SHIELD_Y: 350,
    SHIELD_RADIUS: 5,
    SHIELD_INTERVAL: 100,
    FINAL_STAGE: 5
  } as const
```

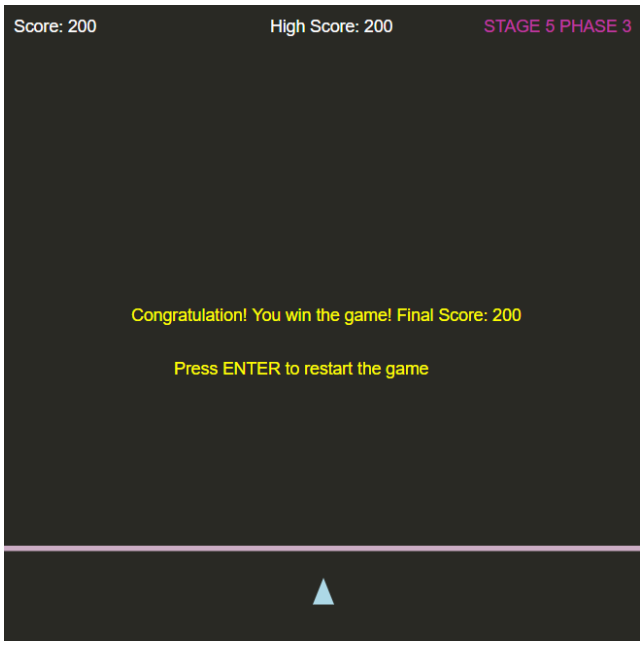
By doing this, we can ensure that our initial state will always be remained as pure since it is declared as immutable.

2. Gameplay State

When the game is running, we always create and return a copy of state with some modification instead of changing or updating the exist state given in order to make sure our function is pure. For example, when the stream was captured by an action of Move, if x of the ship given by current state need to be modified (the ship is now not at boundary), we make a copy of the state attributes given and change the x value by movement calculation and return this copy of state. We purely refer to the state given and do not modify or update it in order to achieve Functional Reactive Programming. If we choose not to modify the current state, we can then return the exist state as long as we guaranteed not to touch or modify it. When the stage < 5 and the aliens within the stage are all killed, it will then return a copy of state same as initial state we declared previously except some special attribute such as score or highScore. Same goes to game lose, when the player was hit by a single bullet, the game will then restart and reset back to stage 1 phase 1 by reading a copy of state from initialState as well except the highScore attribute. In short, the game will always return a new state and assign it to function updateView which give an accurate user interface of the game per tick until the game was finished. Functional Reactive Programming is then achieved by such implementing way.

3. Victory State

When player won the game (passing all 5 stages), the subscription will finally be unsubscribed and the observable stream is now ended. The state is then now unrelated to us anymore since we have finished the game. From start to the end of the game, the state is purely defined and it does not change or been modified once it is created. Observable stream being used, declaration and usage of variables in the game are all immutable and thus we can say that our program is fully following the FRP rule. If the player now presses an Enter key, the whole game will now reload and refresh.



CITATION

- Asteroid Game notes:
  https://tgdwyer.github.io/asteroids/

- Asteroid Game coding:
  https://stackblitz.com/edit/asteroids05?file=index.ts

- svg create basic shape:
  https://poselab.com/en/blog/creating-basic-shapes-with-svg/

- HTML assist website:
  https://www.w3schools.com/howto/howto_css_image_center.asp

- Randomly choose an element from an array:
  https://stackoverflow.com/questions/5915096/get-a-random-item-from-a-javascript-array