Report

— Abstract Information

* Author : Chenghao liu (Individual project)

* Topic : Escape Game

* Images : soldier..png, block.png, bullet.png, enemy.png, background.jpg, landmine.png

— Project Descriptions

* Step 1: **Move** is created as a constructor which contains the properties of object's image, starting points x and y,
the size of the image(width and height) and

isColliding method checking if player collides on the object.

```
function Move(img,x,y,width,height)
{
this.character = new Image();
this.character.src = img;
this.X = x;
this.Y = y;
this.width = width;
this.height = height;
this.Velocity_X = 0;
this.Velocity_Y = 0;
this.Weight = 0;
```

* Step 2: Make use of the constructor we just created. Create **objects** and draw all the characters(player, block, enemy…) on the canvas.

```
var player = new Move("soldier.png",
0,200,32,44);
var block = new Move("block.png",
100,368,32,32)
var enemy = new Move("enemy.png",
500,350,50,50);
```

```
var landmine1= new Move("landmine.png",
250,380,32,20);
var landmine2 = new Move("landmine.png",
10,380,32,20);
var landmine3 = new Move("landmine.png",
350,380,32,20);
var landmine4 = new Move("landmine.png",
50,380,32,20);
```

* Step 3: **timer** is an event which has setInterval method. Call the function draw every 60 milliseconds.

```
function timer()
{
    var time = Math.floor(1000 / 60);
    setInterval(draw,time);
}
```

* Step 4: **draw** is the big function which is to draw player's movements, draw object on canvas by calling drawImage(), avoid glitches (checking when players collides on blocks and enemy etc(will explain details later on).

* Step 5: Create **addListener event**(keydown and keyup) to control the movement of the player. There are three directions players can go: up, left and right which are corresponding to the keycodes: w, a and d. When player press down w, isUp variable set to true and player move up 15 units by plus -15(negative for up) to player.velocity_Y. And player.Y += player.velocity_Y so that player can move. And the same techniques applied to left and right movement. If the variables are true the movements will be triggered. When user releases keyboard, the corresponding variable becomes false. Player stop moving.

— Player moves by 2 units by player.X += player.velocity.X. and velocity.X is 2 or -2. Weight is also applied to up movement. I set weight = 1 so player will go on the ground eventually. If player is on the ground set onGround = true.

```
player.X += player.Velocity_X;
    player.Y += player.Velocity_Y;

    if(isLeft)
    {
        player.Velocity_X = -2;
    }

    if(isRight)
    {
        player.Velocity_X = 2;
    }

    if(!isLeft && !isRight)
    {
        player.Velocity_X = 0;
    }
if(player.Y >= canvas.height -
player.height)
    {
        if(isUp && player.Velocity_Y == 0)
        {
            player.Velocity_Y += -15;

        }
        else if(isUp && !
player.isColliding1(block))
        {
            player.Velocity_Y += -15;
        }
```

* Step 6: In **draw** function. In order to avoid glitches when player sink out of canvas. If player is on the ground (onGround is true and no collision), we put player back to the canvas.

```
else
        {
            player.Velocity_Y = 0;
            player.Y = canvas.height - 
player.height;
            onGround = true;
        }
```

* Step 7: Check collision. In **Move** constructor. There are six **isColliding methods** in my project. But they use the same technique to perform.

- This method is to check when the player reaches the edges of block (up, left and right directions) if the player stops moving.(stay above the block, stop when player is at the left or right sides of the block.)

```
this.isColliding1 = function (block)
{   var collided = false;
    var collidedX = false;
    var collidedY = false;

     // check if player collides the block

     if(player.X + player.width > block.X &&
player.X < block.X + block.width)
     {
         collidedX = true;
     }

      if(player.Y + player.height> block.Y
&& player.Y < block.Y + block.height)
     {
         collidedY = true;
     }
     collided = collidedX && collidedY;
     return collided;
}
```

- To do this we need to check both x and y. If Both x and y dimensions meet collision conditions, it means we collide on an object. In other word, when the player is within the block, we need to check both x and y pixels. And return a combined variable as boolean.

* Step 8 : In **draw function**. There are six methods to check every objects when our player collides on. To avoid glitches (player goes through block, landmines and enemy, cannot jump over the block), we need add few codes.

- Y pixel: when player's location is overlapped with the block(go through block), we put player back to the block and set velocity is zero(stop). If we want to jump over the block, one condition is needed to add as follow.

- X pixel: we need to calculate the distance between position of player and block. When player goes through the block we need to put it back to the left or right side of block. If distance left less than right, player is closer to the left side. Then we drag player to the left side of block and set the speed as zero(player stop moving). Same as checking right side.

```
if(player.isColliding1(block))
   {
      if(player.Y + player.height <= block.Y 
+ player.Velocity_Y)
      {
            if(isUp)
            {
                  player.Velocity_Y += -10;
                  player.Y += player.Velocity_Y;
            }
            else
            {
                  player.Y = block.Y - 
player.height;
                  player.Velocity_Y = 0;
            }
      }

      else if(player.X + player.width + 
player.Velocity_X > block.X )
      {
            var left = Math.abs(player.X - 
block.X);
```

```
        var right = Math.abs(player.X -
block.X - block.width);
        if(left<right)
        {
            player.X = block.X -
player.width;
            player.Velocity_X = 0;
        }
        else
        {
            player.X = block.X +
block.width;
            player.Velocity_X = 0;
        }
    }
  }
```

- ps. The rest of five **isColliding methods** are same as this except one point. Landmines have onGround Boolean variable. If user reach any of the landmines, onGround turns true, game will be over.(This is the code to let game over).

```
if(player.Y + player.height >= landmine1.Y)
            {
                if (onGround &&
player.isColliding3(landmine1))
```

```
                    {
                        alert("Game over");
                        location.reload();
                        return;
                    }
                }
```

* Step 9 : Build a **constructor** for the bullets enemy shoots. In this constructor, we have four attributes. Position x and y, velocity x and boolean active(which used to determine if bullets exists).

```
var bullets = [];
for(var i = 0 ; i < 1000; i++)
{
    bullets[i] = 
    {
        x: 400 - 50 * i,
        y: canvas.height - 15,
        vx: -1,
        active: false
    };
}
```

* Step 10 : create a new function called **drawbullet**. Obviously, this function is to draw the bullets and display the messages when player’s dead or survived.

- This if statement I write in draw function(does not matter where it is , can put in **drawbullet** function). And make every bullet is true

```
    if(Math.random() < 0.025)
     {
          for(var i = 0; i <
bullets.length;i++)
          {
               if(bullets[i].active == false)
               {
                    bullets[i].active = true;
               }
          }
     }
```

- This for loop is to check every bullet. Briefly, if a bullet collides player’s body. Then he will die. To check it, I made **three boolean variables,playerleft, playerright** and

playedown. The last variable is triggered when our player touches bullets vertically.

— When bullets is going to pass through the block, change active property to false , so that bullets will stop as they reach the block.

```
for(let i = 0; i < bullets.length; i++)
    {
        if(bullets[i].active)
        {
            bullets[i].x += bullets[i].vx;
            if(bullets[i].x < -10 ||
bullets[i].x <= block.X + block.width)
            {
                bullets[i]. active = false;
                bullets[i].x = enemy.X;
            }
            var playerleft = bullets[i].x >
player.X ;
            var playerright = bullets[i].x
< player.X + player.width - 10;
            var playerdown = bullets[i].y <
player.Y + player.height;
```

- When the player is killed by bullets, we prompt a message game is over. onGround boolean variable we applied when checking collision.
And onGround is except the case when player stand on the floor.(x, canvas.height).

```
if(onGround && playerleft && playerright &&
playerdown)
            {
                alert("Game over");
                location.reload();
                return;
            }
```

- In this function, we also need to print other **message** when the player is killed by landmines or the player is survived.(4 lanmines, 4 game over message and one success message)

— Game over message displays when player collides landmines. And refresh page to start game again.

— user will win the game when player goes out the right side of canvas

```
 if(player.Y + player.height >=
landmine1.Y)
            {
                if (onGround &&
player.isColliding3(landmine1))
                {
                    alert("Game over");
                    location.reload();
                    return;
                }
            }
```

```
if(player.X + player.width >= canvas.width)
        {
            alert("You win!!");
            location.reload();
            return;
        }
```

- Inside for loop, before survived message, I **draw the bullets**.

```
            ctx.beginPath();

ctx.arc(bullets[i].x,bullets[i].y, 4, 0,
Math.PI * 2);
             ctx.fill();
             ctx.closePath();
```