Manual Técnico.

Juego de Mario Bros (Plataformas)

Proyecto de la Segunda Unidad

Profesor: Mario Humberto Rodríguez Chávez

Integrantes: Adrián Silvestre Amaya Padrón,

Erick Alberto Rodríguez Rodríguez

Indice

Introducción

El Manual Técnico, como su nombre lo indica, contiene las especificaciones técnicas más importantes del sistema desarrollado. Constituye una guía especializada del desarrollo del juego. Se encuentra dirigido fundamentalmente al profesor Mario Humberto Rodríguez Chávez, el cuál imparte la clase de Herramientas Multimedia en la Universidad Politécnica de Victoria, al público que se le pueda otorgar.

Objetivo

Crear un juego fácil de manera a como se pide y entregar un manual Técnico del juego.

Primer Frame (portada):

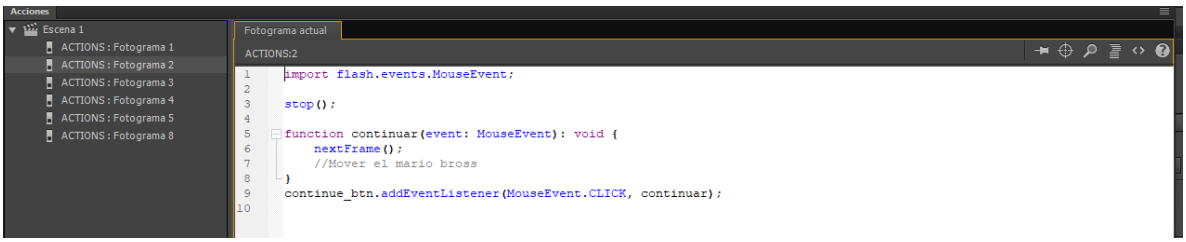
```
stop(); //Detenemos en este frame

var musica: Sound = new Sound(new URLRequest("Select player.mp3")); //Buscamos el audio
var channel: SoundChannel = musica.play(); //Reproducimos

var sonido: Sound = new Sound(new URLRequest("Castle.mp3"));
var channel1: SoundChannel = sonido.play();
channel1.stop();

var espacio: Boolean = false; //Creamos bandera para cuando el jugador presione enter
var playerA:Array = new Array(); //Array donde se guardarán los datos del jugador 1
var playerB:Array = new Array(); //Array donde se guardarán los datos del jugador 2

//Función para apretar tecla
function jugar(event: KeyboardEvent): void { //Funcion de tipo KEY_DOWN
    if (event.keyCode == 32) { //si el evento del teclado es 13, se esta pulsando la tecla de enter
        espacio = true; //Activamos la bandera
    }
}

//Funcion para soltar las tecla
function soltarJugar(event: KeyboardEvent): void { //Funcion de tipo KEY_UP
    if (event.keyCode == 32){ //Si no se esta presionando la tecla izq
        espacio = false; //Desactivamos la bandera
    }
}

function selec(event: Event): void {
    //Si enter esta presionado nos cambiaremos de frame para que se seleccione el jugador
    if (espacio == true) {
        stage.removeEventListener(Event.ENTER_FRAME, selec);
        gotoAndStop(2); //Nos movemos al frame 2
    }
}

//EVENTOS QUE LLAMARAN A LAS FUNCIONES PARA MOVER EL PERSONAJE
```

En este fotograma existen 3 funciones las primeras 2 son del tipo para verificar pulsaciones del teclado y la 3 es de tipo evento.

La primera función (jugar) es para saber si se está presionando la tecla con código ASCII número 32 que es el espacio, una vez que se presiona se pone en verdadero la variable “barra Espacio”. La segunda función es solamente para saber cuándo se deja de presionar la tecla espacio. Y la última función es para mandar al frame 2 cuando ya la variable “barra Espacio” es verdadera y se quita el evento de tipo enterFrame de la respectiva función.

También se inicia la música que se mantiene hasta el momento de jugar.

Instrucciones:



Fotograma numero 2: En este fotograma lo único que hacemos es mandar al otro frame después de mostrar cómo se juega.

Elección de un Personaje:

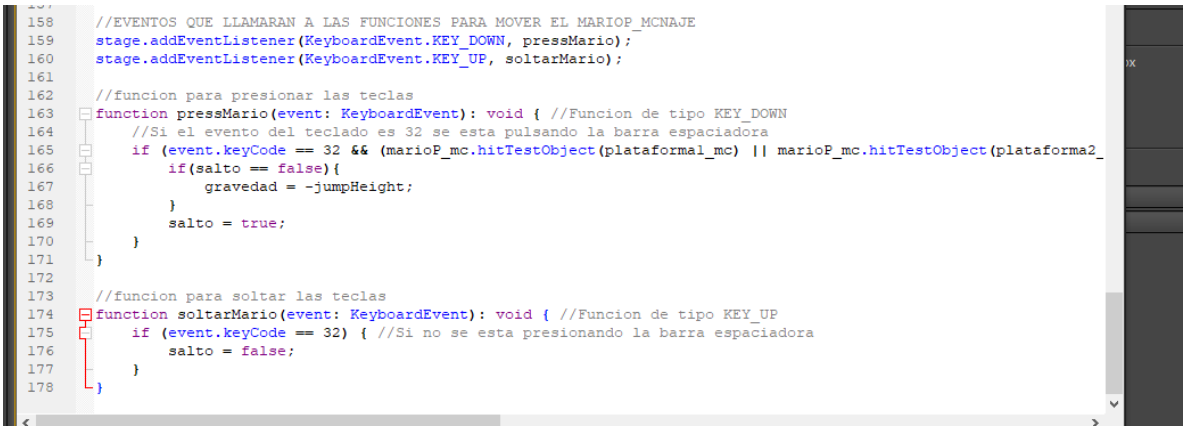

```
function marioB(event: MouseEvent) {
    registrar();
    if (bandera == false) {
        if (persona == 1) { //Si el contador de la persona es 1
            player1.push(player_txt.text); //Guardamos en el array del jugador 1
            player1.push(0); //Ponemos 0 por que será que se escogió a Mario
            error1_txtd.text = ""; //El texto de error lo volvemos vacío
            player_txt.text = ""; //La caja de texto la ponemos vacía
            ingNombre_txtd.text = "Jugador 2:"; //Ponemos que sea jugador 2
        } else { //Sino
            if (persona == 2) { //Cuando persona sea 2
                player2.push(player_txt.text); //Guardamos en el array del jugador 2
                player2.push("0"); //Ponemos 0 por que será que se escogió a Mario
                movFrame();
            }
        }
    } else {
        bandera = false;
    }
}
mario_btn.addEventListener(MouseEvent.CLICK, marioB);
```

En esta función lo que se hará será el proceso de elección del personaje de la persona, esta podría elegir entre Mario Bros (1), Luigi (2), Toad (3), Peach (4) donde cada uno tiene su propia función para la elección. Primeramente, mandamos llamar a la función para validar el registro llamada “registrar”.

```
function registrar() {
    errorl_txtd.text = ""; //El mensaje de error lo ponemos nulo
    nombre1 = player_txt.text; //Guardamos en una variable lo que haya ingresado
    if (nombre1 == "") { //Verificamos si no es nulo
        bandera = true; //Activamos bandera en caso de que este vacío el texto
        errorl_txtd.text = "Ingrese un nombre"; //Mostramos el mensaje de que ingrese un nombre
    } else { //Sino
        for (i = 0; i < (player_txt.text).length; i++) { // Obtenemos la longitud de la cadena
            nombre1 = (player_txt.text).charAt(i); //Extraemos la letra que se encuentra en la posición i
            //Compararemos para ver sino no es un numero
            if (nombre1 == "0" || nombre1 == "1" || nombre1 == "2" || nombre1 == "3" || nombre1 == "4" || nombre1 ==
                errorl_txtd.text = "Ingresa otro nombre"; //En caso de que si sea un número mostramos el mensaje
                player_txt.text = ""; //La caja de entrada
                i = (player_txt.text).length - 1; //Le damos valor a i para que se salga del ciclo
                bandera = true; //La bandera la ponemos true
            }
        }
    }

    if (bandera == false) { //Cuando salga del ciclo y la bandera sea falsa
        persona++; //Aumentamos el contador
    }
}
```

En esta función lo que hacemos primeramente es obtener el valor de la caja de texto y poner la caja de error vacía. Verificamos si es nulo el valor en este caso nombre1, y si no contiene un valor entonces activamos la bandera y mostramos "ingrese un nombre". Todo lo anterior es para el caso de que no exista un valor ingresado en la caja de texto. El siguiente "else" entra a una condición que valida con un ciclo for desde 0 hasta la cantidad de caracteres de la cadena. Y compara si no existe algún numero en cualquier lugar de la palabra ingresada, en caso de ser cierto de que contenga un numero se muestra en la casilla de error un "ingrese otro nombre" y le asignamos a i para que salga del ciclo posteriormente ponemos nuestra bandera verdadera. Al último cuando sale del ciclo existe una condición que pregunta si la bandera es falsa entonces aumentamos el contador de las personas.

Salto del personaje:



```
158  //EVENTOS QUE LLAMARAN A LAS FUNCIONES PARA MOVER EL MARIOP_MCNAJE
159  stage.addEventListener(KeyboardEvent.KEY_DOWN, pressMario);
160  stage.addEventListener(KeyboardEvent.KEY_UP, soltarMario);
161
162  //funcion para presionar las teclas
163  function pressMario(event: KeyboardEvent): void { //Funcion de tipo KEY_DOWN
164      //Si el evento del teclado es 32 se esta pulsando la barra espaciadora
165      if (event.keyCode == 32 && (marioP_mc.hitTestObject(plataformal_mc) || marioP_mc.hitTestObject(plataforma2_
166          if(salto == false){
167              gravedad = -jumpHeight;
168          }
169          salto = true;
170      }
171  }
172
173  //funcion para soltar las teclas
174  function soltarMario(event: KeyboardEvent): void { //Funcion de tipo KEY_UP
175      if (event.keyCode == 32) { //Si no se esta presionando la barra espaciadora
176          salto = false;
177      }
178  }
```

En la línea 159 y 160 se hace el llamado de estas dos funciones que son de tipo key_down y key_up. La primera función llamada "PressMario" la utilizamos para saber si la tecla del espacio esta presionada, esto lo hacemos con la condición de la línea 165 la cual dice si la tecla esta presionada y si el personaje en este caso marioP_mc esta colisionando sobre alguna de las plataformas que existen en el juego (en este caso 3) lo que hará será entrar a otra función que nos verifica si la variable booleana salto es igual a falso. En caso de que esto último sea verdad a la gravedad del personaje se le restara la variable "jumpHeight" y si la variable "salto" es falsa se pondrá verdadera.

La segunda función llamada soltar Mario es de tipo key_Up y lo que hace es validar si la tecla espacio no está presionada, y si no es así se vuelve falsa para que se pueda usar.

Función para contar los precipicios:

```
//Función para detectar cuantos precipicios brincó
function precipicioMario(event: Event) {
    if (preci == false) {
        if (marioP_mc.x >= (plataforma1_mc.x + 280) && marioP_mc.x <= plataforma2_mc.x) {
            cantPreci++;
            trace(cantPreci);
            precipicios_txtd.text = String(cantPreci);
            preci = true;
        }

        if (marioP_mc.x >= (plataforma2_mc.x + 280) && marioP_mc.x <= plataforma3_mc.x) {
            cantPreci++;
            trace(cantPreci);
            precipicios_txtd.text = String(cantPreci);
            preci = true;
        }

        if (marioP_mc.x >= (plataforma3_mc.x + 280) && marioP_mc.x <= plataforma1_mc.x) {
            cantPreci++;
            trace(cantPreci);
            precipicios_txtd.text = String(cantPreci);
            preci = true;
        }
    }
}
```

En esta función de tipo event lo que hacemos es que si la variable preci es falsa y si las coordenadas en x de nuestro personaje son mayores o iguales a las de la siguiente plataforma +280 y menores a la siguiente plataforma entonces aumentamos la cantidad de los precipicios, los mostramos en nuestra caja de texto y ponemos nuestra variable preci verdadera.

Función para la distancia recorrida

```
//arrancar el recorrido
distancia.start(); //Arrancamos el tiempo
distancia.addEventListener(TimerEvent.TIMER, distanciaMario); //mandamos llamar a la funcion de tiempo tipo timerEve
//Función para los metros y los centimetros
function distanciaMario(tiempoevent: TimerEvent): void { //Funcion del timer de la distancia
    centimetros++; //se aumentan los centimetros
    distancia_txtd.text = metros + " m," + centimetros + " cm."; //mostramos desde un inicio los centimeros y metros
    if (centimetros == 100) { //una vez que los centimetros sean igual a 100
        centimetros = 0; //se reinician los centimetros
        metros++; // y se aumentan los metros en 1
        distancia_txtd.text = metros + " m," + centimetros + " cm."; //mostramos para actualizar la distancia
    }
}
```

Lo primero que hacemos es arrancar la distancia que es el evento para el timer. Una vez que la iniciamos mandamos llamar a la función "distanciaMario" que es de tipo timerEvent que lo que hará será incrementar la variable centímetros primeramente y mostrarlos. Cuando los centímetros sean iguales a 100 entonces se reiniciarán a 0 y se aumentara 1 a la variable llamada metros y volvemos a mostrar las variables en la caja de texto distancia_txtd.

Función para el tiempo

```
//Función para el tiempo con minutos
function tiempoMario(tiempoevent: TimerEvent): void {
    tmp++; //aumentamos el contador
    if (tmp == 20 || tmp == 40 || tmp == 60) {
        velocidad += 2;
        movimiento += 0.1;
        jumpHeight++;
        channel1.stop(); //paramos musica
        channel1.stop(); //paramos musica
        channel1 = sonido.play(); //reproducimos musica
    }
    if (tmp == 60) {
        min++; //aumentamos los minutos
        tmp = 0; //reiniciamos el contador
    }

    if (min == 0) { //si minutos es 0
        if (tmp <= 9) { //si tiempo es menor o igual a 9
            tiempo_txtd.text = "00" + ":0" + tmp; //mostramos el tiempo
        } else {
            if (tmp >= 10) { //si tiempo es mayor a 10
                tiempo_txtd.text = "00" + ":" + tmp; //mostramos el tiempo
            }
        }
    } else {
        if (min != 0) { //si tiempo es distinto a 0
            if (tmp <= 9) { //si tiempo es menor o igual a 9
                tiempo_txtd.text = "0" + min + ":0" + tmp; //mostramos el tiempo
            } else {
                if (tmp >= 10) { //si tiempo es mayor o igual a 10
                    tiempo_txtd.text = "0" + min + ":" + tmp; //mostramos tiempo
                }
            }
        }
```

Una vez que comenzamos a jugar se inicia un timer que se le agrega un timerEvent que manda llamar a la función tiempoMario. Lo que hace esta función es que cuenta los segundos y se almacena en tmp. Cuando la variable tmp es igual a 60 se agrega 1 a la variable min y se reinicia, así como mostramos en el txt del tiempo. Cuando los segundos están en 20, en 40 y en 60 se aumenta la variable de la velocidad y del movimiento así como el valor de jumheight (variable utilizada para el salto) así como reiniciamos la música ya que solo dura poco tiempo.

Cuando los minutos aún están en 0 y cuando los segundos son menores o iguales a 10 en la caja de texto se mostrarán dos ceros seguidos de dos puntos y otro cero para que le dé un mejor aspecto a el tiempo del juego, toda esta condición cambiara una vez que los segundos transcurridos pasen de 9. De igual manera si los minutos son diferentes de 0 o sea que ya existe alguno y los segundos son menores o iguales a 9 se mostrara un 0 luego los minutos seguidos de dos puntos, otro cero, y la variable de los segundos.

Función de inicio

```
function inicioMario(event: Event) {
    channel.stop();
    channel1 = sonido.play();
    if (finalJ == false) {
        nombre_txtd.text = player1[0];
        distancia_txtd.text = "";
        precipicios_txtd.text = "";
    } else {
        nombre_txtd.text = player2[0];
    }

    //variables para el timer para mostrar una distancia
    centimetros = 0; //contador para los centimetros
    cont1 = 0; //contador del timer
    metros = 0; //variable metros que se aumentara cuando los centimetros sean 100

    //variables para el timer del tiempo
    tmp = 0; //contador para los segundos
    min = 0; //minutos
    cont2 = 0; //contador del timer

    cantPreci = 0;
    colisionLava = false;
    juegoF = false;
    salto = false;
    brinco = true;
    gravedad = 2;
    movimiento = .2;
    terminar = false;
    marioC_mc.visible = false;

    //Acomodor plataformas en Y
    plataforma1_mc.y = 350 + (Math.random() * 100);
    plataforma2_mc.y = 350 + (Math.random() * 100);
    plataforma3_mc.y = 350 + (Math.random() * 100);
```

El objetivo de esta función llamada "inicioMario" en este caso, son las de iniciar la música que se reproduce al momento de iniciar a jugar, mostrar el nombre del jugador en el respectivo campo de texto (para saber cuál nombre del jugador se mostrar se pone una condición para saber si finalJ es falsa entonces está jugando el primer jugador, en caso de que sea verdadera está jugando el segundo), vaciar los campos en caso de que sea el segundo jugador. Así mismo reiniciamos las variables bandera de las colisiones los brincos, termino del juego, etc. Todas las variables de los contadores del tiempo y de la distancia son igualadas a 0. Se crean los random para acomodar las plataformas en el eje de las ***Y***.

```
        plataforma3_mc.y = 350 + (Math.random() * 100);

        //Las linea antes de la plataforma la asignaremos a la misma altura de la plataforma
        linea.y = plataforma1_mc.y + 5;
        linea2.y = plataforma2_mc.y + 5;
        linea3.y = plataforma3_mc.y + 5;

        //Ejecutamos las funciones
        //Ejecutamos los movimientos de las plataformas
        addEventListener(Event.ENTER_FRAME, platMario1);
        addEventListener(Event.ENTER_FRAME, platMario2);
        addEventListener(Event.ENTER_FRAME, platMario3);

        addEventListener(Event.ENTER_FRAME, fondoMario);
        addEventListener(Event.ENTER_FRAME, caidaMario);
        addEventListener(Event.ENTER_FRAME, lineaMario);
        addEventListener(Event.ENTER_FRAME, colisionMario);
        addEventListener(Event.ENTER_FRAME, acomodarMario);
        addEventListener(Event.ENTER_FRAME, precipicioMario);

        addEventListener(Event.ENTER_FRAME, apagarMario);
        removeEventListener(Event.ENTER_FRAME, inicioMario);
    }
    addEventListener(Event.ENTER_FRAME, inicioMario);
```

Dentro de la misma función de "inicioMario" se le asigna a las líneas que son utilizadas para las colisiones la misma altura de las plataformas +5, Una vez que reiniciamos todas las variables que serán utilizadas para el correcto funcionamiento del juego se manda llamar a los eventos que moverán las plataformas, serán los encargados del funcionamiento del fondo del juego, verificar las colisiones, harán las animaciones al momento de morir etc., y al final quitaremos el evento de esta función para que el juego no se esté reiniciando todo el tiempo.

Función de la gravedad

```
function caidaMario(event: Event): void {
    marioP_mc.y += gravedad; //Aumentamos al personaje en eje de Y
    marioC_mc.y += gravedad; //Aumentamos al personaje en eje de Y
    gravedad += movimiento; //Aumentamos la gravedad
}
```

Esta función sirve para la gravedad y se llama "caidaMario" lo que se hace aquí es ir incrementando las coordenadas del eje de las y del personaje para que simule la gravedad, así como también incrementar la gravedad en un momento dado sumándole el movimiento.

Función para verificar cuando toca la lava

```
function apagarMario(event: Event) {
    if (marioP_mc.hitTestObject(lava_mc) && colisionLava == false) {
        //80 //38
        marioP_mc.visible = false;
        marioC_mc.visible = true;
        marioC_mc.y = lava_mc.y;
        colisionLava = true;
        terminar = true;
        //Desactivamos todas las demás funciones
        removeEventListener(Event.ENTER_FRAME, platMario1);
        removeEventListener(Event.ENTER_FRAME, platMario2);
        removeEventListener(Event.ENTER_FRAME, platMario3);
        removeEventListener(Event.ENTER_FRAME, fondoMario);
        removeEventListener(Event.ENTER_FRAME, colisionMario);
        removeEventListener(Event.ENTER_FRAME, precipicioMario);
        removeEventListener(Event.ENTER_FRAME, caidaMario);
        removeEventListener(KeyboardEvent.KEY_DOWN, pressMario);
        removeEventListener(KeyboardEvent.KEY_UP, soltarMario);
        timer.stop();
        distancia.stop();
    }

    if (colisionLava == true) {
        marioC_mc.y += 1;
        if (marioC_mc.y > 580) {
            //guardarMario();
            trace("Terminó");
            guardarMario();
            removeEventListener(Event.ENTER_FRAME, colisionMario);
            removeEventListener(Event.ENTER_FRAME, caidaMario);
            removeEventListener(Event.ENTER_FRAME, lineaMario); //Desactivamos el choque con la linea
            removeEventListener(Event.ENTER_FRAME, apagarMario); //Con esto removeremos del frame al circulo
            gotoAndStop(8);
        }
    }
}
```

Esta función nos sirve para verificar cuando el jugador toca la lava que es la que determina cuando se termina tu turno, es decir, cuando pierdes, utilizamos la función "hitTestObject" que es la que verifica cuando nuestro personaje (marioP_mc) está sobre la lava (lava_mc), cuando esto pasa ocultamos marioP_mc para que deje de moverse nuestro personaje, ponemos en verdadero nuestra variable "colisionLava" y "terminar". Posteriormente removemos los eventos del movimiento de las plataformas, el fondo, las colisiones, y la caída que es la gravedad, removemos los eventos de la barra espaciadora para saltar y paramos los timers del tiempo y la distancia.

Al momento de poner en verdadero la variable de la colisión inicia la condición que aumenta la coordenada en y +1 de nuestro personaje para simular que cae lentamente. Finalmente mandamos llamar a la función "guardarMario" para guardar nuestros datos, y mandamos al fotograma 8 donde se indica al jugador que ha terminado su turno.

Función para guardar los datos obtenidos por el jugador

```
function guardarMario() {
    channel1.stop(); //Detenemos musica de castle
    if (finalJ == false) { //si final j es falso
        player1.push(distancia_txtd.text); //Guardamos distancia en 2
        player1.push(cantPreci); //guardamos los presipicios en 3
        player1.push(tiempo_txtd.text); //guardamos el tiempo en 4
        presionado = false;
        switch (player2[1]) {
            case 0:
                channel = musica.play(); //Reproducimos musica de inicio
                channel1.stop(); //Detenemos musica de castle
                gotoAndStop(4);
                break;

            case 1:
                channel = musica.play(); //Reproducimos musica de inicio
                channel1.stop(); //Detenemos musica de castle
                gotoAndStop(5);
                break;

            case 2:
                channel = musica.play(); //Reproducimos musica de inicio
                channel1.stop(); //Detenemos musica de castle
                gotoAndStop(6);
                break;

            case 3:
                channel = musica.play(); //Reproducimos musica de inicio
                channel1.stop(); //Detenemos musica de castle
                gotoAndStop(7);
                break;
        }
        finalJ = true;
    } else {
        channel = musica.play(); //Reproducimos musica de inicio
        channel1.stop(); //Detenemos musica de castle
        player2.push(distancia_txtd.text); //Guardamos distancia en 2
        player2.push(cantPreci); //guardamos distancia en 3
        player2.push(tiempo_txtd.text); //guardamos distancia en 4
        gotoAndStop(8);
    }
```

Al momento de ingresar a esta función paramos la música del castillo y se verifica de qué jugador son los datos que se almacenaran, para ellos pregunta si la variable finalJ es verdadero, en caso de que lo sea entonces estamos hablando de que es el jugador final, es decir, el segundo jugador. Una vez que valida que jugador es entonces se vacían los datos que están en los txt de las distancias, los precipicios que saltó el jugador, el tiempo que hizo. Por último, solo vaciamos los campos para que estén en ceros para el siguiente jugador.

Función para mover el fondo

```
//Movimiento del fondo
function fondoMario(event: Event) {
    fondo1_mc.x -= 4; //Fondo1 recorremos hacia atras
    fondo2_mc.x -= 4; //Fondo2 recorrecmos hacia atras
    if (fondo1_mc.x <= -700) { //Cuando el fondo 1
        fondo1_mc.x = 700; //salga que vuelva a entrar
    }

    if (fondo2_mc.x <= -700) { //Cuando el fondo 2
        fondo2_mc.x = 700; //salga que vuelva a entrar
    }
}
```

Esta función realiza el movimiento del fondo, lo que hacemos es ir disminuyendo las coordenadas de nuestros dos movieClips en x para crear el efecto de que estamos avanzando y que cuando la coordenada x del fondo 1 sea menor o igual a -700 vuelva a entrar a la derecha de la ventana. Cuando la coordenada en x del fondo2 también sea menor o igual a -700 entonces también vuelva a entrar.

Acomodar Personaje

```
function acomodarMario(event: Event): void {
    if (marioP_mc.hitTestObject(plataforma1_mc)) { //Cuando choque con la plataforma 1
        marioP_mc.y = plataforma1_mc.y + 0.5; //Hacemos que se suba a la plataforma
        marioC_mc.y = plataforma1_mc.y + 0.5; //Hacemos que se suba a la plataforma
        gravedad = 2; //Gravedad la volvemos 2
        preci = false;
    }

    if (marioP_mc.hitTestObject(plataforma2_mc)) { //Cuando choque con la plataforma 2
        marioP_mc.y = plataforma2_mc.y + 0.5; //Hacemos que se suba a la plataforma
        marioC_mc.y = plataforma2_mc.y + 0.5; //Hacemos que se suba a la plataforma
        gravedad = 2; //Gravedad la volvemos 2
        preci = false;
    }

    if (marioP_mc.hitTestObject(plataforma3_mc)) { //Cuando choque con la plataforma 3
        marioP_mc.y = plataforma3_mc.y + 0.5; //Hacemos que se suba a la plataforma
        marioC_mc.y = plataforma3_mc.y + 0.5; //Hacemos que se suba a la plataforma
        gravedad = 2; //Gravedad la volvemos 2
        preci = false;
    }
    stage.removeEventListener(Event.ENTER_FRAME, acomodarMario); //Activamos el acomodo
}
```

En esta función lo que hacemos es que cuando se choque con una plataforma se acomodará el personaje en la línea de arriba de la plataforma, esta se ejecuta cada vez que se detecta una colisión en la función de ColisionMario.

Colisión con las plataformas

```
function colisionMario(event: Event): void {
    //Cuando choque con una de las 3 platafomas
    if ((marioP_mc.hitTestObject(plataforma1_mc) && marioP_mc.x + 50 > plataforma1_mc.x && marioP_mc.x < plataforma1_mc.x + 300)
        || (marioP_mc.hitTestObject(plataforma2_mc) && marioP_mc.x + 50 > plataforma2_mc.x && marioP_mc.x < plataforma2_mc.x + 300)
    || (marioP_mc.hitTestObject(plataforma3_mc) && marioP_mc.x + 50 > plataforma3_mc.x && marioP_mc.x < plataforma3_mc.x + 300)) { //Cuando c
        stage.removeEventListener(Event.ENTER_FRAME, caidaMario); //Desactivamos la caida en gravedad
        banderal = true;
        if (acomodo == false) {
            acomodo = true;
            preci = false;
            stage.addEventListener(Event.ENTER_FRAME, acomodarMario); //Activamos el acomodo
            //stage.removeEventListener(Event.ENTER_FRAME, lineaMario); //Desctivamos el acomodo con la linea
        }
    } else { //Cuando ya no se detecte que esta chocando con la base
        stage.addEventListener(Event.ENTER_FRAME, caidaMario); //Activamos la caida
        //stage.addEventListener(Event.ENTER_FRAME, lineaMario); //Activamos el acomodo con la linea
        acomodo = false;
    }
}
```

Cuando se detecte que el personaje haya chocado con una de las tres plataformas pero que este entre sus rangos de los lados laterales, es decir que se detecte una colisión por arriba de la plataforma.

Cuando se detecte se retira la función de caída para que el personaje no caiga más, y se verifica para que solo una vez ejecute la función de acomodo.

Cuando el programa ya no esté detectando la colisión este se activará la caída del personaje.

Colisión con líneas al lado de las plataformas

```
function lineaMario(event: Event): void {
    //Cuando choque con una de las lineas que se mueven al lado de las plataformas
    //Desactivaremos que vuelvan a chocar y alinearse
    if (marioP_mc.hitTestObject(linea) && marioP_mc.y > plataforma1_mc.y) {
        marioP_mc.x = linea.x - 50; //Moveremos al personaje hacia atras
        marioC_mc.x = linea.x - 50; //Moveremos al personaje hacia atras
        stage.removeEventListener(Event.ENTER_FRAME, colisionMario); //Desactivamos la caida en las plataformas
    }

    if (marioP_mc.hitTestObject(linea2) && marioP_mc.y > plataforma2_mc.y) {
        marioP_mc.x = linea2.x - 50; //Moveremos al personaje hacia atras
        marioC_mc.x = linea2.x - 50; //Moveremos al personaje hacia atras
        stage.removeEventListener(Event.ENTER_FRAME, colisionMario); //Desactivamos la caida en las plataformas
    }

    if (marioP_mc.hitTestObject(linea3) && marioP_mc.y > plataforma3_mc.y) {
        marioP_mc.x = linea3.x - 50; //Moveremos al personaje hacia atras
        marioC_mc.x = linea3.x - 50; //Moveremos al personaje hacia atras
        stage.removeEventListener(Event.ENTER_FRAME, colisionMario); //Desactivamos la caida en las plataformas
    }
}
```

Se encuentra una línea muy delgada al lado izquierdo de cada plataforma, tiene una diferencia de un pixel.

En esta función lo que hace es que nunca más se detecte una colisión con las plataformas porque ya se había detectado antes una colisión con la línea.

Cuando se halla detectado la colisión con la línea el personaje se moverá hacia la izquierda hasta que toque la lava que esta al fondo del juego, ya una vez tocada la lava el juego detectará que la toco y se ejecuta otra función.

Movimiento de Plataformas

```
//Movimiento de la plataforma
function platMariol(event: Event) {
    //Velocidad a la que se va a mover por pixeles
    plataformal_mc.x -= velocidad;
    linea.x -= velocidad;

    if (plataformal_mc.x < -100) { //Si la plataforma es -100 en eje X
        if (terminar == false) {
            stage.addEventListener(Event.ENTER_FRAME, platMario3); //Activamos la función de plataforma
        }
    }
    //mandar imprimir sentencia funcion ejecutar
    //trace (circle_mc.x);
    if (plataformal_mc.x < -300) { //preguntando si paso el circulo al escenario
        if (terminar == false) {
            plataformal_mc.x = 700; //Reiniciamos al punto de inicio de la plataforma
            plataformal_mc.y = 330 + (Math.random() * 100); //La posición Y la ponemos aleatoria
            linea.y = plataformal_mc.y + 12; //Igualamos la posición con la linea
            linea.x = 700; //Reiniciamos al punto de inicio de la linea
            stage.removeEventListener(Event.ENTER_FRAME, platMariol); //Removemos el evento del mismo
        }
    }
}
```

Es el movimiento de la plataforma número 1, lo que hace esta plataforma es que se moverá con una velocidad inicial, que después se irá aumentando.

Se tiene una comparación de que cuando la plataforma se encuentre con 100 pixeles fuera del escenario se activará el movimiento de la plataforma 3, esto solo se realiza una sola vez.

Después hay otra comparación que cuando la plataforma abandonó el escenario se le dará de inicio al final del escenario (lado derecho) con una altura aleatoria y la línea que va al lado de ella también se le asigna la misma altura y la misma posición y se desactiva la misma función. La cual se volverá activar en el recorrido de la plataforma 2. En las siguientes funciones de las demás plataformas (en total 3) es la misma lógica solo que mandan llamar a su siguiente plataforma).

En la función de platMario2 ahí es lo mismo solo cuando la plataforma 2 va -100 se ejecuta la plataforma 1 y cuando sale se elimina su función.

En la función de platMario3 ahí es lo mismo solo cuando la plataforma 3 va -100 se ejecuta la plataforma 2 y cuando se sale elimina su función.

```
function platMario2(event: Event) {
    //velocida a la que se va a mover por pixeles
    plataforma2_mc.x -= velocidad;
    linea2.x -= velocidad;

    if (plataforma2_mc.x < -100) { //si la plataforma es menor a 100 en eje x
        if (terminar == false) {
            stage.addEventListener(Event.ENTER_FRAME, platMario1); //Activamos la función de plataforma
        }
    }

    if (plataforma2_mc.x < -300) { //si la plataforma es menor a 300 ejecutamos
        if (terminar == false) {
            plataforma2_mc.x = 700; //Reiniciamos al punto de inicio de la plataforma
            plataforma2_mc.y = 330 + (Math.random() * 100); //La posición Y la ponemos aleatoria
            linea2.y = plataforma2_mc.y + 12; //Igualamos la posición con la linea
            linea2.x = 700; //Reiniciamos al punto de inicio de la linea
            stage.removeEventListener(Event.ENTER_FRAME, platMario2); //Removemos el evento del mismo
        }
    }
}
```

```
function platMario3(event: Event) {
    //la velocidad a la que se va a mover por pixeles
    plataforma3_mc.x -= velocidad;
    linea3.x -= velocidad;

    if (plataforma3_mc.x < -100) { //si la plataforma es menor a 100 en eje x
        if (terminar == false) {
            stage.addEventListener(Event.ENTER_FRAME, platMario2); //Activamos la función de plataformastaç
        }
    }

    if (plataforma3_mc.x < -300) { //si la plataforma es menor a 300 ejecutamos
        if (terminar == false) {
            plataforma3_mc.x = 700; //Reiniciamos al punto de inicio de la plataforma
            plataforma3_mc.y = 330 + (Math.random() * 100); //La posición Y la ponemos aleatoria
            linea3.y = plataforma3_mc.y + 12; //Igualamos la posición con la linea
            linea3.x = 700; //Reiniciamos al punto de inicio de la linea
            stage.removeEventListener(Event.ENTER_FRAME, platMario3); //Removemos el evento del mismo
        }
    }
}
```

Código del último frame

Mostrar resultados finales

```
player1N_txtd.text = player1[0]; //Nombre
player1Pre_txtd.text = player1[2]; //Precipicios
player1Me_txtd.text = player1[3]; //Metros
player1Ti_txtd.text = player1[4]; //Tiempo

player2N_txtd.text = player2[0]; //Nombre
player2Pre_txtd.text = player2[2]; //Precipicios
player2Me_txtd.text = player2[3]; //Metros
player2Ti_txtd.text = player2[4]; //Tiempo

if (player1[2]>player2[2]){
    ganador_mc.x = 291;
    //var tween: Tween = new Tween(ganador_mc, "y", Bounce.easeIn, 500, 145, 3, true);
}else{
    ganador_mc.x = 626;
    //var tween1: Tween = new Tween(ganador_mc, "y", Bounce.easeIn, 500, 145, 3, true);
}
```

En este último frame se cambia la música, mostramos los resultados del jugador en las casillas: player1N_txt (nombre), player1Pre_txtd (precipicios), player1Me_txtd (distancia en m y centímetros), player1Ti_txtd (el tiempo), de igual se hace esta acción para el segundo jugador.

Existe una condición que verifica cuál de los dos jugadores brinco mayor cantidad de precipicios, dependiendo de cuál haya sido el jugador que obtuvo mejor puntaje se muestra una bandera (gandor_mc) a un lado de sus datos obtenidos.

Guardar datos en archivo .txt

```
var guardado: FileReference = new FileReference();

var textF: String = "";

textF = "JUGADOR 1" + "\n"; //Guardamos en la variable mas un salto de linea
for (i = 0; i < player1.length; i++) {
    if (i != 1) {
        textF += player1[i] + "\n"; //Guardamos lo que tenga el array jugador1 pero con un salto de linea
    }
}

textF += "\n" + "JUGADOR 2" + "\n"; //Guardamos en la variable mas un salto de linea
for (i = 0; i < player2.length; i++) {
    if (i != 1) {
        textF += player2[i] + "\n"; //Guardamos lo que tenga el array jugador2 pero con un salto de linea
    }
}

guardado.save(textF, "Resultados.txt"); //Para guardar al final
//guardado.save(jugador2, "Resultados.txt");//Para guardar al final
```

Crearemos una variable string para guardar texto con espacio para después guardarlos en el archivo.

Creamos 2 ciclos for en los cuales un es para guardar cada información del player 1 con espacios, al terminar agregamos un espacio más para guardar el player 2. Utilizaremos la variable de Dato de referencia para guardar como "Resultados.txt".

Todos los demás Frames de personajes solo cambian el nombre ejemplo:

*Para Mario:

colsionMario

*Para Luigi:

colisionLuigi

*Para Toad:

colisionToad

*Para Peach:

colisionPeach

De este modo no ocurrirá ningún error en los personajes.

Conclusiones

Al momento de trabajar con el juego, formando una bina se obtienen más putos de vista. Se propusieron diferentes formas de realizar lo que se nos pedía, ya sea al momento de crear los algoritmos o de los diseños y acomodo de los frames para una mejor experiencia para el usuario así como diferentes formas de trabajar. En este proyecto utilizamos todos los conocimientos adquiridos a lo largo de la primera unidad, ya que se agregaron cosas básicas como los tweens, los botones, textos, clips de películas etc. Un proyecto de este tipo es una muy buena práctica para tener otras maneras de usar la programación.