UNIVERSIDAD POLITÉCNICA de VICTORIA

"Construyendo Conciencia"

SERPIENTES Y ESCALERAS

MANUAL TECNICO
HERRAMIENTAS MULTIMEDIA
UNIVERSIDAD POLITECNICA DE VICTORIA

OSCAR FRANCISCO FLORES GALLEGOS
SEBASTIAN HERNANDEZ CEPEDA

1. INDICE

2.-INTRODUCCION

Con el desarrollo de esta práctica pudimos utilizar más fácilmente nuestra lógica al momento de resolver algún problema y poder plasmarlo por medio de código con las enseñanzas que hemos obtenido durante las clases y conocimiento por cuenta propia ya que ha sido un reto de nuestro nivel y que si nos llevó mucho tiempo poder concluirlo, para esto tuvimos que implementar un poco de todo lo que hemos venido utilizando a lo largo de este curso con las practicas que hemos hecho y creo que esto es una mezcla de todo lo que hemos visto.

3. DESARROLLO

Para el desarrollo de este juego del memorama se hicieron aproximadamente 1250 líneas de código entre todos los frames a continuación se explicara paso a paso todas la funciones, variables, condiciones para así llegar a un mejor entendimiento de este código.

1.-PORTADA (1)

En el primer frame es la portada con sus respectivas librerías que va a exportar sus botones con su función y los tweens del texto, botón e imagen.

```
import flash.events.MouseEvent;
import fl.transitions.Tween;
import flash.events.Event;
import fl.transitions.TweenEvent;
import fl.transitions.easing.*;
import flash.utils.Timer;
import flash.events.TimerEvent;

stop();

//Boton para ir al sigueinte frame, y ver las reglas del juego
function Inicio(event:MouseEvent):void{
    gotoAndStop(5);
}
inicio_btn.addEventListener(MouseEvent.CLICK, Inicio);


var EaseOut:Tween = new Tween(serpientes_mc,"x",Elastic.easeOut,-216, 132.25, 3, true);
var EaseOut1:Tween = new Tween(uni_mc,"x",Elastic.easeOut,-216,48.75, 3, true);
var EaseOut2:Tween = new Tween(nombre1_mc,"x",Elastic.easeOut,-216,99.50, 3, true);
var EaseOut3:Tween = new Tween(nombre2_mc,"x",Elastic.easeOut,-216,119.30, 3, true);
var EaseOut4:Tween = new Tween(fecha_mc,"x",Elastic.easeOut,-216,172.10, 3, true);
var EaseOut5:Tween = new Tween(inicio_btn,"x",Elastic.easeOut,-216,270.85, 3, true);
var EaseOut6:Tween = new Tween(serpiente_mc,"x",Elastic.easeOut,-216,390.05, 3, true);


```

2.-REGLAS DE JUEGO (2)

En este frame simplemente va más que nada texto y lo que se implementa en el código son las funciones de los botones indicando a cual frame te llevara después de hacer click en el.

```
import flash.events.MouseEvent;

stop();

//Boton para ir al frame y elegir los jugadores, y ver las reglas del juego
function inicio(event:MouseEvent):void{
    gotoAndStop(2);
}
inicio_btn.addEventListener(MouseEvent.CLICK, inicio);

//Funcion para volver a las reglas
function volver(event:MouseEvent):void{
    gotoAndStop(1);
}
volver3_btn.addEventListener(MouseEvent.CLICK, volver);
```

3.-MODO DE JUEGO (3)

En este frame primeramente indica las librerías que tendrá que importar, posteriormente se hace la declaración de las variables que se usaran y también se hacen invisibles unas cajas de texto y botones hasta que estén llenados los datos correctamente.

En esta sección se hacen las funciones para los modo de juego de 2 y 3 jugadores

```
import flash.events.MouseEvent;
import fl.controls.ColorPicker;
import fl.events.ColorPickerEvent;
import flash.geom.ColorTransform;

stop();

//Declaracion de varaibles
var jugadores:Array = [];
var fichas:Array = [];
var limiteJ:int;
var n:int = -1;

//Ocultar los controles de ingreso de jugadores hasta que se seleccione un modo
nombre_In.visible = false;
guardar_btn.visible = false;
juego_btn.visible = false;
nombre_txt.visible = false;
```

empezamos con la primer función que es de un solo jugador que esta no es válida porque se juega a partir de 2 jugadores en adelante, también se declara el botón con su respectiva función. Esto se hace de igual manera en las funciones para dos y tres jugadores.

```
//Funciones de los botones de seleccion de numero de jugadores
//Un jugador
function UnJugador(event:MouseEvent):void{
	limiteJ = 1;
	nombre_In.visible = true;
	guardar_btn.visible = true;
	nombre_txt.visible = true;
	color.visible = true;
	color_txt.visible = true;
	n = 0;
}
unJugador_btn.addEventListener(MouseEvent.CLICK, UnJugador);

//Dos jugadores
function DosJugadores(event:MouseEvent):void{
	limiteJ = 2;
	nombre_In.visible = true;
	guardar_btn.visible = true;
	nombre_txt.visible = true;
	color.visible = true;
	color_txt.visible = true;
	n = 0;
}
dosJugadores_btn.addEventListener(MouseEvent.CLICK, DosJugadores);

//Tres jugadores
function TresJugadores(event:MouseEvent):void{
	limiteJ = 3;
	nombre_In.visible = true;
	guardar_btn.visible = true;
	nombre_txt.visible = true;
	color.visible = true;
	color_txt.visible = true;
	n = 0;
}
tresJugadores_btn.addEventListener(MouseEvent.CLICK, TresJugadores);
```

Después de las funciones del número de jugadores procedemos a declarar otra función que esta será para el registro de los jugadores, primeramente se valida que no este vacío el campo si no pedirá que se ingrese un nombre. Después se guardara el nombre y también el color seleccionado de la ficha ya que esta implementado un color picker y posteriormente se muestra el botón de iniciar ya que se haya llenado los campos solicitados para poder continuar. Al momento de guardar el nombre también se verifica que no se repita ningún nombre.

```
//Funcion para el registro de jugadores
function RegistroJugadores(event:MouseEvent):void{
    //Valdiar que no este vacio
    if(nombre_In.length == 0){
        alerta_Out.text = "Ingrese un nombre";
    }
    else{
        //Guardar el nombre y color del primer jugador
        if(n == 0){
            jugadores.push(nombre_In.text);
            ficha1.transform.colorTransform = colorJ1;
            fichas.push(ficha1);
            ficha1.visible = true;
            nombre_In.text = "";
        }

        //Revisar que no se repitan los nombres
        if(jugadores.length >= 1){
            for(var g:int = 0; g < jugadores.length; g++){
                if(jugadores[g] == nombre_In.text){
                    repN = 1;
                }
            }

            if(n == 1 && repN == 0){
                alerta_Out.text = "";

                ficha2.transform.colorTransform = colorJ2;
                fichas.push(ficha2);
                ficha2.visible = true;

                jugadores.push(nombre_In.text);;
                repN = 0;

                nombre_In.text = "";
            }
```

Ya para finalizar con este frame se declara primeramente el botón de guardar que es el que guardara los datos en este caso los nombres registrados en un array, también una función para iniciar el juego indicando que por medio del botón juego iniciara pasando al siguiente frame, de igual manera con el botón de volver que te regresa a las reglas del juego.

```
    //Cambiar a la sig ficha
    n++;
    //Reiniciar los indicadores de repeticion
    repN = 0;
}
guardar_btn.addEventListener(MouseEvent.CLICK, RegistroJugadores);

//Funcion para iniciar el juego
function IniciarJuego(event:MouseEvent):void{
    nextFrame();
}
juego_btn.addEventListener(MouseEvent.CLICK, IniciarJuego);

//Funcion para volver a las reglas
function Volver2(event:MouseEvent):void{
    gotoAndStop(5);
}
volver3_btn.addEventListener(MouseEvent.CLICK, Volver2);
```

En lo último de este frame tenemos lo que es el color picker que es para dar un toque de personalización a tus fichas eligiendo el color que el usuario desee.

Primeramente declaramos una variable para después llamar con la función color de esta manera se pondrán el número máximo de fichas para los jugadores que en este caso son un máximo de 3 por lo tanto serán tres ficha y posteriormente para guardar dicho color que el usuario selecciono, se hará sobre una variable especial para cada jugador, así a la hora de hacer el guardado se guardara el color junto con el nombre.

```
//ELEGIR COLOR CON COLOR PICKER
var mycolor:ColorTransform = new ColorTransform();
var colorJ1:ColorTransform = new ColorTransform();
var colorJ2:ColorTransform = new ColorTransform();
var colorJ3:ColorTransform = new ColorTransform();

color.addEventListener(ColorPickerEvent.CHANGE, colorChanger);

function colorChanger(event:ColorPickerEvent):void{
»   mycolor.color = color.selectedColor;
»   if(n == 0){
»   »   colorJ1 = mycolor;
»   }
»   else if(n == 1){
»   »   colorJ2 = mycolor;
»   }
»   else{
»   »   if(n == 2){
»   »   »   colorJ3 = mycolor;
»   »   }
»   }
}
```

4.-JUEGO (4)

Lo primero que tenemos en el código del juego es un array en donde tenemos los nombres de las instancias de las posiciones predeterminadas colocadas dentro del tablero, esto es para tener una trayectoria ya definida para las fichas. Seguido están los arrays donde se almacenara el acumulador para avanzar por el tablero de cada jugador y las variables que se utilizan para la función del dado. Seguido esta las líneas de código con la que se realiza el efecto de que las casillas del nombre, el tiempo y el número de casilla se hagan invisibles y después visibles al terminar el timer, que inicia a la vez que un timer que esta dentro del cuaderno.

```
//Declaracion de variables
//Arrays
var arrayPosiciones:Array = [tab.ub1, tab.ub2, tab.ub3, tab.ub4, tab.ub5, tab.ub6, tab.ub7, tab.ub8, tab.ub9, tab.ub10, tab.ub11, tab.ub12, tab.ub13, tab.ub14
    tab.ub15, tab.ub16, tab.ub17, tab.ub18, tab.ub19, tab.ub20, tab.ub21, tab.ub22, tab.ub23, tab.ub24, tab.ub25, tab.ub26, tab.ub27,
    tab.ub28, tab.ub29, tab.ub30, tab.ub31, tab.ub32, tab.ub33, tab.ub34, tab.ub35, tab.ub36, tab.ub37, tab.ub38, tab.ub39, tab.ub40,
    tab.ub41, tab.ub42, tab.ub43, tab.ub44, tab.ub45, tab.ub46, tab.ub47, tab.ub48, tab.ub49, tab.ub50, tab.ub51, tab.ub52, tab.ub53,
    tab.ub54, tab.ub55, tab.ub56, tab.ub57, tab.ub58, tab.ub59, tab.ub60, tab.ub61, tab.ub62, tab.ub63, tab.ub64, tab.ub65, tab.ub66,
    tab.ub67, tab.ub68, tab.ub69, tab.ub70, tab.ub71, tab.ub72, tab.ub73, tab.ub74, tab.ub75, tab.ub76, tab.ub77, tab.ub78, tab.ub79,
    tab.ub80, tab.ub81, tab.ub82, tab.ub83, tab.ub84, tab.ub85, tab.ub86, tab.ub87, tab.ub88, tab.ub89, tab.ub90, tab.ub91, tab.ub92,
    tab.ub93, tab.ub94, tab.ub95, tab.ub96, tab.ub97, tab.ub98, tab.ub99];
//Ordenamiento de las casillas en el tablero, como una posicion existente y como una posicion en el array
//Casilla, ub(casilla-1), posicion en el array(ub-1) o (casilla-2)
//Casilla 2, ub1, posicion: 0
var avance:Array = new Array();
//Contadores
var a:int;
var numRandom:int;
var controlNumrandom:int = 10;

//Efecto para que el tiempo, el nombre y la casilla aparezcan dentro del cuaderno
jugador_Out.visible = false;
tiempo_Out.visible = false;
casilla_Out.visible = false;

//Este timer coordina la operacion para que aparezcan las casillas
var timerLibro2:Timer = new Timer(1000, 1);
timerLibro2.start();
timerLibro2.addEventListener(TimerEvent.TIMER_COMPLETE, MostrarInfo);
//Se hacen visibles las casillas
function MostrarInfo(event:TimerEvent):void{
    jugador_Out.visible = true;
    tiempo_Out.visible = true;
    casilla_Out.visible = true;
}
```

Lo siguiente que tenemos es una función para colocar las fichas en la posición de inicio o casilla 1 y después mandamos a llamar otra función para cargar valores predeterminados en los arrays dependiendo del número de jugadores seleccionado, esto para evitar que ocurran errores por el valor undefined predeterminado de los arrays.

```
//Colocacion de las fichas en la posicion de inicioe inicializacion de valores en los arrays
function InicioJuego(event:Event):void{
    for(var b:int = 0; b < fichas.length; b++){
        fichas[b].x = tab.ubInicio.x;
        fichas[b].y = tab.ubInicio.y;
    }
    ValoresPredeterminados();
    stage.removeEventListener(Event.ENTER_FRAME, InicioJuego);
}
stage.addEventListener(Event.ENTER_FRAME, InicioJuego);

//Asignar valores a los arrays dependiendo de la cantidad de jugadores
function ValoresPredeterminados():void{
    if(limiteJ == 1){
        avance = [0];
        block = [0];
    }
    else if(limiteJ == 2){
        avance = [0, 0];
        block = [0, 0];
    }
    else{
        if(limiteJ == 3){
            avance = [0, 0, 0];
            block = [0, 0, 0];
        }
    }
}
```

Lo siguiente es la función del dado que se activa al hacer click al botón del dado, lo que se realiza es establecer un ciclo while en donde se generan 10 números aleatorios desde el 1 al 6, esto para aumentar la aleatoriedad. Una vez que termina el ciclo generamos el ultimo numero generado y lo guardamos en el array avance en el lugar correspondiente al jugador en turno, después se verifica que si el acumulador del jugador es mayor a 99, esto es para el rebote al final del tablero, que cuando es verdadero invoca la función Retroceso después de hacer que la ficha se mueva hasta el final. Como solo hasta que este casi al final puede ser mayor de 99, entrara en el else que invoca la función Casilla.

```
//Funcion del dado y para que se mueva la ficha
function Dado(event:MouseEvent):void{
    //Generar un numero aleatorio del 1 al 6
    a = 0;
    while(a < controlNumrandom){
        numRandom = Math.random() * (6) + 1;
        
        dado_Out.text = String(numRandom);
        a++;
    }

    //Sumarlo en el acumulador correspondiente
    avance[i] += numRandom;


    //Si es mayor a 100
    if(avance[i] > 99){
        trace("es mayor de 100");
        Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[98].x, 0.5, true);
        Mov1.addEventListener(TweenEvent.MOTION_FINISH, Retrocede);
    }
    else{
        Casilla();
    }
}
dado_btn.addEventListener(MouseEvent.CLICK, Dado);
```

La función Casilla es en donde se hacen los cálculos, condiciones y movimientos para que las fichas se muevan por el tablero siguiendo la trayectoria de posiciones. Lo primero es invocar la función Delay y calcular la distancia entre la ficha y cada una de las esquinas del tablero, lo cual es fundamental para el giro en las esquinas, después están las condiciones de movimiento. Cada movimiento está condicionado a esperar una el dato guardado en block de cada jugador, esto para tener un orden de movimiento y dependiendo de si se mueve horizontal o vertical. También se revisa si la ficha llego a una esquina o tiene que pasar por una, en el segundo caso se le “obliga” a llegar hasta la esquina y cuando termina el movimiento invoca la función Esquina correspondiente a donde este.

Las condiciones también nos ayudaran a controlar el cambio de block de cada jugador para evitar que se crucen.

Dentro de cada tween se toma el valor del acumulador dentro del array de cada jugador y se le resta 1 para que coincidan con las posiciones del array.

```
//Condiciones del movimiento a traves del tablero
function Casilla():void{
»   Delay();
»   //Calculo de la distancia que falta para cada esquina
»   distancia1 = 15 - avance[i];
»   distancia2 = 25 - avance[i];
»   distancia3 = 40 - avance[i];
»   distancia4 = 48 - avance[i];
»   distancia5 = 61 - avance[i];
»   distancia6 = 67 - avance[i];
»   distancia7 = 78 - avance[i];
»   distancia8 = 82 - avance[i];
»   distancia9 = 91 - avance[i];
»   distancia10 = 93 - avance[i];
»   trace("Jugador: " + i + " avance: " + avance[i] + " block: " + block[i]);
//-------------------------------------------------------------------------------------------------------------
»   //Movimiento sobre las casillas 1 a 16
»   if(block[i] == 0){
»   »   Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[avance[i] - 1].x, 0.5, true);
»   }
»   //Si cae en la casilla 16
»   if(avance[i] == 15){
»   »   block[i] = 1;
»   }
»   //Obligar a la ficha que pase por la casilla 16
»   if(avance[i] > 15 && block[i] == 0){
»   »   Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[14].x, 0.5, true);
»   »   Mov1.addEventListener(TweenEvent.MOTION_FINISH, Esquina1);
»   }
//-------------------------------------------------------------------------------------------------------------
```

En la función Casilla se encuentran algunos “casos especiales” originados por la longitud de los caminos al final del tablero, ya que los giros en ciertas posiciones y con ciertos números del dado provocan un mal funcionamiento, estos casos corrigen este defecto en el programa, los cuales hacen que las fichas lleguen a una esquina primero, e invocan la función Compensación correspondiente.

```
»   //Caso 0 de compensacion para llegar a la casilla 84 desde la 78
»   if(fichas[i].x == tab.ub77.x && fichas[i].y == tab.ub77.y){
»   »   if(numRandom == 6){
»   »   »   trace("entre");
»   »   »   Delay3();
»   »   »   Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[77].x, 0.5, true);
»   »   »   Mov1.addEventListener(TweenEvent.MOTION_FINISH, Compensacion0a);
»   »   }
»   }
```

```
»   //Casos de compensacion para el ultimo giro desde la casilla 89
»   //Caso 1: casilla 90
»   if(fichas[i].x == tab.ub89.x && fichas[i].y == tab.ub89.y)
»   »   if(numRandom > 4){
»   »   »   trace("entre");
»   »   »   Delay3();
»   »   »   Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[90].x, 0.5, true);
»   »   »   Mov1.addEventListener(TweenEvent.MOTION_FINISH, Compensacion1a);
»   »   }
»   //Caso 2: casilla 91
»   if(fichas[i].x == tab.ub90.x && fichas[i].y == tab.ub90.y)
»   »   if(numRandom > 3){
»   »   »   trace("entre");
»   »   »   Delay3();
»   »   »   Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[90].x, 0.5, true);
»   »   »   Mov1.addEventListener(TweenEvent.MOTION_FINISH, Compensacion2a);
»   »   }
»   //Caso 3: casilla 89
»   if(fichas[i].x == tab.ub88.x && fichas[i].y == tab.ub88.y)
»   »   if(numRandom == 6){
»   »   »   trace("entre");
»   »   »   Delay3();
»   »   »   Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[90].x, 0.5, true);
»   »   »   Mov1.addEventListener(TweenEvent.MOTION_FINISH, Compensacion3a);
»   »   }
```

Al final tenemos la última serie de condiciones que son cuando se encuentra en el último tramo del tablero. Lo primero es igual que en el resto de la función, se realiza un movimiento dependiendo del valor de block, lo segundo es cuando el acumulador de un jugador es igual a la última posición, 99, en donde si es verdadero hacemos que la ficha se mueva hasta esa casilla e invocan la función Fin.

```
//-------------------------------------------------------------------------------------------------------
»   //Movimiento sobre las casillas 95 a 100
»   if(block[i] == 10){
»   »   Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[avance[i] - 1].x, 0.5, true);
»   }
»   //Al llegar a 100
»   if(avance[i] == 99){
»   »   Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[98].x, 0.5, true);
»   »   Mov1.addEventListener(TweenEvent.MOTION_FINISH, Fin);
»   }
//-------------------------------------------------------------------------------------------------------
}
```

En las funciones Esquina se encuentran los cálculos y el movimiento cuando la ficha pasa por una esquina. Se invoca la función Delay2 y se toma la variable distancia correspondiente a la esquina, esta se multiplica por -1 para convertirla en su opuesto, luego se obtiene la diferencia entre la distancia y el número del dado. Ese dato que se obtiene se le vuelve a restar al número del dado. La primera resta es para obtener la distancia que avanzo cuando "obligamos" a la ficha a avanzar hasta la esquina, y la segunda es la distancia que falta recorrer dependiendo del número del dado. Después solo se pone en el tween sumando la última diferencia más la cantidad de casillas a la que está la esquina desde el inicio, y se cambia el valor de block.

```
//Funcion para realizar el giro en la esquina, casilla 16
function Esquina1(event:TweenEvent):void{
»   Delay2();
»   //Calculo de la distancia que se avanza cuando se pasa por una esquina
»   //Se mantiene la distancia en un valor positivo
»   distancia1 *= -1;
»   //Se obtiene la diferencia de lo que se va a recorrer en x
»   res = numRandom - distancia1;
»   //Se obtiene la diferencia de lo que se va a recorrer en y
»   res2 = numRandom - res;
»
»   //Se realiza el movimiento sumando lo que ya se debio de recorrer
»   Mov2 = new Tween(fichas[i], "y", None.easeOut, fichas[i].y, arrayPosiciones[res2 + 14].y, 0.5, true);
»   //Se incrementa el bloqueador de movimiento
»   block[i] = 1;
}
```

La función Retroceder realiza cálculos similares a las funciones Esquina, solo que con la diferencia es que el dato de la primera resta se hace sobre el acumulados y después sobre el número del dado. Con esto obtenemos lo que ya avanzo y lo que falta avanzar, para lo cual se pone en el tween la segunda resta más el total de posiciones, y es suma porque res2 siempre es negativo. Al final se acomoda el del acumulador.

```
//Funcion para que se regrese si el numero del dado es mayor a lo necesario para llegar al final
/*Se realizan practicamente los mismos calculos que en las funciones de las esquinas, solo cambian
el uso que se le da a los calculos*/
function Retrocede(event:TweenEvent):void{
    trace("como me pase, tengo que regresar");
    Delay2();
    res = avance[i] - 98;
    res2 = numRandom - res;
    Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[99 + res2].x, 0.5, true);
    avance[i] = avance[i] - res2;
}
```

A continuación están las funciones Comparación, que se invocan cuando ocurre un "caso especial", en la cuales se hace que la ficha llegue a la siguiente esquina de las que se le obligo a moverse y de ahí se mueva a la casilla correspondiente, y se cambia el valor de block, pero no del acumulador, ya que es el que marca cuando ocurre el caso especial.

```
//Funciones caso 0
//Movimientos para que la ficha se mueva de manera correcta en los casos especiales
function Compensacion0a(event:TweenEvent):void{
    trace("entre a 0a");
    Mov2 = new Tween(fichas[i], "y", None.easeOut, fichas[i].y, arrayPosiciones[81].y, 0.5, true);
    Mov2.addEventListener(TweenEvent.MOTION_FINISH, Compensacion0b);
}
function Compensacion0b(event:TweenEvent):void{
    trace("entre a 0b");
    Mov1 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, arrayPosiciones[82].x, 0.5, true);
    block[i] = 8;
}
```

Después se encuentran las funciones Delay, cada uno para un caso específico. La función Delay1 es para cuando ocurre un movimiento normal. La Delay2 se utiliza en el caso de los giros en las esquinas, y cuando una ficha se mueve en una escalera o serpiente para dar 1 segundo más para terminar el movimiento. La Delay3 es para los "casos especiales". En todos se esconde el botón del dado y cuando termina el timer, se invoca una función para volverlo hacer aparecer y cambiar el turno. En los Delay2 y Delay3 se quita el listener de los timer anteriores respectivamente para dar el efecto de que se suma más tiempo.

```
//Variables timer de los delays
var delay1:Timer;
var delay2:Timer;
var delay3:Timer;
//Delay para evitar que una pieza se mueve dos veces en un turno
function Delay():void{
    delay1 = new Timer(1000, 1);
    dado_btn.visible = false;
    delay1.start();
    delay1.addEventListener(TimerEvent.TIMER_COMPLETE, ReanudarDado);
}
function ReanudarDado(event:TimerEvent):void{
    dado_btn.visible = true;
    i++;
}
//Delay para los giros en las esquinas, escaleras y serpientes
function Delay2():void{
    delay1.removeEventListener(TimerEvent.TIMER_COMPLETE, ReanudarDado);
    delay2 = new Timer(1000, 1);
    dado_btn.visible = false;
    delay2.start();
    delay2.addEventListener(TimerEvent.TIMER_COMPLETE, ReanudarDado);
}
//Delay para los casos de compensacion
function Delay3():void{
    delay1.removeEventListener(TimerEvent.TIMER_COMPLETE, ReanudarDado);
    delay2.removeEventListener(TimerEvent.TIMER_COMPLETE, ReanudarDado);
    delay3 = new Timer(1000, (2.5));
    dado_btn.visible = false;
    delay3.start();
    delay3.addEventListener(TimerEvent.TIMER_COMPLETE, ReanudarDado);
}
```

En la siguiente función está el reloj del juego, el cual funciona con un timer que se le obliga a contar de manera indefinida hasta 9, ya que con las condiciones se consigue el efecto de que parezca un reloj más realista. En el caso de décimas de segundo y minuto, solo cuenta hasta 6.

```
//Inicio del timer
timer.start();
timer.addEventListener(TimerEvent.TIMER, Tiempo);

//Tiempo del juego
function Tiempo(event:TimerEvent):void{
	//Segundos
	segundos++;
	if(segundos > 9){
		segundos2++;
		segundos = 0;
	}
	if(segundos2 == 6){
		minutos++;
		segundos2 = 0;
	}
	//Minutos
	if(minutos > 9){
		minutos2++;
		minutos = 0;
	}
	if(minutos2 == 6){
		segundos = 0;
		segundos2 = 0;
		minutos = 0;
		minutos2 = 0;
	}
	//Imprimir tiempo de juego
	tiempo_Out.text = "" + minutos2 + minutos + ":" + segundos2 + segundos;
}
```

En esta función es donde ocurre la impresión del nombre de cada jugador y del número de casilla. También se encuentra una condición para que la variable que controla los turnos (i), se reinicie cuando ya se llegó al último jugador, y una condición para que en el nombre no se imprime undefined.

```
//Mostrar el nombre del jugador actual y asignar su turno
//Variable indicadora del turno
var i:int = 0;
function JugadorActual(event:Event):void{
	//Esta condicion es para la asignacion de turno
	if(i == limiteJ){
		i = 0;
	}

	//Esto es para que no imprima undefined
	if(jugadores[i] == undefined){
		jugador_Out.text = "";
	}
	else{
		//Imprime el nombre de manera correcta
		jugador_Out.text = String(jugadores[i]);
		casilla_Out.text = String(avance[i] + 1);
	}
}
stage.addEventListener(Event.ENTER_FRAME, JugadorActual);
```

Esta es la función de las Escaleras. Para cada una se revisa que una ficha este en la casilla de inicio de una escalera, si es así se invoca el Delay2 , se realiza un movimiento en diagonal hasta la casilla de destino, y se cambia el valor de block y del acumulador dependiendo del tramo del tablero en donde se encuentre la casilla destino.

```
//Condiciones de las Esxcaleras
/*
1)Se determina si la casilla correspondiente es una escalera
2)Si lo es, se mueve a la casilla correspondiente
3)Los valores del los arrays de movimiento se cambian respectivamente a donde se movio
*/
function Escaleras(event:Event):void{
»   if(fichas[i].x == tab.ub2.x && avance[i] == 2){
»   »   Delay2();
»   »   Mov3 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, tab.ub51.x, 0.5, true);
»   »   Mov4 = new Tween(fichas[i], "y", None.easeOut, fichas[i].y, tab.ub51.y, 0.5, true);
»   »   block[i] = 4;
»   »   avance[i] = 51;
»   }
»
```

Para las Serpientes es prácticamente lo mismo que las escaleras, con la única diferencia de que el movimiento será para retroceder.

```
stage.addEventListener(Event.ENTER_FRAME, Escaleras);

//Condiciones de las serpientes
//El proceso es igual al de las escaleras, solo cambia el movimiento
function Serpientes(event:Event):void{
»   if(fichas[i].x == tab.ub54.x && avance[i] == 54){
»   »   Delay2();
»   »   Mov3 = new Tween(fichas[i], "x", None.easeOut, fichas[i].x, tab.ub6.x, 0.5, true);
»   »   Mov4 = new Tween(fichas[i], "y", None.easeOut, fichas[i].y, tab.ub6.y, 0.5, true);
»   »   block[i] = 0;
»   »   avance[i] = 6;
»   }
```

Al final del código del frame está la función Fin, la cual al invocarse quiere decir de que una ficha llego hasta la última casilla, la meta, por lo cual se remueven los listener de las funciones que se invocan por el escenario y se detiene el timer del reloj, para después pasar al siguiente frame.

```
stage.addEventListener(Event.ENTER_FRAME, Serpientes);

//Funcion para terminar el juego cuando alguien llego a la casilla 100
function Fin(event:TweenEvent):void{
»   stage.removeEventListener(Event.ENTER_FRAME, JugadorActual);
»   stage.removeEventListener(Event.ENTER_FRAME, Serpientes);
»   stage.removeEventListener(Event.ENTER_FRAME, Escaleras);
»   timer.stop();
»   nextFrame();
}
```

5.- MOSTRAR RESULTADOS E IMPRIMIR A PDF (5)

Lo primero que encontramos en el frame 5 es la función del botón de volver para regresar al frame del modo de juego.

Después empieza la función del Puntaje que utiliza el método de burbuja el cual evaluara el avance que desempeño cada jugador en el tablero y los acomodara respectivamente de mayor a menor avance, para esto se toma en cuenta que también tiene que haber una organización de los nombres de los usuarios para que estos sean impresos junto con su respectivo puntaje en la partida.

```
import flash.events.Event;
import flash.utils.Timer;
import flash.events.TimerEvent;
import flash.events.MouseEvent;
import flash.net.FileReference;

stop();

//Funcion para volver a las reglas
function volverr(event:MouseEvent):void{
    gotoAndStop(2);
}
volver3_btn.addEventListener(MouseEvent.CLICK, volverr);

stage.addEventListener(Event.ENTER_FRAME, Puntaje);
function Puntaje(event:Event):void{
    //Ordenar los datos por mayor puntaje por metodo burbuja
    if(jugadores.length > 1){
        for(var m:int = 0; m < jugadores.length; m++){
            for(var w:int = 0; w < jugadores.length; w++){
                if(avance[w] < avance[w+1]){
                    //Ordenar puntajes
                    var tempP:Number = avance[w];
                    avance[w] = avance[w+1];
                    avance[w+1] = tempP;
                    //Ordenar nombres
                    var tempN:String= jugadores[w];
                    jugadores[w] = jugadores[w+1];
                    jugadores[w+1] = tempN;
                    
                    trace("Nombre: "+jugadores+" Puntaje: "+ avance);
                }
            }
        }
    }
    stage.removeEventListener(Event.ENTER_FRAME, Puntaje);
}
```

Se incluyen dos funciones más que facilitaran el proceso de impresión, la primera que es la función Mostrar evitara que se cicle la impresión de los datos. Y con un for se guardan los valores obtenidos de los arrays anteriores y así mismo se manda llamar la función ganador que es la que se muestra en el fotograma donde solo se podrá visualizar el nombre del usuario que logro llegar exactamente a la casilla cien.

Aquí se programa el botón para exportar en .txt los datos de los jugadores, se imprime primero el ganador y después se imprimen todos los demás jugadores según el orden de avance que se registró en el tablero

```
//Array para juntar todos los resultados en cadenas
var arrayFinal:Array = new Array;

//Funcion para imprimir los resultados
stage.addEventListener(Event.ENTER_FRAME, Mostrar);
function Mostrar(event:Event):void{
»   //Condicion para que no se cicle la impresion de los resultados
»   var active:Boolean = true;
»   if(active == true){
»   »   stage.removeEventListener(Event.ENTER_FRAME, Mostrar);
»   }
»   //Ciclo para obtener todos los valores guardados en los arrays y convertirlos en string
»   for(var f:int = 0; f < jugadores.length; f++){
»   »   »   arrayFinal.push(String(" " + (f + 1) + "° " + jugadores[f] + " " + avance[f] + 1));
»   »   }
»   //Impresion del mensaje del ganador
»   Ganador();
»   active = false;
»
}

//Mensaje de felicitaciones a la mayor puntuacion
function Ganador():void{
»   felicidades_Out.text = String("¡¡Felicidades a " + jugadores[0] + "!!");
}

//Funcion para exportar a .txt
function Exportar(event:MouseEvent){
»   var archivo:FileReference = new FileReference;
»   archivo.save(arrayFinal + "\r\n");
}
exportar_btn.addEventListener(MouseEvent.CLICK, Exportar);
```

4. CONCLUSION

Con la elaboración de este juego podemos concluir que hay un cierto grado de dificultad en la elaboración de este, pero que nada es imposible, nos llevó 2 semanas desarrollarlo pero sentimos que valió la pena, nunca nos imaginamos que sin saber programar, ya que estamos en segundo cuatrimestre, íbamos a poder desarrollar un juego que es básico lo sabemos pero por algo se empieza y esperamos más juegos y practicas divertidas y entretenidas como esta para nuestra buena preparación a lo largo de la carrera para llegar a ser unos buenos profesionistas el día de mañana.