Universidad Politécnica de Victoria

Manual Técnico.

Unidad ll.

Serpientes y Escaleras.

Alumnos:
Jesús Antonio Miravete Martínez (1730140).
Pedro Luis Espinoza Martínez (1730314).
Grupo: ITI 2-1.
Materia: Herramientas Multimedia.

Índice

Introducción.

En este manual técnico veremos como esta conformado el juego de “Serpientes y Escaleras”, solo son 4 fotogramas diferentes, en los cuales se encuentran:

1. Portada.
2. Selector de Jugadores.
3. Juego de “Serpientes y Escaleras”.
4. Resultados generales de los jugadores.

Observaremos todo el desarrollo del juego, veremos los diferentes metodos que se usaron, como también la forma en que se implementaron para poder hacer funcionar todos los fotogramas.

Este juego cuenta con lo siguiente:

1. 4 Fotogramas diferentes.
2. Cuenta con un multijugador de hasta 3 personas.
3. Cuenta con 100 posiciones en el tablero.
4. Cuenta con 3 diferentes avatares de jugadores.
5. Cuenta con animación en el dado, incluyendo interpolaciones de movimiento.
6. Para jugar sera necesario de al menos 2 jugadores.

FRAME 1

En el primer frame tenemos nuestra portada, en ella tenemos algunas imágenes que se mueven y un boton para entrar al juego.

```
1  //=============================================================================================
2  //Se importan las librerias necesarias para eventos del mouse, como tambien sobre los Tweens.
3  //=============================================================================================
4  import flash.events.MouseEvent;
5  import fl.transitions.*;
6  import fl.transitions.easing.*;
7  import fl.transitions.Tween;
8  stop();
9  var StrongEaseOutTodot:Tween= new Tween(todot_mc,"y",Strong.easeInOut,-1511,47.7,2,true);
10 var StrongEaseOutsig:Tween= new Tween(siguiente2_btn,"y",Strong.easeInOut,1511,650.7,2,true);
11 function EntrarJugadores(event: MouseEvent): void {
12     gotoAndStop(2);
13
14 }
15 //Asignamos la funcion "Entrar" al boton "entrar_btn".
16 siguiente2_btn.addEventListener(MouseEvent.CLICK, EntrarJugadores);
```

Lo primero que hacemos, en las lineas 4-7 es importar las librerias que necesitaremos para los tweens que haran que se muevan las imágenes, ademas de las librerias que contienen los eventos del mouse.

Ademas se puede ver como en la linea 8 colocamos un stop(); esto lo hacemos para que nuestro frame se mantenga en la portada al ejecutarse.

```
4  import flash.events.MouseEvent;
5  import fl.transitions.*;
6  import fl.transitions.easing.*;
7  import fl.transitions.Tween;
8  stop();
```

Despues tenemos nuestros tweens. El primero es el que hace que todos los elementos excepto el boton se muevan. El segundo se encarga de animar solamente el boton:

```
9  var StrongEaseOutTodot:Tween= new Tween(todot_mc,"y",Strong.easeInOut,-1511,47.7,2,true);
10 var StrongEaseOutsig:Tween= new Tween(siguiente2_btn,"y",Strong.easeInOut,1511,650.7,2,true);
```

Luego en la linea 11 tenemos nuestra funcion EntrarJugadores, esta funcion lo que hace es llevarnos al fotograma 2:

```
11 function EntrarJugadores(event: MouseEvent): void {
12     gotoAndStop(2);
13
14 }
15 //Asignamos la funcion "Entrar" al boton "entrar_btn".
16 siguiente2_btn.addEventListener(MouseEvent.CLICK, EntrarJugadores);
```

FRAME 2

En nuestro segundo frame es donde los usuarios ingresan sus nombres, ya sean 2 o 3 jugadores.

Lo primero que hacemos al igual que en el frame anterior es importar las librerías que necesitaremos, de igual manera colocamos un stop para que nuestra animación se mantenga en este frame. Además hacemos que el botón siguiente se mantenga invisible desde el inicio:

```
4   import flash.events.MouseEvent;
5   import fl.transitions.*;
6   import fl.transitions.easing.*;
7   import fl.transitions.Tween;
8   stop();
9   siguiente_btn.visible = false;
```

A continuación tenemos nuestra función entrar, esta será la encargada de llevarnos al frame del juego una vez que se presione el botón siguiente:

```
13  function Entrar(event: MouseEvent): void {
14      gotoAndStop(3);
15
16  }
17  //Asignamos la funcion "Entrar" al boton "entrar_btn".
18  siguiente_btn.addEventListener(MouseEvent.CLICK, Entrar);
19
```

Después declaramos las variables y Arrays que utilizaremos.

nombre: almacenara los nombres que se ingresen en el textinput.
contNombres: variable contadora que llevara el control de la cantidad de jugadores ingresados
aNombres: Array que almacenara los nombres validos
aFichas: Array en el que enseguida en la línea 27 le agregamos las 3 diferentes fichas de jugadores.

```
20  //declaracion de variables
21  var nombre: String = "";
22  var contNombres: int = 0;
23  //Declaracion de arrays
24  var aNombres: Array = new Array;
25  var aFichas:Array = new Array;
26  //Llenamos el array aFichas con las 3 imagenes que tenemos para las fichas de jugadores
27  aFichas=[ficha1,ficha2,ficha3];
```

Nota: ficha1, ficha2 y ficha2 son los nombres de instancias de las fichas, sin embargo estas no son las que se utilizaran en el tablero de juego.

Enseguida comenzamos con nuestra función ingresar, la cual se encargara de validar los nombres y guardarlos.

Lo primero que hacemos al entrar es que el mensaje de alerta se limpie, esto en caso de que haya aparecido anteriormente (línea 30).

Después tenemos un if, el cual se encargara de verificar si ya se ingresó el límite de 3 jugadores en base a la variable contNombres, la cual más adelante se incrementa.

Si esta condición se cumple entonces mandamos el mensaje de alerta ¡Limite de Jugadores alcanzado! Sin embargo, si no se cumple entonces entra a un nuevo if, el cual se encarga de validar que se haya ingresado un nombre, si no se ingresó nada entonces manda el mensaje de alerta Ingrese un nombre Valido(línea 35), si no entonces tomamos el nombre y lo guardamos en el Array(líneas 37 y 38), además incrementamos contNombres lo que indicara que se ingresó un número más, esto para que cuando se llegue a 3 ya no se puedan ingresar mas:

```
29  function ingresar(event: MouseEvent): void {
30      msjAlerta.text = "";//limpiamos la caja de texto por c
31      if (contNombres == 3) {//Comparamos si se llego al lim
32          msjAlerta.text="¡Limite de Jugadores alcanzado!";
33      } else {//si no entonces comparamos si la caja de text
34          if (nombre_txt.text == "") {//Si lo esta entonces
35              msjAlerta.text = "Ingrese un nombre Valido";
36          } else {//Pero si no esta vacio entonces tomamos l
37              nombre = nombre_txt.text;
38              aNombres.push(nombre);
39              contNombres++//Incrementamos el contador de ju
40          }
```

Aun dentro de la misma función, después del último if, tenemos un nuevo if, el cual verificara si ya se ingresaron 2 jugadores, si se cumple la condición entonces se hará visible el botón siguiente, esto ya que el juego es de mínimo 2 jugadores, puesto que no tendría sentido jugar uno solo;

```
if (contNombres == 2) {//Comparamos
    //Si se cumple entonces hacemos
    //Hacemos esto para que solo se
    siguiente_btn.visible = true;
}
```

Casi para finalizar con esta función tenemos un switch, el cual se encargara de saber cuándo se debe mostrar un nuevo nombre y su ficha, esto para que cada que se ingrese un nombre se vaya mostrando un “registro” de los nombres ingresados, junto con la ficha que se le asignara a cada jugador para que sepan cual les toca.

El switch solo tiene 3 casos, en cada uno lo que hace es mostrar un nombre y una ficha, la forma en que está hecha la función hará que cada que se ingrese un nuevo nombre el switch muestre ese nuevo nombre además de su ficha correspondiente, la ficha y el nombre se relacionan en base a sus posiciones en sus Arrays correspondientes, en las posiciones 0 y 1 en caso de 2 jugadores, y en las posiciones 0, 1 y 2 en caso de 3:

```
switch(contNombres){//En cada caso solo colocamos un nombre
    case 1:
        aFichas[0].x=327;
        aFichas[0].y=426;
        nombre1.text="1.- "+aNombres[0];
        break;
    case 2:
        aFichas[1].x=327;
        aFichas[1].y=526;
        nombre2.text="2.- "+aNombres[1];
        break;
    case 3:
        aFichas[2].x=327;
        aFichas[2].y=626;
        nombre3.text="3.- "+aNombres[2];
        break;
}
```

Para terminar con esta función lo único que resta es limpiar el textinput, para que al ingresar un nombre, la caja de texto quede lista para ingresar el siguiente nombre, sin tener que borrar el ingreso anterior manualmente:

```
65          nombre_txt.text="";//Limpiamos la caja de texto para in
66      }//Asignamos la funcion al boton ingresar
67      ingresar_btn.addEventListener(MouseEvent.CLICK,ingresar);
```

Ya fuera de la función anterior, hacemos invisibles una capa transparente, además de una imagen que contiene las reglas y el boton que cerraría las reglas.

Después creamos la función Reglas en la cual al presionar el cuadro con una pluma, el cual actúa como un botón, hará visibles los 3 elementos que se hicieron invisibles anteriormente además de darles una animación con tweens, esto lo hacemos para crear una “pantalla extra” la cual muestra las reglas del juego al jugador

```
71  trans_mc.visible=false;
72  reglas_mc.visible=false;
73  salirreg_btn.visible=false;
74  function Reglas (event:MouseEvent):void{
75      reglas_mc.visible=true;
76      trans_mc.visible=true;
77      salirreg_btn.visible=true;
78      var ReglasT:Tween= new Tween(reglas_mc,"y",Strong.easeInOut,-700,64.5,1,true);
79      var ReglasT2:Tween= new Tween(salirreg_btn,"y",Strong.easeInOut,-700,110.9,1,true);
80  }
81  reglas_btn.addEventListener(MouseEvent.CLICK,Reglas);
82
```

Para terminar con este frame, tenemos la función SalirReglas. Esta función se encarga de sacar del escenario la pantalla de reglas y el botón para salir de ellas, además de hacer invisible el fondo transparente. Esta función se le agrega al mismo botón al que se le agrega tween para salir, el cual se encuentra dentro de la pantalla de las reglas

```
83  function SalirReglas (event:MouseEvent):void{
84      trans_mc.visible=false;
85      var ReglasT3:Tween= new Tween(reglas_mc,"y",Strong.easeInOut,reglas_mc.y,-700,1,true);
86      var ReglasT4:Tween= new Tween(salirreg_btn,"y",Strong.easeInOut,salirreg_btn.y,-700,1,true);
87  }
88  salirreg_btn.addEventListener(MouseEvent.CLICK,SalirReglas);
```

FRAME 3

En el frame es donde se lleva a cabo todo el juego de serpientes y escaleras.

Para empezar importamos las librerías que necesitaremos, además hacemos un stop (); para mantenernos en este frame:

```
4   import flash.events.MouseEvent;
5   import fl.transitions.*;
6   import fl.transitions.easing.*;
7   import fl.transitions.Tween;
8   import flash.utils.Timer;
9   import flash.events.Event;
10  import flash.events.TimerEvent;
11  stop();
```

Después hacemos todos los tweens para cuando se entra al frame. Aplicamos uno a cada ficha de jugador, uno para el dado, uno para el tiempo y uno para el jugador de turno actual y el que seguirá después, además de los marcos para cada uno de los elementos anteriores.

También hacemos invisible el botón de resultados:

```
13  var StrongEaseOutTodo: Tween = new Tween(todo_mc, "y", Strong.easeInOut, -1511, 0, 2, true);
14  var StrongEaseOutF1: Tween = new Tween(ficha1t, "y", Strong.easeInOut, 1511, 681.4, 2, true);
15  var StrongEaseOutF2: Tween = new Tween(ficha2t, "y", Strong.easeInOut, 1511, 681.4, 2, true);
16  var StrongEaseOutF3: Tween = new Tween(ficha3t, "y", Strong.easeInOut, 1511, 681.4, 2, true);
17  var StrongEaseOutD: Tween = new Tween(dado_mc, "x", Strong.easeInOut, -1511, 120.9, 2, true);
18  var StrongEaseOutT: Tween = new Tween(tiempo_txt, "x", Strong.easeInOut, -1511, 94.7, 2, true);
19  var StrongEaseOutJa: Tween = new Tween(jactual_txt, "x", Strong.easeInOut, 1511, 1029.95, 2, true);
20  var StrongEaseOutJs: Tween = new Tween(jsiguiente_txt, "x", Strong.easeInOut, 1511, 1029.95, 2, true);
21  resultados_btn.visible = false;
```

Después creamos algunos Arrays y variables que necesitaremos.

aFichas: almacena las imágenes de las fichas que se usaran en el tablero(No son las mismas que se muestran al ingresar los nombres).
contFichas: Almacena 3 valores, los cuales llevaran un control de la posición en la que se encuentra cada ficha en todo momento.
tirada: Almacenara el número que genere el random.

contTurno: Variable contadora que ayudara a saber el turno del jugador actual, en otras palabras, nos ayudara a saber con qué ficha debemos trabajar.
Se puede ver cómo hacemos 3 veces el mismo push, esto ya que serán los 3 valores iniciales de las posiciones de las fichas.

```
23  var aFichasT: Array = new Array;
24  aFichasT = [fichalt, ficha2t, ficha3t];
25  var contFichas: Array = new Array;
26  contFichas.push(-1);
27  contFichas.push(-1);
28  contFichas.push(-1);
29  var tirada: int; //Variable que almacena
30  var contTurno: int = 0;
```

Ahora creamos la función tirarDado, la cual se encargara de generar el número random y otras cosas.

Lo primero que hacemos es asignarle a la variable tirada un número random entre el 1 y el 6(línea 34), después iniciamos el timerdado y mandamos al dado a su segundo frame lo cual hará que se inicie una pequeña animación del mismo.

Además al dado le removemos el evento, para que ya no se pueda presionar nuevamente hasta cierto momento:

```
33  function tirarDado(event: MouseEvent): void {
34      tirada = Math.random() * 6 + 1; //A la variable tirada le
35      //resultado_tirada.text=""+tirada;//Lo mostramos en el tex
36      //timer.start();
37      timerdado.start();
38      dado_mc.gotoAndPlay(2);
39      dado_mc.removeEventListener(MouseEvent.CLICK, tirarDado);
40
41  } //Asignamos la funcion al boton tirar
42  dado_mc.addEventListener(MouseEvent.CLICK, tirarDado);
```

Después declaramos el timer tirardado y las variables que usaremos junto a él.

Las variables tdado y contado serán las que se incrementaran cada que pase el tiempo determinado en el timer, pero contado será la que usaremos para nuestras operaciones:

```
46  var timerdado: Timer = new Timer(200, tdado++);
47  var tdado: int = 0;
48  var contdado: int = 0;
```

Una vez que tenemos nuestro timer creamos la función en la que lo usaremos, en este caso la función TimerDado.

Lo primero que hacemos en esta función es incrementar en 1 a la variable contado.

Después tenemos un if, en el que comparamos si contado vale 5, esto para saber cuándo ya paso un segundo, cuando esta condición se cumpla entonces reiniciamos el contador, iniciamos a timer (Timer que se explicara más adelante), y detenemos el timer del dado.

Además, dentro del mismo if tenemos un switch, el cual comparara el número generado aleatoriamente entre 1 y 6, y dependiendo del número que haya tocado se mandara a la imagen a un frame específico, los cuales contienen las diferentes caras del dado:

```
49  function TimerDado(event: TimerEvent): void { //funcion para el timer
50      contdado++;
51      if (contdado == 5) {
52          contdado = 0;
53          timer.start();
54          timerdado.stop();
55          switch (tirada) {
56              case 1:
57                  dado_mc.gotoAndStop(1);
58                  break;
59              case 2:
60                  dado_mc.gotoAndStop(11);
61                  break;
62              case 3:
63                  dado_mc.gotoAndStop(21);
64                  break;
65              case 4:
66                  dado_mc.gotoAndStop(31);
67                  break;
68              case 5:
69                  dado_mc.gotoAndStop(41);
70                  break;
71              case 6:
72                  dado_mc.gotoAndStop(51);
73                  break;
74          }
75      }
76  }
77  timerdado.addEventListener(TimerEvent.TIMER, TimerDado);
```

Ya fuera de la función del dado, lo que hicimos después fue declarar 2 arrays y 2 variables:

aX: Array que almacenara las posiciones en x de cada casilla.
aY: Array que almacenara las posiciones en y de cada casilla.
i y j: Variables contadoras que utilizaremos en los ciclos for.

```
80  //Declaracion de arrays que a
81  var aX: Array = new Array;
82  var aY: Array = new Array;
83  var i: int;
84  var j: int;
```

Ahora declaramos variables a las cuales les asignamos las posiciones iniciales en x y en y en base al tablero, lo hacemos utilizando las posiciones en x y y del tablero y sumándoles los valores necesarios, esto para que si se mueve de lugar el tablero las posiciones se sigan generando en el tablero:

ax: hará que empiece en la posición x más 14 en el mismo eje, esto hará que se centre en la primer posición, se utilizara para llenar las filas de izquierda a derecha.

ay: hará que empiece en la posición de x más 580, esto hará que se centre fila inferior del tablero, se usara para asignar posiciones en y por fila.
axF: hará que empiece en la posición de ax más 567, esto hará que empiece en el lado contrario del tablero en que empieza ax, ya que se utilizara para llenar las filas que van de derecha a izquierda

```
87    var ax: int = tablero_mc.x + 14;
88    var ay: int = tablero_mc.y + 580;
89    var axF: int = ax + 567;
```

Enseguida creamos un for, el cual repetirá 5 veces 2for internos. Cada for interno se repetir 10 veces:

El primero inicia haciendo push de la posición inicial en x, después se incrementa en 63, esto hará que cada vuelta se vaya creando una posición más a la derecha, se incrementa en 63 ya que eso es lo necesario en el tablero para cambiar de una casilla a otra, cuando este for termine, habrá llenado las posiciones en x de una fila impar entera.

El segundo hará exactamente lo mismo que el primero, con la única diferencia de que iniciara en la posición final de x, y se ira decrementando en 63, esto hará que cada vuelta se vaya creando una posición nueva a la izquierda, cando este for termine, habrá llenado las posiciones en x de una fila par entera.

Lo hacemos de esta forma, ya que nuestro tablero en las filas impares las casillas van hacia la derecha y en las filas pares van hacia la izquierda.

El for externo se repite 5 veces, ya que por vuelta llena 2 filas a la vez, si lo hiciéramos 10 veces llenaría 20 filas, ósea, las primeras 10 y 10 inexistentes en el tablero.

```
91    for (i = 0; i < 5; i++) { //rep
92        for (j = 0; j < 10; j++) {
93            aX.push(ax); //Agrego a
94            ax += 63; //Se incremen
95        }
96        for (j = 0; j < 10; j++) {
97            aX.push(axF); //Agrego
98            axF -= 63; //se decreme
99        }
100       ax = tablero_mc.x + 14; //s
101       axF = ax + 567; //se reinic
102   }
```

Ahora lo que sigue es llenar el Array de posiciones en y, para ello realizamos un nuevo for, el cual repite 10 veces y que contiene un solo for interno el cual también se repite 10 veces.

El for interno es el que se encarga de llenar las posiciones en y, empieza ingresando la posición inicial que contiene la variable ay, una vez que se terminó el for interno decrementamos a ay en 63, la incrementamos fuera del for interno ya que llena fila por fila, y todas las casillas de una

misma fila siempre valdrán lo mismo, y lo decrementamos en 63 para que la siguiente vuelta la fila que llene quede 63 más arriba que la anterior, asi crearíamos las filas:

```
103     for (i = 0; i < 10; i++) { //Fo
104         for (j = 0; j < 10; j++) {
105             aY.push(ay); //agrega e
106         }
107         ay -= 63; //se decrementa e
108     }
```

Ahora creamos un nuevo temer, el cual ya se mencionó una vez anteriormente en el timer de tirar el dado, además de variables que usaremos junto a él.

Este nuevo timer lo utilizaremos en una función que se explicara a continuación:

```
110 var timer: Timer = new Timer(500, tmp++);
111 //contadores usados para este timer
112 var tmp: int = 0;
113 var cont: int = 0;
114 var cont2: int = 0;
```

Después de las variables declaramos una de tipo booleano que nos ayudara cuando el jugador haya superado la casilla 100 y aun se siga moviendo.

El timer lo utilizaremos en la función moverEnTablero, que como su nombre lo dice, es la que se encargara de mover las fichas en el tablero.

Lo primero que hacemos en esta función es incrementar las 2 variables contadoras declaradas junto al timer:

```
116 var sePaso:Boolean=false;
117 function moverEnTablero(event: TimerEvent): void {
118     cont++ //incrementamos cont en 1, cont ayudara
119     cont2++ //incrementamos cont2, cont2 ayudara a
```

Después entramos a un if, el cual comparara si cont vale 1, esto para que solo se realicen las siguientes operaciones cada medio segundo, ya que la variable se incrementa cada medio segundo gracias al timer que ya definimos.

Dentro de este if, tenemos otro if en el cual comparamos a la variable sePaso, para saber el jugador se pasó de la casilla 100, si se cumple entonces realizamos 2 tweens a la misma imagen, uno en x y otro en y.

Sabremos a que imagen se le aplicara el tween utilizando aFichasT[contTurno], recordemos que aFichas es el Array que almacena las imágenes y contTurno es el que almacena el turno actual recibiendo valores 0, 1 y 2, de esta manera, utilizaremos la imagen del Array aFichgas en la posición que contenga contTurno.

Ya que sabemos a qué imagen se le aplicara el tween entonces podremos hacerlo.

PRIMERA PARTE DE LA CAPTURA (Líneas 120-135)

```
if (cont == 1) { //Si cont 1 es igual a 1 entonces hacemos tweens
    if(sePaso==true){
        var tweenF2x: Tween = new Tween(aFichasT[contTurno], "x",
        var tweenF2y: Tween = new Tween(aFichasT[contTurno], "y",
        cont = 0; //reseteamos cont, para que al siguiente medio s
        contFichas[contTurno]-- //incrementamos contFl para guarda
    }else{
        var tweenFx: Tween = new Tween(aFichasT[contTurno], "x", S
        var tweenFy: Tween = new Tween(aFichasT[contTurno], "y", S
        cont = 0; //reseteamos cont, para que al siguiente medio s
        contFichas[contTurno]++ //incrementamos contFl para guarda
        if(contFichas[contTurno]>=99){
            sePaso=true;
        }
    }
}
```

SEGUNDA PARTE DE LA CAPTURA (Líneas 120-135)

En esta parte de la captura podemos ver los demás parámetros.

Podemos ver que los tweens inician en las posiciones actuales en x y y de la imagen, y terminan en la posición que se encuentre en los Arrays aX y aY en la posición del Array contFichas en la posición contTurno. Recordemos que aX y aY son los Arrays de las posiciones x y y, contFichas es el que lleva el control de la casilla actual de las fichas, y contTurno es el que sabe que ficha toca en este momento, por lo tanto moveríamos a la imagen a las posiciones en x y y que s encuentren en la casilla anterior a la posición actual de la ficha en turno, se mueve a la anterior por el -1, y esto se hace para que cuando el jugador se pase de la casilla 100 rebote las casillas sobrantes.

```
s a la imagen en los ejes x y y hacia los valores x y y de la siguiente casilla

, Strong.easeOut, aFichasT[contTurno].x, aX[contFichas[contTurno] - 1], 0.1, true);
, Strong.easeOut, aFichasT[contTurno].y, aY[contFichas[contTurno] - 1], 0.1, true);
 segundo su valor vuelva a ser 0 y se vuelva a cumplir la condicion
iar la nueva posicion de la ficha

 Strong.easeOut, aFichasT[contTurno].x, aX[contFichas[contTurno] + 1], 0.1, true);
 Strong.easeOut, aFichasT[contTurno].y, aY[contFichas[contTurno] + 1], 0.1, true);
 segundo su valor vuelva a ser 0 y se vuelva a cumplir la condicion
iar la nueva posicion de la ficha
```

Al terminar esos 2 tweens reseteamos a cont, para que después de medio segundo su valor vuelva a ser uno y se vuelvan a ejecutar estas acciones, esto para crear un efecto de fichas moviéndose

de casilla en casilla. Además decrementamos el valor que lleva el control de la posición de la ficha actual, para actualizar su posición.

Después, si la condición no se cumple, significa que el jugador sigue moviéndose en casillas que no superan la 100, por lo tanto se realizan tweens exactamente de la misma manera, solo que en lugar de -1 ponemos +1, para que en lugar de retroceder una casilla, avance una casilla. Al terminar esos tweens se incrementa el valor que lleva el control de la posición de la ficha actual, para actualizar su posición.

Para terminar con estos if, debemos explicar que inicialmente se entrara a la segunda opción del if, por lo tanto es aquí donde tenemos que comparar si se superó la casilla 100, por lo tanto después de las últimas operaciones hacemos un nuevo if, en el que comparamos a la posición actual de la ficha en turno, si es mayor o igual a 99 significa que llego a la 100, por lo tanto se pasó, asi que a la variable sePaso le damos el valor de true, esto haría que a la siguiente vuelta empiece a decrementarse en lugar de incrementarse, lo que haría el rebote

Al terminar con todo lo anterior, fuera del if de los tweens, realizamos un nuevo if, donde compramos si contador 2 es igual o mayor a tirada, esto nos ayudara a saber si la ficha ya hizo los movimientos que el dado dijo que debía hacer, si se cumple la condición entonces detenemos este mismo timer, reseteamos a cont2 para la siguiente tirada, iniciamos el timer 2, el cual se explicara a continuación, y hacemos falsa a la variable seParo, esto para que en caso de que haya caído exactamente en la casilla 100, no afecte al decir si gano o aun no:

```
136         if (cont2 >= tirada) { //Comparamos si cont2 es igual
137             timer.stop(); //Detenemos el timer
138             cont2 = 0; //reseteamos cont2 para cuando se vuelv
139             timer2.start(); //iniciamos el timer 2
140             sePaso=false;
141         }
142     }
143     timer.addEventListener(TimerEvent.TIMER, moverEnTablero);
```

Ahora explicaremos el timer2, este timer lo utilizaremos con la función timerComparar, la cual lo único que hace es que después de medio segundo le dé un valor de true a la variable listo, la cual hará que una función que se explicara enseguida empiece su proceso, y lo hacemos asi para que la siguiente función se ejecute medio segundo después, y asi se vea un buen efecto visual al llegar a una serpiente o escalera y desplazarse a donde esta lleve:

```
148 var timer2: Timer = new Timer(500, tmp2++);
149 //contadores usados para este timer
150 var tmp2: int = 0;
151 var contF: int = 0;
152 function timerComparar(event: TimerEvent): void { //funcion
153     contF++ //Incrementamos contF en 1, nos ayudara a saber
154     if (contF == 1) { //Si contF es igual a 1 entonces:
155         listo = true; //la varaible booleana listo se hara
156         timer2.stop(); //detenemos el timer2, ya que ya no
157         contF = 0; //reseteamos contF para la siguiente tir
158     }
159 }
160 timer2.addEventListener(TimerEvent.TIMER, timerComparar);
```

Ahora sigue explicar la función más importante del juego, la que ayuda a saber si el jugador cayó en una serpiente o en una escalera.

Antes de explicarla, hay que mencionar que en la línea 162 es donde se declara la variable booleana listo, la cual será la encargada de activar el proceso de esta función.

Al crear la función, lo primero que tenemos es un if, donde comparamos si la variable listo es igual a true, por ello decimos que esta es la encargada de activar la función, si esta se cumple se ejecutaran todos los métodos de la función.

Lo primero que hace al cumplirse es volverla instantáneamente falsa, puesto que estas operaciones solo se deben ejecutar una vez por tirada, además, le volvemos a asignar al dado su evento, para que se pueda volver a pulsar:

```
162  var listo: Boolean = false; //Variable booleana que ayudara a s
163  function serpientesYEscaleras(event: Event): void { //funcion p
164      if (listo == true) { //Si listo es igual a true entonces es
165          listo = false; //a listo le asignamos el valor de false
166          dado_mc.addEventListener(MouseEvent.CLICK, tirarDado);
```

a

Enseguida de lo anterior y aun dentro del if, tenemos un switch, el cual comparara a contFichas en la posición contTurno, para saber en que posición se encuentra la ficha actual después de haberse movido las casillas que dijo el dado.

El switch en cada uno de sus casos realiza 2 tweens a la imagen en turno, uno en x y uno en y, los cuales iniciaran en la posición actual de la imagen y terminaran en la casilla a la que lleva ya sea la serpiente o la escalera.

En total tenemos 9 escaleras y 13 serpientes, por lo que son 22 casos diferentes, en los cuales lo único que cambia es el valor para entrar al caso, y la posición en los Arrays aX y aY a la que se ira.

Por ejemplo, en el caso 7, significa que la ficha se encuentra en la casilla 8, esto ya que nuestras posiciones las tenemos en un Array, y recordemos que los Arrays empiezan desde la posición 0, entonces, si entra a este caso significa que está en la casilla 8, en la cual se encuentra una escalera, escalera la cual lleva a la casilla 26, por lo tanto decimos que termina en aY[25] y aY[25], por la misma razón de que el caso 7 corresponde a la casilla 8:

```
switch (contFichas[contTurno]) { //Switch que ayudara a saber en que posicion se esta
    //============
    //Escaleras.
    //============
    case 7: //caso 1, primera escalera
        //Hacemos tweens que llevaran la ficha a la casilla a la que llega la escalera
        var tweenElx1: Tween = new Tween(aFichasT[contTurno], "x", Strong.easeOut, aFichasT[contTurno].x, aX[25], 0.5, true);
        var tweenEly1: Tween = new Tween(aFichasT[contTurno], "y", Strong.easeOut, aFichasT[contTurno].y, aY[25], 0.5, true);
        contFichas[contTurno] = 25; //a contFi le damos el valor de 25 para actualizar la posicion actual de la ficha
        break;
    case 20:
        var tweenElx2: Tween = new Tween(aFichasT[contTurno], "x", Strong.easeOut, aFichasT[contTurno].x, aX[81], 0.5, true);
        var tweenEly2: Tween = new Tween(aFichasT[contTurno], "y", Strong.easeOut, aFichasT[contTurno].y, aY[81], 0.5, true);
        contFichas[contTurno] = 81;
        break;
```

Tenemos un único caso especial en nuestro switch de serpientes y escaleras, y es el caso 79, ósea, en el que la ficha termino en la casilla 80, este caso es especial ya en la casilla 80 se encuentra una escalera que lleva al jugador a la casilla 100, por lo tanto, además de hacer el tween y actualizar su posición, detenemos todos los timer, removemos el evento de click al dado para que no se pueda seguir tirando, removemos 2 funciones al escenario para evitar errores, y hacemos visible el botón que nos mandara al frame de resultados, esto pasa ya que como llevo al jugador a la casilla 100, este gano:

```
case 79:
    var tweenElx8: Tween = new Tween(aFichasT[contTurno], "x", Strong.easeOut, aFichasT[contTurno].x, aX[99], 0.5, true);
    var tweenEly8: Tween = new Tween(aFichasT[contTurno], "y", Strong.easeOut, aFichasT[contTurno].y, aY[99], 0.5, true);
    contFichas[contTurno] = 99;
    if(sePaso==false){
        timertiempo.stop();
        timerdado.stop();
        timer.stop();
        timer2.stop();
        dado_mc.removeEventListener(MouseEvent.CLICK, tirarDado);
        stage.removeEventListener(Event.ENTER_FRAME, sigJugador);
        stage.removeEventListener(Event.ENTER_FRAME, serpientesYEscaleras);
        resultados_btn.visible = true;
    }
    break;
```

Y ya para terminar con el if que indica que se terminó de mover la ficha, fuera del switch incrementamos a la variable contTurno, para que la siguiente vez que se dé click en el dado, sea la ficha del siguiente jugador la que se mueva, además, tenemos también un if, el cual nos dice si el jugador actual era el último jugador o no, si era el último, entonces el siguiente jugador deberá ser el primero, por lo tanto reseteamos a contTurno:

```
contTurno++;//incrementamos el conta
if (contTurno == contNombres) {//si
    //entonces no le sumaremos 1, lo
    contTurno = 0;
}
```

Después y aun en la función, tenemos un if el cual compara si la cantidad de jugadores es 2, si es asi entonces hacemos invisible la 3er ficha, para que no se vea quieta en el frame:

```
297     if (contNombres == 2) { //Co
298         ficha3t.visible = false;
299     }
```

Nora: el if anterior realmente podría estar fuera de una función.

Ya para finalizar con esta función tenemos un último if, en el cual comparamos si la ficha que acaba de moverse termino en la casilla 100 y si la variable sePaso está en falso. Si esta condición se cumple significa que la ficha llego a la casilla 100 con una tirada exacta, la cual no la hizo rebotar, además de que llego sin la ayuda de la escalera en la casilla 80, por lo tanto hacemos exactamente lo mismo que en ese caso, solo que sin los tweens:

```
301     if (contFichas[contTurno] == 99 && sePaso==false) {
302         //hacemos lo mismo que en el caso en que llegas a la casilla 100 po
303         //solo que esta vez sin el tween
304         timertiempo.stop();
305         timerdado.stop();
306         timer.stop();
307         timer2.stop();
308         dado_mc.removeEventListener(MouseEvent.CLICK, tirarDado);
309         stage.removeEventListener(Event.ENTER_FRAME, sigJugador);
310         stage.removeEventListener(Event.ENTER_FRAME, serpientesYEscaleras);
311         resultados_btn.visible = true;
312     }
```

Ahora tenemos un timer más, el cual se encarga de llevar el tiempo de la partida.

Lo primero que hacemos es declararlo junto a las variables que usaremos con él. Una vez declaradas estas variables iniciamos el timer, ya que queremos que empiece desde que se llega al frame del juego, ósea, en cuanto empieza la partida, ya que será el timer encargado de saber cuánto duro la partida.

Una vez hecho lo anterior, iniciamos con la función, donde lo primero que hacemos es incrementar la variable seg en 1, esto hará que cada segundo se sume 1.

Después tenemos un if, donde compararemos si seg es igual a 60, esto indicara que ya paso un minuto, por lo tanto, si se reseteamos a la variable seg, pero incrementamos a la variable min en 1, esto lo hacemos para que después al reflejarse el nuevo tiempo después del if, se actualice a minutos:

```
319 var timertiempo: Timer = new Timer(1000, tiempo++);
320 var tiempo: int = 0;
321 var seg: Number = 0;
322 var min: Number = 0;
323 timertiempo.start();
324 function Tiempo(event: TimerEvent): void {
325     seg++
326     if (seg == 60) {
327         min++;
328         seg = 0;
329     }
330     tiempo_txt.text = min + ":" + seg + " m.";
331 }
332 timertiempo.addEventListener(TimerEvent.TIMER, Tiempo);
```

Casi para finalizar este frame, tenemos la función sigJugador. Esta función se encarga de mostrar el nombre del jugador en turno, asi como también el nombre del jugador que seguirá después de él.

Para ello, lo primero que hacemos es colocar el nombre del jugador actual, esto lo hacemos colocando en el texto dinámico, el valor del Array aNombres en la posición de contTurno.

Después tenemos un if, el cual nos ayudara a saber cuál es el nombre del siguiente jugador que debemos colocar. Para ello comparamos si contTurno es igual a la cantidad de nombres -1, lo hacemos menos 1 ya que al hacer esta comparación contNombres valdrá 1 más de lo que necesitamos, aunque después se corrija. Si la condición se cumple significa que el siguiente jugador es el jugador 1, por lo tanto colocamos el nombre del jugador utilizando a aNombres en la posición 0, sin embargo, si no se cumple, entonces en el texto dinámico colocamos el valor del Array aNombres en la posición contTurno +1, el +1 es para que se muestre el siguiente:

```
334  function sigJugador(event: Event): void {
335      jactual_txt.text = aNombres[contTurno];//coloco el juga
336      if (contTurno == contNombres - 1) {//si el numero de tu
337          //entonces debemos colocar el nombre del jugador 1
338          jsiguiente_txt.text = aNombres[0];
339      } else {//si no entonces colocamos el siguiente nombre
340          jsiguiente_txt.text = aNombres[contTurno + 1];
341      }
342  }
343  stage.addEventListener(Event.ENTER_FRAME, sigJugador);
```

Por último y para finalizar con el frame 3, tenemos la función resultados, esta función es tan simple como que nos mandara al frame 4 al presionar el botón resultados, el cual se muestra únicamente cuando un jugador llega a la casilla 100:

```
344  //funcion que nos lleva al frame de resultados
345  function Resultados(event: MouseEvent): void {
346      gotoAndStop(4);
347  }
348  resultados_btn.addEventListener(MouseEvent.CLICK, Resultados);
```

FRAME 4

En este frame es donde mostramos los resultados del juego.

Lo primero que hacemos es declarar una variable objeto aux, con sus variantes:

aux.cont: almacena temporalmente el valor de la casilla del jugador
aux.cont: almacena temporalmente el nombre del jugador

```
2  var aux:Object = new Object;
3  aux.cont=0;
4  aux.nombres="";
```

Una vez declaradas las variables anteriores, lo siguiente que hacemos es ordenar a los jugadores de modo que el que obviamente el que llego al final sea el ganador, pero ordenaremos a los demás jugadores de modo que quedara más arriba el que más lejos haya quedado en el tablero al final.

Para ello hacemos un for que se repetirá un número de veces igual al número de jugadores, dentro del cual habrá un for exactamente igual, solo que este se repetirá una vez menos, dentro de este último for tendremos una condición, donde comparamos si el número de casilla del jugador de la posición actual es menor al de la siguiente posición. Si esta se cumple entonces cambiamos de lugar en los Arrays tanto el nombre de los jugadores como su contador de casilla.

Lo hacemos haciendo los siguientes pasos:

1.- a la aux correspondiente le asignamos el valor de la casilla actual.
2.- a la casilla actual le asignamos el valor de la siguiente casilla
3.- a la siguiente casilla le asignamos el valor de la aux correspondiente

```
7   for(i=0;i<aNombres.length;i++){
8       for(j=0;j<aNombres.length-1;j++){
9           if(contFichas[j]<contFichas[j+1]){//
10              //si el numero es menor signific
11              //de esos 2 jugadores,
12              //lo hacemos utilizando las auxi
13              //lo hago con los puntos por que
14              aux.cont=contFichas[j];
15              contFichas[j]=contFichas[j+1];
16              contFichas[j+1]=aux.cont;
17              //lo hago con el nombre para que
18              aux.nombres=aNombres[j];
19              aNombres[j]=aNombres[j+1];
20              aNombres[j+1]=aux.nombres;
21          }
22      }
23  }
```

Nota: Utilizamos las variables auxiliares para no perder datos al momento de cambiar de posición los valores.

Después tenemos una pequeña línea en la cual lo que hacemos es mostrar el tiempo que duro la partida, para que se muestre junto los resultados:

```
24  //muestro el tiempo que duro la partida
25  tiempo_txt.text=min+"m. "+seg+"s.";
```

Enseguida tenemos un switch, el cual se encargara de comparar cuantos jugadores son, y en base a eso mostrara 2 o 3 resultados:

```
26  //switch que imprime los resultados depend
27  switch(aNombres.length){
28      case 2:
29          puesto1.text="1.-  "+aNombres[0];
30          puesto2.text="2.-  "+aNombres[1];
31          break;
32      case 3:
33          puesto1.text="1.-  "+aNombres[0];
34          puesto2.text="2.-  "+aNombres[1];
35          puesto3.text="3.-  "+aNombres[2];
36          break;
37  }
```

Después tenemos la función exportar, esta hace que al presionar el botón exportar se ejecute un switch, el cual dependiendo del número de jugadores exportara 2 o 3 resultados a un documento pdf:

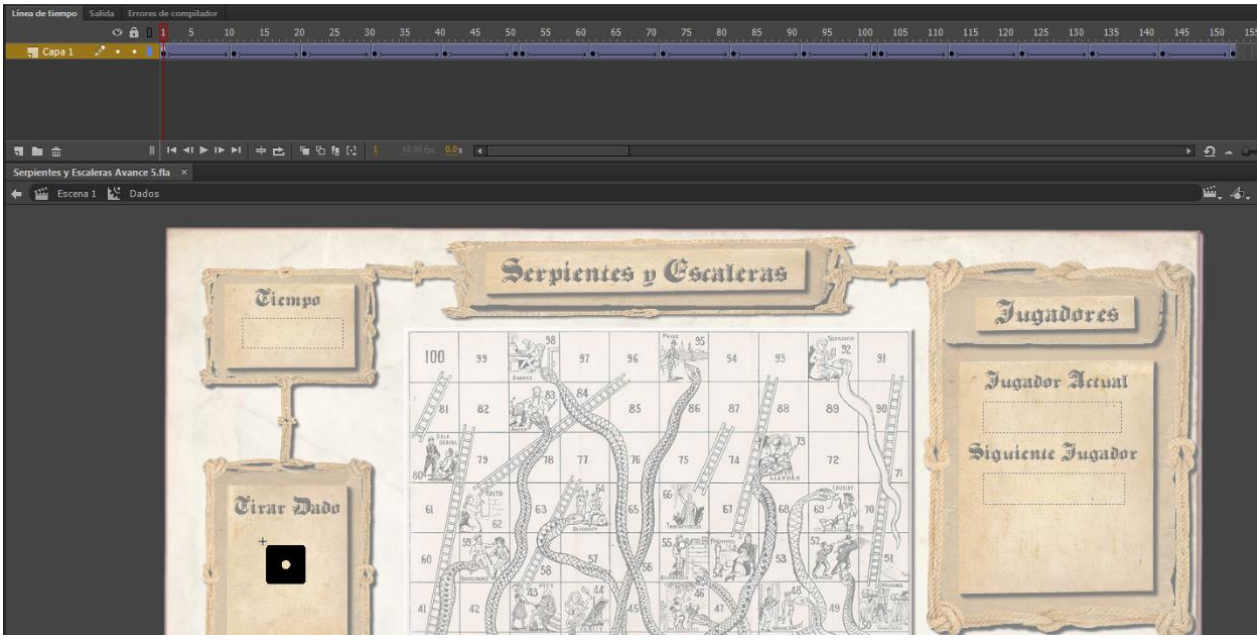
```
38  //Funcion para exportar a pdf
39  var exportPDF: FileReference = new FileReference();
40  function exportar(event:MouseEvent):void{
41      switch (aNombres.length) {//Switch que dependiendo de si so
42          case 2:
43              exportPDF.save("1.- " + aNombres[0] + " \r\n"+
44              "2.- " + aNombres[1] + " \r\n", "Resultados.pdf");
45              break
46          case 3:
47              exportPDF.save("1.- " + aNombres[0] + " \r\n"+
48              "2.- " + aNombres[1] + " \r\n"+
49              "3.- " + aNombres[2] + " \r\n", "Resultados.pdf");
50              break;
51      }
52  }
```

Por ultimo tenemos la función del botón volver, la cual hace que nos vayamos al frame 2 al presionar el botón:

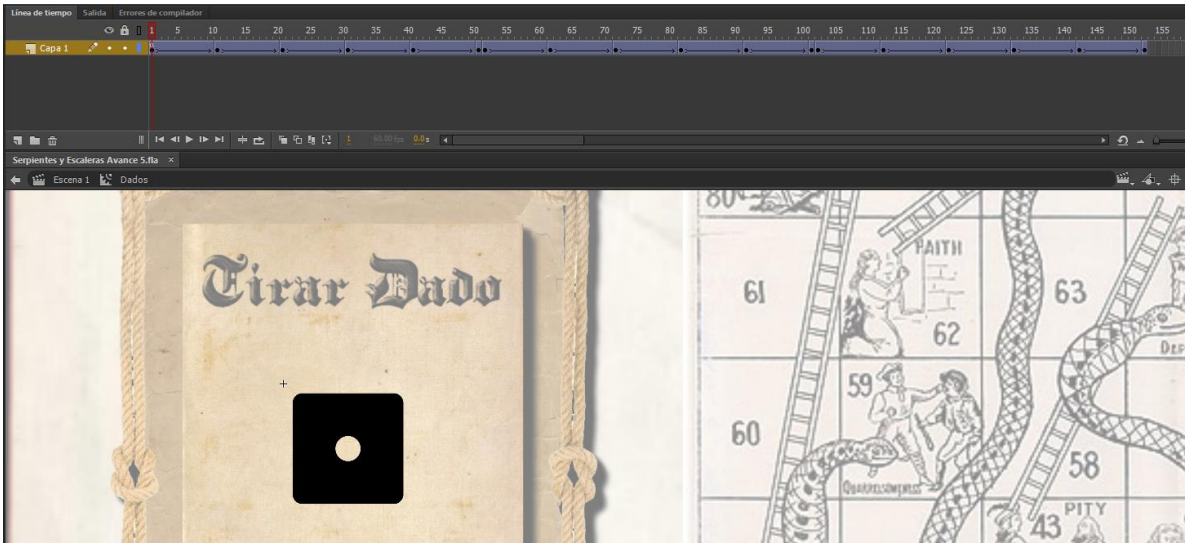
```
43  //Funcion para volver a la pantalla de seleccion de jugadores
44  function volver(Event:MouseEvent):void{
45      gotoAndStop(2);
46  }
47  volver_btn.addEventListener(MouseEvent.CLICK,volver);
```

INTERPOLACIÓN DEL DADO.

Adentro de la imagen del dado podremos notar que existen diferentes fotogramas, estos contienen una interpolación clásica que se encarga que ejecutar todos los fotogramas cuando este sea activado.



Como podemos observar, en el primer fotograma se muestra la primera cara del dado, cada cara del dado tiene 10 fotogramas en una interpolación entre todas, esto es debido a que esos 10 fotogramas toman tiempo en ejecutarse, dando así un tiempo mayor en donde se muestre la animación del cambio de caras.



En esta función del timer que se ejecutara una vez generado un número aleatorio de nuestra variable tirada, existen 6 diferentes casos, ya que son los posibles números que te puede generar dicha variable, cada caso mandara a un fotograma distinto dentro del dado, como podremos observar son de 10 en 10, ya que como mencionamos cada cara del dado abarca 10 fotogramas.

```
49  function TimerDado(event: TimerEvent): void { //funcion para
50      contdado++;
51      if (contdado == 5) {
52          contdado = 0;
53          timer.start();
54          timerdado.stop();
55          switch (tirada) {
56              case 1:
57                  dado_mc.gotoAndStop(1);
58                  break;
59              case 2:
60                  dado_mc.gotoAndStop(11);
61                  break;
62              case 3:
63                  dado_mc.gotoAndStop(21);
64                  break;
65              case 4:
66                  dado_mc.gotoAndStop(31);
67                  break;
68              case 5:
69                  dado_mc.gotoAndStop(41);
70                  break;
71              case 6:
72                  dado_mc.gotoAndStop(51);
73                  break;
74          }
75      }
76  }
77  timerdado.addEventListener(TimerEvent.TIMER, TimerDado);
```

Conclusión.

Podemos concluir que el juego “Serpientes y Escaleras” fue desarrollado de tal manera que fuera un algoritmo automatizado para saber en qué posición del eje de las x y y se debían mover los jugadores.

Implementando la opción de exportar los resultados generales de los jugadores, como el tiempo que duro la partida.