Universidad Politécnica de Victoria

Manual Técnico.

Unidad I.

Memorama.

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

Índice

Introducción.

En este manual tecnico nos encargaremos de explicar detalladamente cada funcion y fragmento de codigo que permite que nuestro memorama funcione, explicaremos por que lo hicimos de tal manera y como funciona. Tambien explicaremos como realizamos algunos efectos visuales en cada uno de los frame y en el mismo memorama.

Empleando diferentes lógicas al momento de desarrollarlo, como también al momento del diseño tanto del Front-end, como tanbien del Back-end.

Este juego cuenta con lo siguiente:

1. 5 Fotogramas diferentes.
2. Cuenta con un multijugador de hasta 4 personas.
3. Cuenta con 24 cartas, 12 pares.
4. Te permite guardar los resultados en un .txt.
5. Cuenta con animaciones variadas, incluyendo interpolaciones de movimiento.

Fotograma 1.

-Se importan las librerias necesarias para eventos del mouse, como tambien sobre los Tweens.

```
4   import flash.events.MouseEvent;
5   import fl.transitions.*;
6   import fl.transitions.easing.*;
7   import fl.transitions.Tween;
8   stop(); //Es necesario hacer un stop para ev
9   //=======================================
```

-Se agregara un “stop();”

-Agregamos Tweens para los cuadros de texto e imágenes en la portada.

```
12  var StrongEaseOut1:Tween= new Tween(estrella1_1_mc,"x",Strong.easeInOut,1511,184.2,2,true);
13  var StrongEaseOut2:Tween= new Tween(estrella2_1_mc,"x",Strong.easeInOut,1511,274,2,true);
14  var StrongEaseOut3:Tween= new Tween(estrella3_1_mc,"x",Strong.easeInOut,1511,314.15,2,true);
15  var StrongEaseOut4:Tween= new Tween(estrella4_1_mc,"x",Strong.easeInOut,1511,402.15,2,true);
16  var StrongEaseOut5:Tween= new Tween(estrella5_1_mc,"x",Strong.easeInOut,1511,512,2,true);
17
18  var StrongEaseOut6:Tween= new Tween(estrella1_2_mc,"x",Strong.easeInOut,-1511,835.3,2,true);
19  var StrongEaseOut7:Tween= new Tween(estrella2_2_mc,"x",Strong.easeInOut,-1511,849.1,2,true);
20  var StrongEaseOut8:Tween= new Tween(estrella3_2_mc,"x",Strong.easeInOut,-1511,809,2,true);
21  var StrongEaseOut9:Tween= new Tween(estrella4_2_mc,"x",Strong.easeInOut,-1511,721,2,true);
22  var StrongEaseOut10:Tween= new Tween(estrella5_2_mc,"x",Strong.easeInOut,-1511,620.7,2,true);
23
24  var StrongEaseOut11:Tween= new Tween(titulo_mc,"y",Strong.easeInOut,1511,64.45,1.5,true);
25  var StrongEaseOut12:Tween= new Tween(entrar_btn,"y",Strong.easeInOut,-1511,653.95,1.5,true);
26
27  var StrongEaseOut13:Tween= new Tween(nombre1_mc,"x",Strong.easeInOut,-1511,697.6,1.5,true);
28  var StrongEaseOut14:Tween= new Tween(nombre2_mc,"x",Strong.easeInOut,1511,94.05,1.5,true);
29
30  var StrongEaseOut15:Tween= new Tween(uni_mc,"x",Strong.easeInOut,-1511,350.9,1.5,true);
31  var StrongEaseOut16:Tween= new Tween(materia_mc,"x",Strong.easeInOut,1511,391.05,1.5,true);
32  var StrongEaseOut17:Tween= new Tween(grupo_mc,"x",Strong.easeInOut,-1511,479.05,1.5,true);
33
34  var StrongEaseOut18:Tween= new Tween(arriba_mc,"y",Strong.easeInOut,-1511,-43.85,3,true);
35  var StrongEaseOut19:Tween= new Tween(abajo_mc,"y",Strong.easeInOut,1511,690.45,3,true);
36
37  var StrongEaseOut20:Tween= new Tween(derecha_mc,"x",Strong.easeInOut,1511,1083.05,3,true);
38  var StrongEaseOut21:Tween= new Tween(izquierda_mc,"x",Strong.easeInOut,-1511,-32.6,3,true);
```

- Función para ir al selector de jugadores.

Por medio de un boton se ejecutara esta funcion, nos mandara al fotograma 3.

```
42  function Entrar(event:MouseEvent):void{
43  gotoAndStop(3);
44  }
45  //Asignamos la funcion "Entrar" al boton "entrar_btn".
46  entrar_btn.addEventListener(MouseEvent.CLICK,Entrar);
```

Fotograma 2.

COLOCACION ALEATORIA DE LAS CARTAS.

Para colocar las cartas de manera aleatoria lo primero que hicimos fue declarar 3 Arrays:

```
19    var aX: Array = new Array; //array de posiciones en x
20    var aY: Array = new Array; //array de posiciones en y
21    var aR: Array = new Array; //array de randoms
```

aX: Array que almacenara posiciones para el eje x
aY: Array que almacenara posiciones para el eje y
aR: Array que almacenara diferentes números Random

Con el Array aR lo primero que hacemos es darle 24 veces el valor 25. Hacemos esto 24 veces ya que al final almacenara 24 valores:

```
43    for (i = 0; i < 24; i++) {
44        aR.push(25);
45    }
```

Con los Arrays aX y aY lo que hacemos es asignarles los valores de las diferentes posiciones en x e y respectivamente que queremos, estas posiciones serán en las que se colocaran las 24 cartas del memorama:

```
aX = [18, 18, 152, 152, 286, 286, 420, 420, 554, 554, 688, 688, 18, 18, 152, 152, 286, 286, 420, 420, 554, 554, 688, 688];
aY = [31, 213, 31, 213, 31, 213, 31, 213, 31, 213, 31, 213, 395, 582, 395, 582, 395, 582, 395, 582, 395, 582, 395, 582];
```

Una vez que los 3 Arrays tienen valores predeterminados entramos a la función “Aleatorio”, en la cual tenemos 2 procesos muy importantes, el primero de ellos es el llenado del Array aR con números del 1-24 en un orden aleatorio sin repetirse:

```
for (i = 0; i < 24; i++) {
    //Le doy un valor random a la variable
    rdm = Math.random() * 24 + 1;
    //=============================================================================================================
    //For para comparar el nuevo random con los que ya hay.
    //=============================================================================================================
    for (j = 0; j < 24; j++) {
        //Si el nuevo random es igual al de la casilla actual la variable existe se hara true.
        if (rdm == aR[j]) {
            existe = true;
        }
        //Al final se habria comparado con todo el array, y con que una vez haya dado true es suficiente.
    }
    //Si "existe" es igual a true entonces decremento el contador del for externo para que no cuente esa vuelta
    if (existe == true) {
        i--;
        //Si no, entonces le asigno el valor del random al array en la posicion actual.
    } else {
        aR[i] = rdm;
    }
    //Reinciamos la variable existe para que no se confunda en la siguiente vuelta.
    existe = false;
}
```

Para que esto funcionara primero que nada creamos un For, el cual se repetirá 24 veces, desde 0 a 23. Este For siempre realizara un random entre 1 y 24 y el resultado lo almacenara en la variable rdm.

```
for (i = 0; i < 24; i++) {
    //Le doy un valor random a la variable
    rdm = Math.random() * 24 + 1;
    //====================================
```

Después lo que debemos hacer es evaluar ese valor random para saber si no lo ha generado antes, puesto que deseamos que ningún valor se repita. Para ello lo que hacemos es un nuevo for dentro del for que ya tenemos, el cual compara el nuevo número random con todas las posiciones del Array aR, utilizando la condición de que si el número random es igual a la posición actual gracias al for, la variable Booleana "existe" que fue declarada con un valor false, se vuelve true :

```
for (j = 0; j < 24; j++) {
    //Si el nuevo random es igual
    if (rdm == aR[j]) {
        existe = true;
    }
    //Al final se habria comparado
}
```

Una vez que termine el for anterior se hará una condición, en la cual evaluaremos a la variable "existe" anterior mente usada, de tal forma que si es igual a true entonces decrementamos el contador del for externo, esto lo hacemos para evitar que el for de una vuelta sin haber ingresado un dato random nuevo, puesto que si es igual a true significa que el valor que se dio se repitió, y no queremos ese si no uno nuevo. Si la condición no se cumple, ósea que la variable "existe" es igual a false, significa que el random generado no se repite, entonces lo que hacemos es colocar el número random en el Array aR en la posición actual del for exterior.

Al final reseteamos la variable "existe" para la siguiente vuelta:

```
    if (existe == true) {
        i--;
        //Si no, entonces l
    } else {
        aR[i] = rdm;
    }
    //Reinciamos la variabl
    existe = false;
}
```

En pocas palabras, si el numero generado ya se había ingresado antes, el for se repetirá con el mismo número en su contador hasta que el random genere un nuevo número, para que asi no se termine el for hasta llenar el Array de números random aR con números diferentes ordenados aleatoriamente.

Una vez que se terminó de llenar el Array aR, lo que sigue es colocar las 24 imágenes en las 24 posibles posiciones que definimos en los Arrays aX y aY.

Para ello realizamos un nuevo for que también se repite 24 veces del 0 al 23, el cual repetirá un switch con 24 casos (La imagen solo muestra 4 casos pero son 24):

```
95      for (i = 0; i < 24; i++) {
96          switch (aR[i]) {
97              case 1:
98                  img1.x = aX[i];
99                  img1.y = aY[i];
100                 break;
101             case 2:
102                 img2.x = aX[i];
103                 img2.y = aY[i];
104                 break;
105             case 3:
106                 img3.x = aX[i];
107                 img3.y = aY[i];
108                 break;
109             case 4:
110                 img4.x = aX[i];
111                 img4.y = aY[i];
112                 break;
```

Lo que hace el for anterior es con la ayuda del switch ir evaluando el valor de la variable aR en la posición actual del for, y dependiendo de que numero se encuentre en dicha posición entrara a un caso diferente, donde se asignara a una imagen la posición en los ejes x y y que se encuentra en la posición actual de los Arrays aX y aY.

Ejemplo, si en la posición 0 del Array de números random aR se encuentra el número 4, entonces se entrara al caso 4, en el cual a la imagen 4 se le agregarían las posiciones en x y y que almacenan los Arrays aX y aY en las posiciones 0, de esta manera la imagen 4 se colocaría en la primer posición posible para las 24 cartas, en lugar de la 4ta posición que es la que le corresponde originalmente. Al final del for se abrían colocado todas las imágenes en posiciones aleatorias gracias a los números random.

Al final de la función Aleatorio, después de todo lo anterior, se inicia un timer que servirá para el tiempo en el juego, y también se coloca el nombre del primer jugador, para que se sepa que es su turno (También asignamos la función Aleatorio al botón Siguiente en la pantalla de reglas).

```
    timer3.start(); //iniciamos el contador de tiempo para el limite de un minuto
    jugadorJugando.text = aNombres[0]; //colocamos el nombre del jugador que esta jugando
}
//Asignamos la funcion "Aleatorio" al boton "reglas_btn".
reglas_btn.addEventListener(MouseEvent.CLICK, Aleatorio);
```

-Variables para los contadores de los 12 pares de cartas.

```
213    var par1: int = 0;
214    var par2: int = 0;
215    var par3: int = 0;
216    var par4: int = 0;
217    var par5: int = 0;
218    var par6: int = 0;
219    var par7: int = 0;
220    var par8: int = 0;
221    var par9: int = 0;
222    var par10: int = 0;
223    var par11: int = 0;
224    var par12: int = 0;
```

-Variable para el puntaje general de cada jugador.

```
228    var puntaje: Number = 0;
```

-Variables de los contadores “Timers”.

```
232    //"Timer" para la funcion de pares iguales, con su contador de tiempo ("cont1").
233    var timer1: Timer = new Timer(1000, cont++);
234    var cont: int = 0;
235    //"Timer" para la funcion de pares diferentes, con su contador de tiempo ("dif").
236    var timer2: Timer = new Timer(1000, dif++);
237    var dif: int = 0;
238    //"Timer" para la funcion de tiempo, con su contador de tiempo ("seg").
239    var timer3: Timer = new Timer(1000, tmp++);
240    var tmp: int = 0;
```

El “timer1” corresponde a la función para comparar si los pares son iguales, para que exista un tiempo de 1 segundo para que se puedan mostrar las cartas.

El “timer2” corresponde a la función para comparar si los pares seleccionados son diferentes, para dar un tiempo de 1 segundo, para que el usuario se dé cuenta de dicha comparación.

El “timer3” corresponde a la función para el contador de tiempo por jugador.

Para el “timer3”:

Lo primero que se hizo obviamente fue declarar el timer y a la variable seg:

seg: Incrementara cada que en el timer pase 1 segundo y es la que utilizaremos para comparar.

```
var timer3: Timer = new Timer(1000, tmp++);
var tmp: int = 0;

var seg: int = 0;
```

Una vez declarado creamos la función Tiempo en la cual lo único que hacemos es incrementar la variable seg en 1 cuando se ejecuta la función, ósea cada segundo, y también vamos mostrando el tiempo en la pantalla de juego.

Lo más importante de esta función es la condición, donde comparamos si segundos es igual a 60, si esto se cumple significa que ya paso un minuto, ósea que se acabó el tiempo del jugador, por lo que hacemos que la variable fin sea igual a true y a la razonDeFin le decimos que “Se terminó el tiempo!” esto lo hacemos para que la función terminar sepa que se terminó el juego por esta razón:

```
function Tiempo(event: TimerEvent): void {
    seg++
    tiempo_txt.text = seg + " s.";
    if (seg == 60) {
        fin = true;
        razonDeFin.text = "Se termino el tiempo!";
    }
}
//Arrancar el tiempo.
timer3.addEventListener(TimerEvent.TIMER, Tiempo);
```

-Variables que funcionan como contadores, que se asignaran a las funciones de los “Timers” a los “Timers”.

```
244  var cont1: int = 0;
245  var contdiferentes: int = 0;
246  var seg: int = 0;
```

-Variable que se usara para darle un valor numérico si es que se cumple una condición.

```
250  var diferentes: int = 0;
```

- Las Funciones de "**Imagen1**" hasta "**Imagen24**" son iguales, cambiando sus variables de los contadores de "**par1**" hasta "**par12**", con sus respectivas pares de imágenes, ("**img1**" e "**img2**" pertenecen al "**par1**"), ("**img3**" e "**img4**" pertenecen al "**par2**") y así sucesivamente.

Al presionar el botón "**img1**" el contador "**par1**" se le sumara 1, esto con la finalidad de comprar en otra función.

```
270  function Imagen1(event: MouseEvent): void {
271      //Al presionar el boton "img1" el contador "par1" se le sumara 1.
272      par1++;
273      //Adentro de la "img1" estaran 2 fotogramas con una imagen en cada un
274      img1.gotoAndPlay(2);
275      //Removeremos el "EventListener", para que no se pueda dar otro click
276      img1.removeEventListener(MouseEvent.CLICK, Imagen1);
277      //Si el contador "par1" es igual a 2, esto pasara al dar click a las
278      if (par1 == 2) {
279          //Empezara un contador de tiempo "timer1".
280          timer1.start();
281      }
282  }
283  img1.addEventListener(MouseEvent.CLICK, Imagen1);
284  //========================================================================
285  //Funcion para la imagen 2 ("img2"), esta carta es un boton.
286  //========================================================================
287  function Imagen2(event: MouseEvent): void {
288      par1++;
289      img2.gotoAndPlay(2);
290      img2.removeEventListener(MouseEvent.CLICK, Imagen2);
291      if (par1 == 2) {
292          timer1.start();
293      }
294  }
295  img2.addEventListener(MouseEvent.CLICK, Imagen2);
```

Línea 273: Adentro de la "img1" estarán 2 fotogramas con una imagen en cada uno, se ira al fotograma 2 en donde está la carta con la imagen destapada.

Línea 275: Removeremos el "EventListener", para que **no** se pueda dar otro clic a la misma carta en ese mismo momento.

Línea 277: Si el contador "par1" es igual a 2, esto pasara al dar clic a las 2 imágenes pares ("img1" y "img2").

Línea 279: Empezara un contador de tiempo "timer1

- Función para la imagen 24 ("**img24**"), esta carta es un botón.

-Cómo podremos observar esta es la última imagen ("**img24**"), en donde cambia es en el número de "**par**", siguiendo los mismos pasos que con la imagen 1.

```
561  function Imagen24(event: MouseEvent): void {
562      par12++;
563      img24.gotoAndPlay(2);
564      img24.removeEventListener(MouseEvent.CLICK, Imagen24);
565      if (par12 == 2) {
566          timer1.start();
567      }
568  }
569  img24.addEventListener(MouseEvent.CLICK, Imagen24);
```

- Función para comparar si los pares son iguales.

En esta función incluiremos un contador de tiempo, para hacer que se muestren las 2 cartas y te de un segundo para mirarlas, al hacer esto, se detendrá el contador de tiempo, y se reiniciaran las variables de ese Timer.

Línea 578: Se detendrá el "timer1", para evitar que se siga contando aun cuando ya no lo requiramos.

```
573  function ParesIguales(event: TimerEvent): void {
574      //"cont1" se incrementara en 1 cada vez que se incremente el contador del "timer1" ("cont")
575      cont1++;
576      //Si "cont1" es igual a 1.
577      if (cont1 == 1) {
578          //Se detendra el "timer1", para evitar que se siga contando aun cuando ya no lo requirar
579          timer1.stop();
580          //Se incrementara en 1 a la variable "puntaje", para aumentar en 1 cuando tengas un par
581          puntaje++;
582          puntaje_txt.text = String(puntaje);
583          trace("Si jalo");
584          trace(puntaje);
585          //Se borraran los valores del contador para el "timer1" ("cont"), y para el contador "c
586          //Esto evitara que se ciclen los contadores cada vez que selecciones una carta.
587          cont1 = 0;
588          cont = 0;
589      }
590      /*En esta misma funcion, si las variables de contadores "par1,par2,par3,etc." son iguales a
591      (dando a entender que seleccionaste las dos imagenes iguales, haran invisibles sus respecti
592      if (par1 == 2) {
593          var STween1: Tween = new Tween(img1, "y", Strong.easeInOut, img1.y, -1511, 3, true);
594          var STween2: Tween = new Tween(img2, "y", Strong.easeInOut, img2.y, -1511, 3, true);
595          //img1.visible=false;
596          //img2.visible=false;
597      }
```

Línea 590: En esta misma función, si las variables de contadores "par1, par2, par3, etc." son iguales a 2 (dando a entender que seleccionaste las dos imágenes iguales, harán invisibles sus respectivas imágenes.

Aquí termina la función de comparación de pares iguales, podremos notar que llega hasta el “par12”, entre el “par1” y el “par12” se cumplirán las mismas condiciones (si todos los pares son igual a 2, si se cumple cualquiera de estas condiciones de ejecutaran 2 Tweens, estos servirán para mover las cartas fuera del fotograma.

```
657         }
658         if (par12 == 2) {
659             var STween23: Tween = new Tween(img23, "y", Strong.easeInOut, img23.y, -1511, 3, true);
660             var STween24: Tween = new Tween(img24, "y", Strong.easeInOut, img24.y, -1511, 3, true);
661             //img23.visible=false;
662             //img24.visible=false;
663         }
664     }
665     //Arrancar el "timer1".
666     timer1.addEventListener(TimerEvent.TIMER, ParesIguales);
```

- Función para la validación de pares diferentes.

Valida si son diferentes las imágenes que seleccionaste, dando a entender que son impares las imágenes seleccionadas.

En esta función se repetirán los "if", dependiendo de las variables "par".

```
692 function Validar(event: Event): void {
693     //Si "par1" es igual a 1 y "otro par diferente" es igual a 1, dando a entender que se
694     //una carta se le sumara en 1 al contador de su respectivo par.
695     if ((par1 == 1 && par2 == 1) || (par1 == 1 && par3 == 1) || (par1 == 1 && par4 == 1)
696         //Se iniciara el "timer2".
697         timer2.start();
698     }
699     if ((par2 == 1 && par1 == 1) || (par2 == 1 && par3 == 1) || (par2 == 1 && par4 == 1)
700         timer2.start();
701     }
702     if ((par3 == 1 && par1 == 1) || (par3 == 1 && par2 == 1) || (par3 == 1 && par4 == 1)
703         timer2.start();
```

Si "par1" es igual a 1 y "otro par diferente" es igual a 1, dando a entender que seleccionaste pares diferentes, recordando que al seleccionar una carta se le sumara en 1 al contador de su respectivo par. Esta comparación se hará con todos los 12 pares, si se cumple cualquiera de esa condición se iniciara el “timer2”.

- Función para saber si son diferentes pares.

```
670 function DiferentesTimer(event: TimerEvent): void {
671     //"contdiferentes" se incrementara en 1 cada vez que se in
672     contdiferentes++;
673     //Si "contdiferentes" es igual a 1.
674     if (contdiferentes == 1) {
675         //Se detendra el "timer2", para evitar que se siga con
676         timer2.stop();
677         trace("Si jalo");
678         //Se borraran los valores del contador para el "timer2
679         //Esto evitara que se ciclen los contadores cada vez q
680         dif = 0;
681         contdiferentes = 0;
682         //Se le asignara el valor de 1 a la variable "diferent
683         diferentes = 1;
684     }
685 }
686 //Arrancar el "timer2".
687 timer2.addEventListener(TimerEvent.TIMER, DiferentesTimer);
```

Cuando se ejecute el “timer2”, "contdiferentes" se incrementara en 1 cada vez que se incremente el contador del "timer2" ("dif").

Si "contdiferentes" es igual a 1, se detendrá el "timer2", para evitar que se siga contando aun cuando ya no lo requiramos, también se reiniciaran en valor a “0” los contadores del timer y de “contdiferentes”, como también se le asignara el valor de “1” a la variable “diferentes”.

Al iniciar el "timer2", se ejecutaran las instrucciones de este, y ahí hay una variable "diferentes" se le asignara el valor de 1, al ser así, se ejecutaran las siguientes instrucciones.

```
734     if (diferentes == 1) {
735         trace("Si jalo2");
736         //A todas las 24 imagenes
737         img1.gotoAndStop(1);
738         img2.gotoAndStop(1);
739         img3.gotoAndStop(1);
740         img4.gotoAndStop(1);
741         img5.gotoAndStop(1);
742         img6.gotoAndStop(1);
743         img7.gotoAndStop(1);
744         img8.gotoAndStop(1);
745         img9.gotoAndStop(1);
746         img10.gotoAndStop(1);
747         img11.gotoAndStop(1);
```

A todas las 24 imágenes se irán a su fotograma 1, que es donde tienen su imagen de su carta volteada.

```
758         img22.gotoAndStop(1);
759         img23.gotoAndStop(1);
760         img24.gotoAndStop(1);
```

A todas las 24 imágenes se les asignara sus "EventListener" para que puedan volver a poder seleccionar dichas cartas.

```
762         img1.addEventListener(MouseEvent.CLICK, Imagen1);
763         img2.addEventListener(MouseEvent.CLICK, Imagen2);
```

Hasta las 24 imágenes.

```
772         img11.addEventListener(MouseEvent.CLICK, Imagen11);
773         img12.addEventListener(MouseEvent.CLICK, Imagen12);
774         img13.addEventListener(MouseEvent.CLICK, Imagen13);
775         img14.addEventListener(MouseEvent.CLICK, Imagen14);
776         img15.addEventListener(MouseEvent.CLICK, Imagen15);
777         img16.addEventListener(MouseEvent.CLICK, Imagen16);
778         img17.addEventListener(MouseEvent.CLICK, Imagen17);
779         img18.addEventListener(MouseEvent.CLICK, Imagen18);
780         img19.addEventListener(MouseEvent.CLICK, Imagen19);
781         img20.addEventListener(MouseEvent.CLICK, Imagen20);
782         img21.addEventListener(MouseEvent.CLICK, Imagen21);
783         img22.addEventListener(MouseEvent.CLICK, Imagen22);
784         img23.addEventListener(MouseEvent.CLICK, Imagen23);
785         img24.addEventListener(MouseEvent.CLICK, Imagen24);
```

A las variables de los 12 pares ("par1" hasta el "par12"), se reiniciara su valor inicial ("0").

```
787             par1 = 0;
788             par2 = 0;
789             par3 = 0;
790             par4 = 0;
791             par5 = 0;
792             par6 = 0;
793             par7 = 0;
794             par8 = 0;
795             par9 = 0;
796             par10 = 0;
797             par11 = 0;
798             par12 = 0;
```

A la variable "diferente" se reiniciara su valor inicial ("0").

```
800             diferentes = 0;
801             //Por ultimo se detendra su "timer2".
802             timer2.stop();
803         }
804     }
805     //Este es el "ENTER_FRAME" de la funcion "Validar".
806     stage.addEventListener(Event.ENTER_FRAME, Validar);
```

Y por último se detendrá el “timer2”, acabando la función “validar”.

FUNCIÓN PARA CUANDO UN JUGADOR TERMINA SU TURNO.

La función terminar lo que hace es detectar cuando un jugador termino su turno y almacenar sus puntos y tiempos conseguidos, además de verificar de qué forma termino e iniciar el juego otra vez si aún quedan jugadores por jugar.

Antes que nada necesitamos declarar dos Arrays:

aPuntos: Almacena los puntos que consigan los jugadores.
aTiempo: Almacena los diferentes tiempos que hayan hecho los jugadores.

```
var aPuntos: Array = new Array;
var aTiempo: Array = new Array;
```

También necesitamos crear un nuevo timer para un pequeño proceso de tiempo entre jugador y jugador. Declaramos el timer 2 que cada 1000 milisegundos se incrementara con la variable tmp2. Declaramos también la variable contadora contRegresiva que cada que pase un segundo en el timer se incrementara en 1 dentro de la función TiempoRegresivo, será la que usemos para el proceso.

Por ultimo asignamos la función TiempoRegresivo al timer4:

```
var timer4: Timer = new Timer(1000, tmp2++);
var tmp2: int = 0;
//devlaracion de variable cuenta regresiva
var contRegresiva: int = 0;

function TiempoRegresivo(event: TimerEvent): void {
    contRegresiva++
}
//Arrancar el tiempo.
timer4.addEventListener(TimerEvent.TIMER, TiempoRegresivo);
```

Ahora entramos ya a lo que es la función Terminar. Lo primero que se hace es comparar si la variable puntaje que se incrementa cada que un jugador encuentra un par es igual a 12, entonces a la variable booleana fin que empieza en false, le damos el valor de true, y a un texto dinámico le asignamos el mensaje “Terminaste!”:

```
808  function terminar(event: Event): void {
809      if (puntaje == 12) {//verificamos si
810          fin = true;
811          razonDeFin.text = "Terminaste!";/
812      }
```

La razón por la que le damos a fin el valor true es para saber cuándo el jugador ya termino y así poder ejecutar el método que se explicara a continuación. Y la razón por la que a razonDeFin le damos el mensaje “Terminaste!” es para que el jugador sepa de qué forma termino su turno.

Nota: Se programaron 2 formas de terminar los turnos, la primera es que el jugador encuentre los 12 pares, por eso lo anterior. La segunda es que se acabe el tiempo, esto ocurre cuando el timer3 llega a 60.

A continuación empezamos con el if más importante de la función, en el cual comparamos si la variable fin es igual a true, si es así significa que el jugador termino su turno, así que empezaremos todos los procesos necesarios para empezar el turno del siguiente jugador, o enviar a la pantalla de resultados.

Para ello, si se cumple la condición lo primero que hacemos es detener el timer3, para que ya no se siga aumentando el tiempo del jugador que acaba de terminar, e iniciamos el timer4, para empezar con la espera de 5 segundos entre jugador y jugador.

Después entramos a otra condición, en la que comparamos si aún quedan jugadores o no (Recordar que la variable objeto datos.numJ almacena el número de jugadores que se seleccionó desde el frame 3). Si se cumple entonces mostramos los mensajes de siguiente jugador y el mensaje “en”, así como también le asignamos a un texto dinámico el nombre del jugador al que le tocara jugar a continuación.

En cambio, si la condición no se cumple, ósea, que ya no quedan más jugadores, entonces solo en lugar de colocar el nombre del siguiente jugador colocamos el mensaje “Calculando Resultados”:

```
if (fin == true) {//comparamos si fin es igual a true
    //si es asi entonces detenemos el tiempo del juego
    timer3.stop();
    timer4.start();
    if (contJugadores < datos.numJ) {//comparamos si a
        //si es asi entonces hacemos visibles mensajes
        msjSiguienteJ.visible = true;
        en.visible = true;
        nombreSig_txt.text = aNombres[contJugadores];/
    } else {
        //si no entonces mostramos un mensaje "Calcula
        nombreSig_txt.text = "Calculando Resultados";
    }
```

Después del if anterior hacemos visibles diferentes elementos que conforman la pantalla de espera entre jugador y jugador. Lo último que mostramos es el resultado de la resta de 5 menos lo que valga contRegresiva, esto dará un efecto de cuenta regresiva de 5 a 1, y cuando termine iniciara el siguiente jugador:

```
trans_mc.visible = true;
razonDeFin.visible = true;
endScreen_mc.visible = true;
endScreen.visible = true;
nombreSig_txt.visible = true;
cuentaRegresiva_txt.visible = true;
cuentaRegresiva_txt.text = "" + (5 - contRegresiva);
```

Una vez hecho lo anterior lo que sigue es saber si ya pasaron los 5 segundos, para que solo después de esto hacer todas las operaciones necesarias.

Cuando se cumpla la condición de los 5 segundos haremos los procesos, los cuales van dentro de otro if, en el cual volvemos a comparar si aún quedan jugadores pendientes o si ya acabaron todos.

Si aún quedan jugadores entonces entramos a un gran proceso, en el cual lo primero que se hace es hacer visibles las imágenes como se muestra en la imagen (Se hacen visibles las 24 imágenes, no solo las 10 que se ven):

```
if (contRegresiva == 5) {//comparamos s
    if (contJugadores < datos.numJ) {//
        //si es asi entonces mostramos
        img1.visible = true;
        img2.visible = true;
        img3.visible = true;
        img4.visible = true;
        img5.visible = true;
        img6.visible = true;
        img7.visible = true;
        img8.visible = true;
        img9.visible = true;
        img10.visible = true;
```

Una vez hecho esto lo que sigue es volver a randomizar y colocar las imágenes.

Para poder hacer esto primero volvemos a inicializar todas las posiciones del Array aR en 25:

```
//================================================
//volvemos inicializar el array de randoms en 25
//================================================
for (i = 0; i < 24; i++) {
    aR[i] = 25;
}
```

Después debemos repetir todo el proceso que se realizó en la función Aleatorio anteriormente explicada.

Después del switch que copiamos de la función Aleatorio a la función terminar, lo que hacemos es que nuestras imágenes vuelvan todas a su fotograma 1, en el cual contienen toda la parte de atrás de la carta (Esto se hace para las 24 imágenes, no solo para las 10 que se muestran):

```
timer3.start();
//hacemos que todas las imagenes se muestren volteadas
img1.gotoAndStop(1);
img2.gotoAndStop(1);
img3.gotoAndStop(1);
img4.gotoAndStop(1);
img5.gotoAndStop(1);
img6.gotoAndStop(1);
img7.gotoAndStop(1);
img8.gotoAndStop(1);
img9.gotoAndStop(1);
img10.gotoAndStop(1);
```

Enseguida reseteamos todas nuestras variables de pares, para eliminar cualquier cambio realizado por el jugador anterior:

```
par1 = 0;
par2 = 0;
par3 = 0;
par4 = 0;
par5 = 0;
par6 = 0;
par7 = 0;
par8 = 0;
par9 = 0;
par10 = 0;
par11 = 0;
par12 = 0;
```

Después hacemos invisibles los elementos que mostramos anteriormente, para que 5 segundos después de que se mostraron se quiten. También colocamos el nombre jugador para el nuevo turno e incrementamos a la variable contJugadores en 1, para que en la siguiente vuelta algunas operaciones como mostrar el nombre del jugador o la comparación de si aún hay más jugadores la usen con el nuevo valor:

Así es como terminamos el proceso en casi de que aun haya jugadores por jugar.

```
endScreen_mc.visible = false;
endScreen.visible = false;
razonDeFin.visible = false;
msjSiguienteJ.visible = false;
nombreSig_txt.visible = false;
en.visible = false;
cuentaRegresiva_txt.visible = false;
jugadorJugando.text = aNombres[contJugadores];
contJugadores++//incrementamos el contador de
```

Ahora, cuando la condición no se cumple, ósea que ya no hay más jugadores, lo que hacemos es tan simple como agregar a los Arrays aPuntos y a tiempo los puntos y el tiempo respectivamente que hizo el jugador que acaba de terminar. Además también detenemos el timer4 para detener la cuenta regresiva.

Se puede ver que después al escenario le removemos la función terminar, esto lo hacemos ya que al cambiar al frame 5, que es lo último que hacemos y lo más importante de esta parte después de guardar los datos, nos marcaba un error que no dejaba de mostrarse, entonces al remover el evento evitamos esto:

```
} else {
    //si no entonces significa que no hay mas jugadores, por
    aPuntos.push(puntaje);
    aTiempo.push(seg);
    timer4.stop();
    stage.removeEventListener(Event.ENTER_FRAME, terminar);
    gotoAndStop(5);
}
```

Por último, fuera ya del if de si quedan jugadores o no, pero dentro del if de los 5 segundos, lo que hacemos es limpiar el puntaje que se muestra, para que el siguiente jugador no vea desde el inicio los puntos que se hicieron anteriormente.

Detenemos el timer4 para detener la cuenta regresiva aun que se entre a la condición de que quedan más jugadores, y por la misma razón agregamos a los Arrays aPuntos y aTiempo el tiempo y puntos del jugador que termino.

Para finalizar reseteamos las variables de la cuenta regresiva, los segundos y el puntaje, esto para que el siguiente jugador empieza desde 0 y no desde donde se quedó el jugador anterior, y ya fuera de la función asignamos la misma al escenario:

```
1057                puntaje_txt.text = "";//limpiamos el mar
1058                timer4.stop();//detenemos el temporizado
1059                fin = false;//hacemos que fin sea igual
1060                aPuntos.push(puntaje);//almacenamos los
1061                aTiempo.push(seg);//almacenamos el tiemp
1062                //reseteamos puntos, tiempo y cuenta reg
1063                contRegresiva = 0;
1064                seg = 0;
1065                puntaje = 0;
1066            }
1067        }
1068    }
1069    //asignamos la funcion terminar al escenario
1070    stage.addEventListener(Event.ENTER_FRAME, terminar);
```

Fotograma 3.

Lo primero que hacemos es importar las librerías necesarias para eventos del mouse y tweens.

Después creamos 5 tweens, 1 para cada carta de selección de jugadores y uno para el mensaje:

```
import flash.events.MouseEvent;
import fl.transitions.*;
import fl.transitions.easing.*;
import fl.transitions.Tween;
//=============================================================================================
//Tweens para las cartas de opcion de numero de jugadores y el mensaje
//=============================================================================================
var F3EaseOut1: Tween = new Tween(jugadores_mc, "y", Strong.easeInOut, -1511, 77.05, 2, true);
var F3EaseOut2: Tween = new Tween(carta1_btn, "x", Strong.easeInOut, -1511, 98.45, 2, true);
var F3EaseOut3: Tween = new Tween(carta2_btn, "x", Strong.easeInOut, -1511, 363.95, 2, true);
var F3EaseOut4: Tween = new Tween(carta3_btn, "x", Strong.easeInOut, 1511, 615.05, 2, true);
var F3EaseOut5: Tween = new Tween(carta4_btn, "x", Strong.easeInOut, 1511, 875.45, 2, true);
```

A continuación creamos una variable objeto, a la cual le creamos la variante numJ la cual almacenara el número de jugadores que seleccione el usuario:

```
var datos: Object = new Object;
datos.numJ = 0;
```

FUNCIONES PARA SELECCIONAR NÚMERO DE JUGADORES.

Enseguida de crear la variable objeto, creamos una función por cada carta de selección de jugadores, cada función manda al frame donde se ingresaran los nombres, con la única diferencia de que cada uno le asigna a la variable objeto un valor diferente (en total son 4 funciones, donde solo cambia la el nombre de la carta, la carta a la que se le asigna, y el valor que le da a la variable, los valores que se dan son 1, 2, 3 y 4):

```
//=============================================================================================
//Funcion para el selector de jugadores de la carta 1 ("carta1_btn"), que es un boton.
//=============================================================================================
function Carta1(event: MouseEvent): void {
    datos.numJ = 1;
    gotoAndStop(4);
}
carta1_btn.addEventListener(MouseEvent.CLICK, Carta1);
//=============================================================================================
//Funcion para el selector de jugadores de la carta 2 ("carta2_btn"), que es un boton.
//=============================================================================================
function Carta2(event: MouseEvent): void {
    datos.numJ = 2;
    gotoAndStop(4);
}
carta2_btn.addEventListener(MouseEvent.CLICK, Carta2);
```

Fotograma 4.

Primero que nada importamos la librería de eventos del mouse.

Después declaramos las variables y el Array que se muestran:

nombre: almacenara el nombre que ingresen los usuarios.
contNombres: es una variable que llevara el control de cuantos nombres se han ingresado.
aNombres: Array que almacenara los nombres ingresados.

```
2    import flash.events.MouseEvent;
3    //declaracion de variables
4    var nombre: String = "";
5    var contNombres: int = 0;
6    //declaracion de array que almace
7    var aNombres: Array = new Array;
```

Hacemos que el texto dinámico num_txt sea igual al contador de nombres +1, esto hará que cuando pida nombres vaya pidiendo el nombre del jugador n, el cual ira cambiando cada que se ingrese un nombre. Además hacemos invisibles algunos elementos que no queremos que se muestren aun:

```
8    //Mostramos el mensaje inicial de numer
9    num_txt.text = "" + (contNombres + 1);
10   //hacemos invisibles algunos elementos
11   jugar_btn.visible = false;
12   cuadromsj_2.visible = false;
13   mensaje_2.visible = false;
```

FUNCIÓN PARA VALIDAR Y GUARDAR LOS NOMBRES.

La función nombres validara los nombres y los almacenara en un Array, al mismo tiempo que se asegura de que solo se pueda jugar hasta que se ingresen todos los nombres.

Para esto lo primero que hacemos dentro de la función es un if, en el que comparamos diferentes estados en los que podría estar la caja de texto, los cuales consideramos nombres inválidos, por lo tanto si se cumple esta condición nos mostrara el mensaje “Ingrese un nombre”.

Si la condición anterior no se cumple, entonces tomaremos el nombre ingresado y lo agregaremos al Array aNombres, además de que incrementaremos en 1 al contador contNombres, para que después se actualice el número mostrado y se acerque a la cifra límite de nombres:

```
14   //funcion para validar y tomar los nombres
15   function nombres(event: MouseEvent): void {
16       //comparamos si la caja de texto esta vacia o tiene el mensaje de advertencia
17       if (nombre_txt.text == "" || nombre_txt.text == "Ingrese un nombre") {
18           //si se cumple mostramos el mensaje de advertencia
19           nombre_txt.text = "Ingrese un nombre";
20       } else {
21           //si no significa que se ingreso un nombre valido
22           nombre = nombre_txt.text;//tomamos el nombre
23           aNombres.push(nombre);//lo almacenamos en el array
24           contNombres++;//incrementamos una variable contadora
25           num_txt.text = "" + (contNombres + 1);//actualizamos el numero de jugador al para ingresar nombre
26           nombre_txt.text = "";//limpiamos la caja de texto
27       }
```

Después comparamos si contNombres es igual al número de jugadores que selecciono el usuario, si es así entonces se hacen invisibles la caja de texto y el botón para guardar nombres, y en su lugar colocamos un mensaje y el botón jugar, para que así ya no se puedan agregar más nombres y solo puedas ir a jugar.

```
30      if (contNombres == datos.numJ) {
31          cuadromsj.visible = false;
32          mensaje.visible = false;
33          nombre_txt.visible = false;
34          siguiente_btn.visible = false;
35          num_txt.visible = false;
36          jugar_btn.visible = true;
37          cuadromsj_2.visible = true;
38          mensaje_2.visible = true;
39      }
40  }
```

Y ya para terminar con el frame 4, tenemos la función que nos permitirá ir al frame del luego, en la cual solo colocamos un gotoAndStop al frame 2, en el cual se encuentra el juego:

```
41  //asignamos la funcion nombres al boton siguiente
42  siguiente_btn.addEventListener(MouseEvent.CLICK, nombres);
43  //funcion para ir al juego despues de ingresar todos los nombres
44  function irJuego(event: MouseEvent): void {
45      gotoAndStop(2);
46  }
47  //asignamos la funcion para ir al juego al boton jugar
48  jugar_btn.addEventListener(MouseEvent.CLICK, irJuego);
```

Fotograma 5.

En este frame lo que haremos es solamente mostrar los resultados de los jugadores, así como exportarlos a un documento txt si el usuario lo desea.

Para ello lo primero que hacemos es crear una variable objeto aux, a la cual después les creamos las variantes:

aux.puntos: almacenara temporalmente los puntos de la posición n+1
aux.nombres: almacenara temporalmente el nombre de la posición n+1
aux.tiempo: almacenara temporalmente el tiempo de la posición n+1

A su vez declaramos las variables contadoras l y k para los for que utilizaremos:

```
//variables objeto que actuan como switch al cambiar posiciones en los array
var aux: Object = new Object;
aux.puntos = 0;
aux.nombres = "";
aux.tiempo = 0;
//variables contadoras para los for
var l: int = 0;
var k: int = 0;
```

Después creamos una variable que usaremos para exportar los resultados al archivo txt:

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

METODO PARA ORENAR LOS ARRAYS DE FORMA QUE EL GANADOR QUEDE EN LA POSICION 0

(FRAME 5)

A continuación tenemos un for muy importante, este nos ayudara a ordenar los Arrays de forma que el jugador que junto más puntos quede en la posición 0, y el jugador con menos puntos quede en la última posición.

Para ello dentro del for tenemos otro for, el cual contiene la condición de que si los puntos en la posición actual son menos que los de la siguiente posición, entonces hago los cambios de la siguiente manera:

1.- a la variable auxiliar correspondiente le asignamos lo que tiene la posición actual
2.- a la posición actual le asignamos lo que tiene la siguiente posición
3.- a la siguiente posición le asignamos lo que contiene la variable auxiliar

Utilizamos las variables auxiliares para no perder ningún dato al momento de hacer los cambios de posiciones, si lo hiciéramos directamente perderíamos un dato.
Se realizan esos 3 pasos para los 3 diferentes Arrays aNombres, aPuntos y aTiempo:

```
for (l = 0; l < datos.numJ; l++) {
    //for que va comparando casilla por casilla
    for (k = 0; k < datos.numJ - 1; k++) {
        //si los puntos de la posicion actual son meno
        if (aPuntos[k] < aPuntos[k + 1]) {
            //con la ayuda de los aux hago los cambios
            //cambio de array puntos
            aux.puntos = aPuntos[k];
            aPuntos[k] = aPuntos[k + 1];
            aPuntos[k + 1] = aux.puntos;
            //cambio de array nombres
            aux.nombres = aNombres[k];
            aNombres[k] = aNombres[k + 1];
            aNombres[k + 1] = aux.nombres;
            //cambio de array minutos
            aux.tiempo = aTiempo[k];
            aTiempo[k] = aTiempo[k + 1];
            aTiempo[k + 1] = aux.tiempo;
        }
    }
}
```

NOTA: Hacemos un for dentro de un for ya que el interno comparara casilla por casilla, y el externo hará que se repita esta acción, esto hará que se ordene totalmente de forma descendente, ya que si lo hiciéramos una sola vez podrían ordenarse algunos valores pero otros no.

Una vez ordenados los Arrays por puntos, toca ordenarlos por tiempo en caso de que algunos hayan hecho los mismos puntos, así ganara el que los haya hecho en menor tiempo.

Para esto realizamos lo mismo que el método anterior, solo que aquí nuestra condición tiene más comparaciones. Comparamos si los puntos de la posición actual son iguales a los de la siguiente posición, y si el tiempo es mayor al tiempo de la siguiente posición, si se cumple significa que tienen el mismo puntaje, pero que el jugador de la siguiente posición lo hizo en menos tiempo, por lo tanto realizamos los cambios necesarios con la ayuda de las auxiliares:

```
for (l = 0; l < datos.numJ; l++) {
    //for que va comparando casilla por casilla
    for (k = 0; k < datos.numJ - 1; k++) {
        //si los puntos de la posicion actual son iguales a los de la siguiente y
        //entonces se cambian los valores
        if ((aPuntos[k] == aPuntos[k + 1]) && (aTiempo[k] > aTiempo[k + 1])) {
            //con la ayuda de los aux hago los cambios para no perder ningun dato
            //cambio de array puntos
            aux.puntos = aPuntos[k];
            aPuntos[k] = aPuntos[k + 1];
            aPuntos[k + 1] = aux.puntos;
            //cambio de array nombres
            aux.nombres = aNombres[k];
            aNombres[k] = aNombres[k + 1];
            aNombres[k + 1] = aux.nombres;
            //cambio de array minutos
            aux.tiempo = aTiempo[k];
            aTiempo[k] = aTiempo[k + 1];
            aTiempo[k + 1] = aux.tiempo;
        }
    }
}
```

Originalmente en nuestro frame tenemos 4 cuadros donde se colocaran los resultados, sin embargo no siempre serán los 4 los que se muestren, por lo tanto los hacemos invisibles todos desde el inicio:

```
59      resultado_1.visible = false;
60      result_1.visible = false;
61      resultado_2.visible = false;
62      result_2.visible = false;
63      resultado_3.visible = false;
64      result_3.visible = false;
65      resultado_4.visible = false;
66      result_4.visible = false;
```

METODO PARA MOSTRAR LOS RESULTADOS.

Lo siguiente es mostrar los resultados, como se dijo anteriormente no siempre se mostraran 4 resultados, por lo tanto utilizamos un switch en el cual comparamos el valor de datos.numJ que almacena el número de jugadores, de tal manera que dependiendo del número que tenga será el número de resultados y cuadros que se muestren.

En total son 4 resultados, en los que se va aumentando los objetos que se hacen visibles y los registros que se mostraran utilizando los Arrays y sus posiciones:

```
68  switch (datos.numJ) {
69      case 1: //caso 1 solo un puesto y un registro
70          resultado_1.visible = true;
71          result_1.visible = true;
72          result_1.text = "1.- " + aNombres[0] + "    " + aPuntos[0] + "pts.    " + aTiempo[0] + "s.";
73          break;
74      case 2: //caso 2 solo dos puestos y dos registros
75          resultado_1.visible = true;
76          result_1.visible = true;
77          result_1.text = "1.- " + aNombres[0] + "    " + aPuntos[0] + "pts.    " + aTiempo[0] + "s.";
78          resultado_2.visible = true;
79          result_2.visible = true;
80          result_2.text = "2.- " + aNombres[1] + "    " + aPuntos[1] + "pts.    " + aTiempo[1] + "s.";
81          break;
```

FUNCION PARA EXPORTAR LOS RESULTADOS.

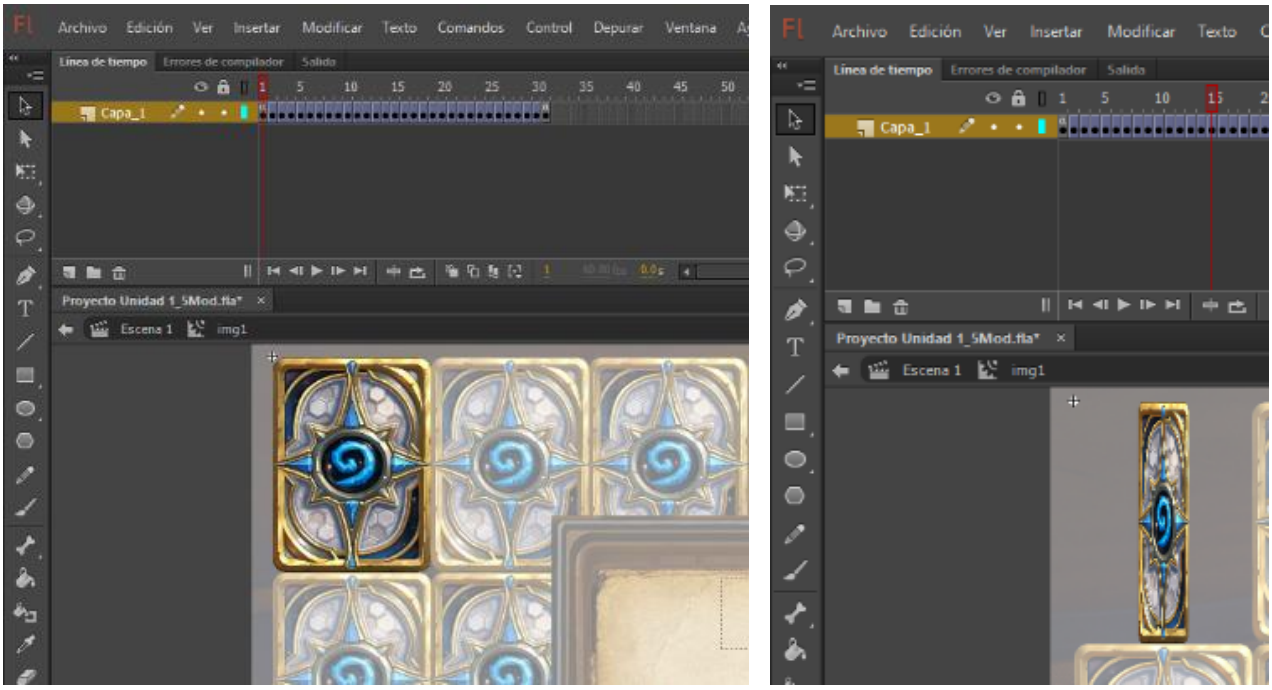
Por ultimo tenemos la función exportar. Esta función se encargara de exportar los resultados si el usuario presiona el botón exportar.

Lo hacemos dentro de un switch por la misma razón que lo anterior, que no siempre se exportaran 4 resultados, pero a diferencia del anterior aquí utilizamos la variable exportTXT declarada anteriormente , y lo que hacemos es asignarle lo mismo que mostramos como resultados, los valores de los Arrays en las diferentes posiciones, con un salto de línea después de cada registro para que se muestren en forma de lista, y obviamente solo se exportara la cantidad necesaria dependiendo del número de jugadores:

```
function exportar(event:MouseEvent):void{
    switch (datos.numJ) {
        case 1:
            exportTXT.save("1.- " + aNombres[0] + "   " + aPuntos[0] + "pts.   " + aTiempo[0] + "s. \r\n"+, "Resultados");
            break;
        case 2:
            exportTXT.save("1.- " + aNombres[0] + "   " + aPuntos[0] + "pts.   " + aTiempo[0] + "s. \r\n"+
            "2.- " + aNombres[1] + "   " + aPuntos[1] + "pts.   " + aTiempo[1] + "s. \r\n", "Resultados");
            break
        case 3:
            exportTXT.save("1.- " + aNombres[0] + "   " + aPuntos[0] + "pts.   " + aTiempo[0] + "s. \r\n"+
            "2.- " + aNombres[1] + "   " + aPuntos[1] + "pts.   " + aTiempo[1] + "s. \r\n"+
            "3.- " + aNombres[2] + "   " + aPuntos[2] + "pts.   " + aTiempo[2] + "s. \r\n", "Resultados");
            break;
        case 4:
            exportTXT.save("1.- " + aNombres[0] + "   " + aPuntos[0] + "pts.   " + aTiempo[0] + "s. \r\n"+
            "2.- " + aNombres[1] + "   " + aPuntos[1] + "pts.   " + aTiempo[1] + "s. \r\n"+
            "3.- " + aNombres[2] + "   " + aPuntos[2] + "pts.   " + aTiempo[2] + "s. \r\n"+
            "4.- " + aNombres[3] + "   " + aPuntos[3] + "pts.   " + aTiempo[3] + "s. \r\n", "Resultados");
            break;
    }
}
```

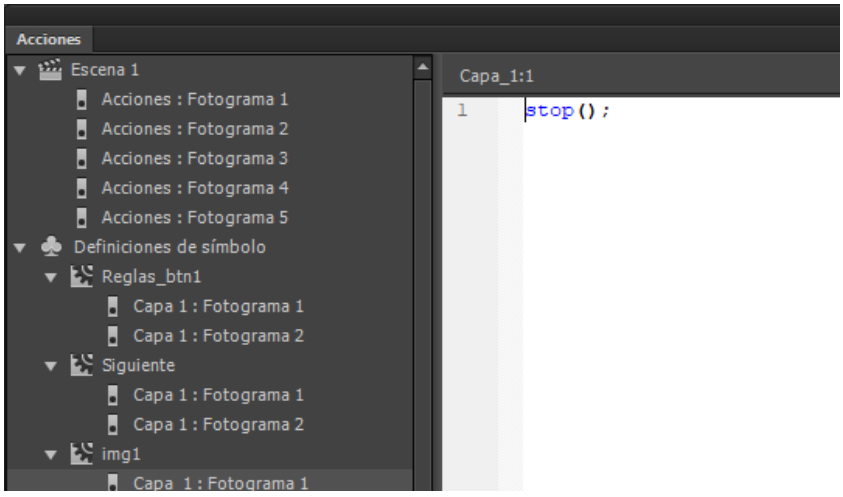
Interpolación de movimiento.

Para hacer una interpolación de movimiento deberemos de seleccionar la carta que queramos hacer la animación del volteado de carta.



Una vez hecho esto aparecerá otra línea de tiempo, en esa línea del tiempo pusimos varios fotogramas con una imagen de la carta haciéndose más pequeña, todo esto con una "interpolación clásica".

En el primer fotograma deberá de ir un "stop ();", como también en el último fotograma.



Conclusión.

En resumidas palabras, este es un juego de un memorama desarrollado en flash, con la colaboración de 2 personas, empleando cada quien su lógica para el desarrollo de este juego, e implementándola de la mejor manera.

En el transcurso del desarrollo de esta actividad aprendimos diferentes formas de ir desarrollando nuestras funciones e ideas que implementamos, nuevas animaciones que implementamos en cartas (Interpolaciones de movimiento), en la parte del diseño, como también en la parte de la lógica.

Tuvimos que acoplarnos en la parte de las ideas a desarrollar, y buscar la mejor manera de llevarlo a cabo, salvo que pensábamos diferente en la parte lógica, pudimos llevar a flote esta actividad.