
# PROYECTO 1

## Herramientas Multimedia

**MSI Mario Humberto Rodríguez Chávez**

**Alumnos:**

**Lumbreras Ruiz Jesús Alberto**

**Cordova Alanis Cruz Alejandro**

**HM - ITI-07122**

# Contenido



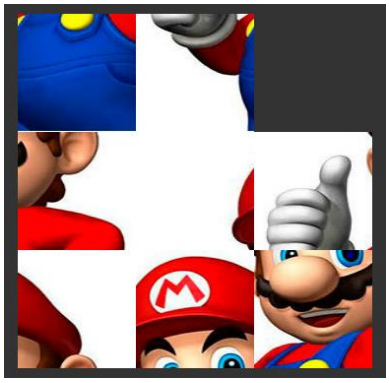
Integrantes: Jesús Alberto Lumbreras Ruiz
Cruz Alejandro Cordova Alanis

# Introducción

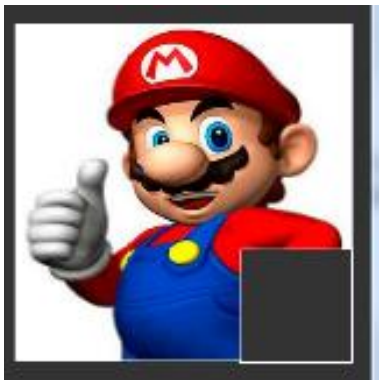
El juego consiste en armar un rompecabezas de 9 piezas, es juego es multijugador con un máximo de 5 jugadores y un mínimo de 2 jugadores. Este juego reta al usuario a reacomodar piezas deslizando hacia la derecha, izquierda, arriba, abajo, con el fin de que la imagen quede de una manera predefinida [imagen 1.2]. Las partes de este puzzle se muestran de manera aleatoria en pantalla [imagen 1.1].

Este tipo de juegos tiene distintas formas de llamarse o referirse a él como tal; algunos le dicen Puzzle, otros el juego de los números, rompecabezas deslizante, etc. Total el juego siempre será el mismo.

De la manera en cómo se escucha se podría decir que solo es mover piezas de allá para acá y al último se elige quién gana, pero detrás de todo eso existe un conocimiento puesto por parte de los alumnos Jesús Alberto y Cruz Alejandro, conocimiento generado en la materia Herramientas Multimedia; código echo totalmente en Lenguaje ActionScript 3.0 en el Programa Adobe Animate cc, el cual en las siguientes páginas se explicará más a detalle.



[Imagen 1.1]



[Imagen 1.2]

# DESARROLLO:

## Fotograma 1:

```
1  import flash.events.MouseEvent;
2  import fl.transitions.Tween;
3  import fl.transitions.TweenEvent;
4  import fl.transitions.easing.*;
5  import flash.events.Event;
```

Estas son las librerías que utilizaremos en nuestro primer fotograma, los números 1,5 es para eventos que se harán al utilizar el mouse, los números 2,3,4 se utiliza para los Tweens de algunos textos dinámicos y botones.

```
8  var unil_mc:Tween = new Tween (uni_mc, "y", Elastic.easeInOut, 400, 32, 3, true);
9  var upvl_mc:Tween = new Tween (upv_mc, "y", Elastic.easeOut, 400, 48, 2, true);
10 var profesorl_mc:Tween = new Tween (profesor_mc, "x", Bounce.easeInOut, 1, 218, 3, true);
11 var nprofesorl_mc:Tween = new Tween (nprofesor_mc, "x", Bounce.easeOut, 550, 85, 3, true);
12 var alumnos1_mc:Tween = new Tween (alumnos_mc, "x", Elastic.easeOut, 1, 227, 3, true);
13 var alumnolx_mc:Tween = new Tween (alumnol_mc, "x", Elastic.easeOut, 550, 108, 3, true);
14 var alumno2x_mc:Tween = new Tween (alumno2_mc, "x", Elastic.easeOut, 1, 116, 3, true);
15 var grupol_mc:Tween = new Tween (grupo_mc, "x", Bounce.easeOut, 550, 205, 2, true);
16 var entrarl_btn:Tween = new Tween (entrar_btn, "y", Elastic.easeInOut, 10, 365, 3, true);
```

Son las variables que contienen el movimiento (Tween) de cada uno de mis textos dinámicos de mi portada, lo que hace es crear una variable Tween la cual hará según el eje y el tipo de efecto en las coordenadas indicadas un movimiento de entrada el cual se ejecutará una vez inicie nuestro fotograma.

```
//FUNCION PARA EL BOTON DE ENTRAR
function entrar(Event:MouseEvent):void{
    gotoAndStop(2);
}
entrar_btn.addEventListener(MouseEvent.CLICK,entrar);
```

Es la función para el botón de entrar la cual, al dar clic en nuestro botón, va a irse automáticamente a el fotograma 2 en donde esta la interfaz de nuestro juego, la última línea de código manda llamar la función de arriba, la cual cumple con lo anteriormente dicho.

```
//VARIABLES DE TIPO CONTADOR
var cont:int;
cont=0;
var contPlayer:int;
contPlayer=0;
```

Variables para los contadores que utilizaremos en los siguientes fotogramas, las inicializamos en el fotograma 1 para no tener problemas al momento de regresar o de desplazarnos por los otros fotogramas. La variable cont hace el conteo para pedir el nombre de los jugadores según el numero de jugadores que se escribió. La variable contPlayer es la que nos guarda todo en nuestros 3 arrays de nombre de jugador, tiempo y movimientos.

# Fotograma 2:

```
29  //ARRAYS PARA GUARDAR LOS DATOS DE CADA JUGADOR (TIEMPO Y MOVIMIENTOS)
30  var TimerArray:Array = new Array();
31  var MovArray:Array = new Array();
32  var jugadores:Array = new Array();
33
34  var bandera:Boolean;
35  bandera=false;
36  var banderaPlayer:Boolean;
37  banderaPlayer=false;
38
39  //VARIABLES QUE UTILIZAREMOS PARA SACAR EL JUGADOR GANADOR
40  var TimeWin:int = 0;
41  var PlayerWin:String;
```

Declaramos nuestros 3 arrays en donde guardaremos nuestros nombres, tiempos y movimientos de cada jugador, y declararemos una variable de tipo bandera para utilizarla después en una validación. Las 2 últimas variables se utilizarán para comparar y almacenar al mejor jugador al que armo el puzzle en menor tiempo.

```
1
2   import flash.events.MouseEvent;
3   //VARIABLES QUE UTILIZAREMOS PARA LA VERIFICACION DE LOS DATOS QUE INGRESARA EL USUARIO DE ENTRADA
4   var Player:String;
5   var numpl:Number;
6   //VARIABLE PARA GUARDAR EL TIMER Y POSTERIORMENTE COMPARARLO
7   var sg: int = 0
8   //VARIABLES PARA EL TIMER
9   var tmp: int = 0;
10  var cont1: int = 0;
11  var min: int = 0;
12  var timer: Timer = new Timer(1000, cont1++);
13
14  //COLOCAREMOS EN INVISIBLE LAS CAJAS DE TEXTO PARA COLOCAR EL NOMBRE DEL "N" JUGADOR
15  players_txt.visible=false;
16  player_txt.visible=false;
17  ingresar1_btn.visible=false;
```

Utilizaremos nuevamente la librería del MouseEvent, utilizaremos las 2 primeras variables para el nombre del jugador y el numero de jugadores. La 3 variable se usará para ir guardando los segundos del timer de cada jugador para después utilizarla y comparar con los demás y así sacar al mejor jugador y las ultimas 4 variables se utilizan para el timer de nuestro fotograma de juego, y por último colocamos nuestros 2 textos y el botón de el ingreso de cada nombre del jugador, esto se hace para que no aparezcan hasta que se coloque el número de jugadores

```
19  //FUNCION PARA EL BOTON DE INGRESAR DEL NUMERO DE JUGADORES
20  function IngresarNP(Event:MouseEvent):void{
21      numpl=Number(numpl_txt.text);
22      if ((isNaN(numpl)) || (numpl==0)){
23          error_txt.text="INGRESA UN NUMERO";
24          bandera=true;
25      }
26      if ((numpl<2) || (numpl>5)){
27          //SI EL NUMERO DE PLAYERS NO ESTA EN RANGO DE 1 A 5 ENTONCES MANDAREMOS UN MENSAJE DE ERROR
28          error_txt.text="INGRESA UN NUMERO DE 2 A 5";
29          bandera=true;
30      }
31      //SI LA BANDERA ESTUVO EN FALSE ENTONCES NO HUBO PROBLEMAS EN LA VALIDACION
32      if (bandera==false){
33          error_txt.text="";
34          //PONDREMOS EN INVISIBLE EL NUMERO DE PLAYERS
35          numpl_txt.visible=false;
36          NPlayers_txt.visible=false;
37          ingresar_btn.visible=false;
38          //PONEMOS LOS TEXTOS EN VISIBLE PARA PODER INGRESAR EL NOMBRE DE LOS JUGADORES
39          players_txt.visible=true;
40          player_txt.visible=true;
41          ingresar1_btn.visible=true;
42          players_txt.text="Ingresa el nombre del 1 jugador";
43      }
44      bandera=false;
45  }
46  ingresar_btn.addEventListener(MouseEvent.CLICK,IngresarNP);
```

Esta función se encarga de validar el numero de jugadores, se almacena en numpl lo que tenga nuestra caja de texto numpl_txt y lo primero es validar si es una letra o si está vacío si es así entonces mandamos un mensaje de error y encenderemos nuestra variable de bandera para que no se pueda continuar con lo demás, si lo de arriba no se cumplió entonces validamos ahora que si se coloca un numero menor a 2 o mayor a 5 e igualmente si esto se cumple entonces mostramos error y encendemos nuestra bandera.

Si la bandera esta en falso, es decir que nunca se encendió (se enciende si hay errores en la validación del numero de jugadores) entonces lo que se hará es poner en invisible lo que utilizamos para el numero de jugadores y pondremos en visible lo que utilizaremos para ingresar los jugadores.

```
function IngresarNombres(Event:MouseEvent):void{
    Player=player_txt.text;
    if (Player==""){
        error_txt.text="INGRESA UN NOMBRE";
        bandera=true;
    }
    if (Number(Player)){
        error_txt.text="SOLO LETRAS";
        bandera=true;
    }
    if (bandera==false){
        error_txt.text="";
        player_txt.text="";
        jugadores[cont]=Player;
        players_txt.text="Ingresa el nombre del "+(cont+2)+" jugador";
        cont=cont+1;
    }
    bandera=false;
    if (cont==numpl){
    players_txt.text="Jugador "+jugadores[contPlayer];
    NamePl_txt.text="Elige puzzle";
    player_txt.visible=false;
    ingresar1_btn.visible=false;
    banderaPlayer=true;
}
}
ingresar1_btn.addEventListener(MouseEvent.CLICK,IngresarNombres);
```

Esta función lo que hace es guardar en la variable (Player) lo que tenga nuestra caja de texto (player_txt) y posteriormente se valida que no se deje en vacío y si es así entonces manda un mensaje de error y enciende nuestra bandera y esto lo hace también para verificar que no se coloquen números.

Si la bandera esta apagada (no hubo errores) entonces se guarda en nuestro array jugadores el nombre del jugador y se manda un msg para que se ingrese el siguiente jugador.

Por ultimo se valida si nuestro contador llega al numero de jugadores que se ingreso al inicio entonces manda un msg de que el jugador 1 escoja el puzzle y nuestras cajas de texto y nuestro botón se ponen en invisible para que no se pueda mover nada y solo se pueda elegir el puzzle.

```
//FUNCION PARA LA SELECCION DE LA IMAGEN DE MARIO
function GoMario(Event:MouseEvent):void{
    if (banderaPlayer==true)
    {
        gotoAndStop(3);

    }
}
mario_btn.addEventListener(MouseEvent.CLICK,GoMario);
```

Esta función hace que al dar clic en la imagen del puzle que elige el jugador, se va al fotograma en donde se encuentra este puzle y empieza el juego.

# Fotograma 3

```
6   //VARIABLE QUE UTILIZAREMOS COMO ACUMULADOR PARA GUARDAR LOS MOVIMIENTOS DE CADA JUGADOR
7   var mov:int;
8   mov=0;
9
10  player1_txt.text=jugadores[contPlayer];
11  var BanderaNext:Boolean;
12  BanderaNext=false;
13  var banderaMario:Boolean;
14  banderaMario=false;
15
16  //VAMOS A PONER EN INVISIBLE TODO LO QUE ES PARA CUANDO SE RESUELVA EL PUZZLE O SE RINDA
17  //Y SIGA EL OTRO JUGADOR
18  nextMario_txt.visible=false;
19  nextMario1_txt.visible=false;
20  ganarMario_mc.visible=false;
21  botonMario_btn.visible=false;
22  defeatmario_mc.visible=false;
```

La variable mov la utilizaremos como acumulador de los movimientos de el jugador en turno, se activa cada que se da clic en una pieza de las imágenes y esta se mueve, vamos a colocar el nombre del jugador en turno y vamos a declarar 2 variables tipo bandera que nos ayudaran posteriormente al hacer validaciones.

Después colocaremos en invisible todo lo que se hará cuando el jugador resuelva el puzzle o se rinda.

```
24  //ARRANCAR EL TIEMPO
25  timer.start();
26  function tiempo1(tiempoevent: TimerEvent): void {
27      tmp++;
28      sg++;
29      if(tmp>59){
30          tmp=0;
31          min++;
32      }
33      if(tmp<10){
34          tiempo1_txt.text =min+":"+"0"+ tmp + " s.";
35      }
36      else{
37          tiempo1_txt.text =min+":"+ tmp + " s.";
38      }
39  }
40  timer.addEventListener(TimerEvent.TIMER, tiempo1);
```

Vamos a arrancar el timer y crearemos la función la cual va a llevar el conteo del timer del jugador en turno, nuestras variables tmp y sg se irán aumentando si el timer esta encendido y después validamos para cuando llegue a 59 segundos entonces cambie y muestre el minuto y vamos a inicializar nuestro tmp para que siga aumentando los segundos y mostrando el tiempo en nuestro tiempo1_txt.

```
43  //ACCIONES PARA GENERAR NUMEROS ALEATORIOS
44  var numl:Array = new Array();
45  var il:uint;
46  var jl:uint;
47  var nol1:uint;
48  var nol2:uint;
49  var banl:Boolean;
50  nol2=Math.random()*9;
51  numl[0]=nol2;
52  //CICLO PARA DAR NUMEROS DEL 1 AL 9 SIN QUE SE REPITAN Y PODER ACOMODAR LA PARTE DE LA IMAGEN SEGUN EL NUMERO RANDOM
53  for(il=1; il<9; il++){
54      banl=false;
55      nol2=Math.random()*9;
56      numl[il]=nol2;
57      for(jl=0; jl<il;jl++){
58          if(numl[jl]==numl[il]){
59              banl=true;
60              jl=il;
61          }
62      }
63      if(banl==true){
64          il=il-1;
65      }
66  }
```

Aquí declaramos 6 variables para la parte de generar números aleatorios, el primero es un array en donde guardaremos 9 números aleatorios los cuales son del 0 al 8 y los cuales no se pueden repetir, para eso utilizaremos un ciclo anidado For, el cual genera un numero aleatorio y después hace un barrido con otro ciclo para saber si es un numero repetido y si es así se enciende la bandera lo que hace que a nuestro primer contador se le reste 1 para que vuelva a guardar otro numero aleatorio en esa misma posición para evitar números repetidos y esto se hace hasta obtener nuestros 9 números.

```
68  //ACCIONES PARA EL ACOMODO DE LAS IMAGENES
69  var imagenl:Array= new Array();
70  //ARRAY PARA GUARDAR LAS INSTANCIAS DE CADA UNA DE LAS PARTES DE LAS IMAGENES
71  imagenl[0]=mario1;
72  imagenl[1]=mario2;
73  imagenl[2]=mario3;
74  imagenl[3]=mario4;
75  imagenl[4]=mario5;
76  imagenl[5]=mario6;
77  imagenl[6]=mario7;
78  imagenl[7]=mario8;
79  imagenl[8]=mario9;
```

Declararemos un array el cual nos va a almacenar las instancias de cada una de las partes de nuestro rompecabezas esto se hace para posteriormente utilizar un ciclo el cual vamos a ver a continuación.

```
81  for(il=0; il<9; il++){
82      //SEGUN EL NUMERO RANDOM SE ASIGNA UNA COORDENADA A CADA DE LAS PARTES DE LAS IMAGENES
83      switch(numl[il]){
84          case 0:
85              imagenl[il].x=79;
86              imagenl[il].y=79;
87          break;
88          case 1:
89              imagenl[il].x=199;
90              imagenl[il].y=79;
91          break;
92          case 2:
93              imagenl[il].x=319;
94              imagenl[il].y=79;
95          break;
96          case 3:
97              imagenl[il].x=79;
98              imagenl[il].y=199;
99          break;
100         case 4:
101             imagenl[il].x=199;
102             imagenl[il].y=199;
103         break;
104         case 5:
105             imagenl[il].x=319;
106             imagenl[il].y=199;
107         break;
108         case 6:
109             imagenl[il].x=79;
110             imagenl[il].y=319;
111         break;
112         case 7:
113             imagenl[il].x=199;
114             imagenl[il].y=319;
115         break;
116         case 8:
117             imagenl[il].x=319;
118             imagenl[il].y=319;
119         break;
120     }
```

Utilizaremos un ciclo For el cual ira del 0 al 8 aumentando de 1 en 1 el cual utilizamos en un switch case el cual hace que según sea el contador es el numero random que se guardo anteriormente en nuestro array (num1) y es la situación que se elige en estas 9 posibles situaciones se encuentran las 9 posiciones en las que puede quedar una imagen el contador (i1) es el que hace todo esto, escoge una situación y en base a la instancia almacenada en nuestro otro array elige la imagen, esto se hace hasta que las 9 imágenes queden totalmente colocadas y acomodadas en nuestras 9 posiciones.

```
123    //FUNCIONES PARA LAS IMAGENES
124    mario1.addEventListener(MouseEvent.CLICK, mover1);
125    mario2.addEventListener(MouseEvent.CLICK, mover2);
126    mario3.addEventListener(MouseEvent.CLICK, mover3);
127    mario4.addEventListener(MouseEvent.CLICK, mover4);
128    mario5.addEventListener(MouseEvent.CLICK, mover5);
129    mario6.addEventListener(MouseEvent.CLICK, mover6);
130    mario7.addEventListener(MouseEvent.CLICK, mover7);
131    mario8.addEventListener(MouseEvent.CLICK, mover8);
```

Estas funciones son para las 8 imágenes que vamos a mover y a intercambiar lugar con nuestra imagen vacía, aquí estamos llamando a estas funciones esto ocurre al dar clic en ellas

```
132  //FUNCIONES PARA INTERCAMBIAR LAS IMAGENES Y GENERAR UN MOVIMIENTO
133  //FUCNION PARA LA IMAGEN 1
134  function moverl(e:MouseEvent){
135      //ESTA CONDICION ES PARA SABER SI A LA IZQUIERDA DE NUESTRA IMAGEN SE ENCUENTRA EL VACIO Y SI ES ASI SE
136      //MUEVE HACIA LA IZQUIERDA INTERCAMBIANDO POSICION CON EL VACIO
137      if(mariol.x-120==mario9.x && mariol.y==mario9.y){
138          var comariol:Tween= new Tween(mariol, "x", None.easeOut, mariol.x, mario9.x, 0.5, true);
139          var comario2:Tween= new Tween(mario9, "x", None.easeOut, mario9.x, mariol.x, 0.5, true);
140          mov++;
141          movimientos_txt.text=mov+"";
142      }
```

Esta función es para el movimiento de la primer imagen es mouse event porque actúa al darle clic a ese pedazo de nuestra imagen partida, lo que hace la condición que se ve en pantalla es restarle a nuestro pedazo de imagen el ancho de la propia imagen (120) en el eje de las “x” para comprobar si es igual a la instancia de la imagen “vacía” y también se verifica que nuestro eje “y” sea igual al vacío si estas 2 condiciones se cumplen entonces quiere decir que a la izquierda de nuestra imagen se encuentra el vacío e intercambian lugares entre si mediante 2 tweens y se aumenta nuestro contador de movimientos y se muestra en pantalla.

```
143      //ESTA CONDICION ES PARA SABER SI A LA DERECHA DE NUESTRA IMAGEN SE ENCUENTRA EL VACIO Y SI ES ASI SE
144      //MUEVE HACIA LA DERECHA INTERCAMBIANDO POSICION CON EL VACIO
145      if(mariol.x+120==mario9.x && mariol.y==mario9.y){
146          comariol= new Tween(mariol, "x", None.easeOut, mariol.x, mario9.x, 0.5, true);
147          comario2= new Tween(mario9, "x", None.easeOut, mario9.x, mariol.x, 0.5, true);
148          mov++;
149          movimientos_txt.text=mov+"";
150      }
```

Esta condición hace lo mismo que la pasada, pero ahora aumenta en eje de “x” esto se hace para verificar si a la derecha se encuentra el vacío (es lo contrario a el anterior que verificaba si a la izquierda estaba el vacío).

```
151         //ESTA CONDICION ES PARA SABER SI ARRIBA DE NUESTRA IMAGEN SE ENCUENTRA EL VACIO Y SI ES ASI SE
152         //MUEVE HACIA LA ARRIBA INTERCAMBIANDO POSICION CON EL VACIO
153         if(mariol.y-120==mario9.y && mariol.x==mario9.x){
154             comariol= new Tween(mariol, "y", None.easeOut, mariol.y, mario9.y, 0.5, true);
155             comario2= new Tween(mario9, "y", None.easeOut, mario9.y, mariol.y, 0.5, true);
156             mov++;
157             movimientos_txt.text=mov+"";
158             trace("abajo");
159         }
```

Esta condición también es para verificar pero esta es para saber si arriba de nuestra imagen , se resta la altura de nuestra imagen en el eje de las “y” y también se verifica el eje de las “x” esto para saber si arriba se encuentra el vacío y si es así, se intercambian posiciones.

```
    //ESTA CONDICION ES PARA SABER SI A ABAJO DE NUESTRA IMAGEN SE ENCUENTRA EL VACIO Y SI ES ASI SE
    //MUEVE HACIA ABAJO INTERCAMBIANDO POSICION CON EL VACIO
    if(mariol.y+120==mario9.y && mariol.x==mario9.x){
        comariol= new Tween(mariol, "y", None.easeOut, mariol.y, mario9.y, 0.5, true);
        comario2= new Tween(mario9, "y", None.easeOut, mario9.y, mariol.y, 0.5, true);
        mov++;
        movimientos_txt.text=mov+"";
    }
}
```

Esta es para verificar si debajo de la imagen se encuentra el vacío, se le suma la altura en el eje de las “y” y se verifica el eje de las “x” para saber si abajo esta el vacío y si es así se intercambian posiciones.

```
//FUNCION PARA CUANDO EL USUARIO RESUELVA EL PUZZLE
function terminarMario(e:Event){
    if (banderaMario==false){
        if((mariol.x==79 && mariol.y==79)&&(mario2.x==199 && mario2.y==79)){
            banderaMario=true;
            //VAMOS A GUARDAR EL TIMER Y EL MOVIMIENTO DEL JUGADOR EN NUESTROS ARRAYS
            TimerArray[contPlayer]=tiempo1_txt.text;
            MovArray[contPlayer]=movimientos_txt.text;
            //VAMOS A HACER LA COMPARACION DE LOS TIEMPOS DE LOS JUGADORES PARA SACAR EL MEJOR JUGADOR
            if (contPlayer<1){
                TimeWin=sg;
                PlayerWin=jugadores[contPlayer];
            }
            else{
                if (sg<TimeWin){
                    TimeWin=sg;
                    PlayerWin=jugadores[contPlayer];
                }
            }
```

Esta función es para cuando se termina el juego, se condiciona para saber si las imágenes están en las posiciones adecuadas y si esto se cumple entonces se guarda en los array el tiempo y los movimientos del jugador, y se condiciona para sacar el jugador más rápido al armar el puzle esto es para al último mostrar al ganador.

```
            //AUMENTAREMOS EL CONTADOR QUE MUESTRA EL NOMBRE DEL SIGUIENTE JUGADOR
            contPlayer++;
            //CONDICIONAREMOS SI YA JUGARON LOS JUGADORES INGRESADOS ENTONCES SE IRA AL FRAME DE LAS PUNTUACIONES
            if (contPlayer==numpl){
                gotoAndStop(8);
                timer.stop();
            }
            //VAMOS A PONER EN INVISIBLE TODO LO QUE CONTENGA EL FRAME PARA QUE PUEDA APARECER LO DEL SIGUIENTE FRAME
            if (contPlayer<numpl){
                mario1.visible=false;
                mario2.visible=false;
                mario3.visible=false;
                mario4.visible=false;
                mario5.visible=false;
                mario6.visible=false;
                mario7.visible=false;
                mario8.visible=false;
                mario9.visible=false;
                Player1_txt.visible=false;
                player1_txt.visible=false;
                Movimientos_txt.visible=false;
                movimientos_txt.visible=false;
                tiempo1_txt.visible=false;
                Tiempo1_txt.visible=false;
                rendirse1_btn.visible=false;
                //VAMOS A PONER VISIBLES EL NOMBRE DEL SIGUIENTE JUGADOR Y UN CONTADOR DE EL 3 AL 1 PARA MANDARLO AL FRAI
                //Y MOSTRAREMOS EN PANTALLA EL NOMBRE EL SIGUIENTE JUGADOR
                nextMario1_txt.text=jugadores[contPlayer];
                nextMario_txt.visible=true;
                nextMario1_txt.visible=true;
                ganarMario_mc.visible=true;
                timer.stop();
                botonMario_btn.visible=true;
            }
```

Se aumenta el contador para al momento de salir, se muestre el nombre del siguiente jugador y no se tenga problemas al momento de guardar en los arrays, se condiciona para saber si el contador ya igualo al número de jugadores y si esto es así entonces se va al fotograma de los resultados, si es menor entonces se colocan nuestras partes de la imagen en invisible al igual que las cajas de texto del nombre, tiempo y movimientos y se pone en pantalla botón para el siguiente jugador así como el nombre.

```
//CREAREMOS LA FUNCION PARA CUANDO SE DE CLIC EN EL BOTON DE NEXT Y SE VAYA AL FRAME DE SELECCION DE PUZZLE
function Next(Event:MouseEvent):void{
        gotoAndStop(2);
        //VAMOS A PONER INVISIBLE LOS TXT DINAMICOS PARA INGRESAR DATOS Y TAMBIEN EL BOTON DE INGRESAR
        //PARA QUE NO APAREZCAN AL MOMENTO DE REGRESAR AL FRAME PARA QUE EL OTRO JUGADOR SELECCIONE SU PUZZLE
        player_txt.visible=false;
        ingresar1_btn.visible=false;
        NPlayers_txt.visible=false;
        ingresar_btn.visible=false;
        numpl_txt.visible=false;
        players_txt.visible=true;
        players_txt.text="Jugador "+jugadores[contPlayer]+" elige puzzle";
}
botonMario_btn.addEventListener(MouseEvent.CLICK,Next);
```

Esta función es para el botón de siguiente jugador, este aparece cuando se rinden o cuando se arma el puzle lo que hace es poner en invisible todo lo que sale en nuestro menú y en un texto escribir el nombre del jugador a elegir el puzle.

```
431  //SE CREARÁ LA FUNCIÓN PARA EL BOTÓN RENDIRSE EN CASO DE QUE EL USUARIO NO PUEDA REALIZAR EL PUZZLE
432  function merindoM(e:MouseEvent){
433      banderaMario=true;
434      //VAMOS A GUARDAR EL TEXTO DE SE RINDIÓ EN NUESTROS ARRAYS
435      TimerArray[contPlayer]="Rendido";
436      MovArray[contPlayer]="Rendido";
437      //SI ES EL PRIMER JUGADOR EL QUE SE RINDE VAMOS A GUARDAR EN TIMEWIN UN VALOR MUY ALTO PARA SIMULAR QUE P
438      if (contPlayer<1){
439          TimeWin=10000;
440      }
441      if (contPlayer>=1){
442          TimeWin=TimeWin;
443      }
444      //AUMENTAREMOS EL CONTADOR QUE MUESTRA EL NOMBRE DEL SIGUIENTE JUGADOR
445      contPlayer++;
446      //CONDICIONAREMOS SI YA JUGARON LOS JUGADORES INGRESADOS ENTONCES SE IRA AL FRAME DE LAS PUNTUACIONES
447      if (contPlayer==numpl){
448          gotoAndStop(8);
449          timer.stop();
450      }
```

Esta función es para el botón de rendirse, si se da clic en este botón entonces actuara esta función se guardara en los arrays de tiempo y movimientos la palabra “Rendido”, después creamos una condición  para en dado caso de que sea el primer jugador y se rinda se guarde en nuestra variable de ganador con un valor demasiado alto para que así se pueda sustituir más fácil con otro jugador ya que entre menos tiempo es el mejor jugador, igual se condiciona para saber si ya se igualo el contador de jugadores con el número de jugadores, si es así entonces se va al fotograma 8.

```
451         //VAMOS A PONER EN INVISIBLE TODO LO QUE CONTENGA EL FRAME PARA QUE PUEDA APARECER LO DEL SIGUIENTE FRAME
452         if (contPlayer<numpl){
453             mario1.visible=false;
454             mario2.visible=false;
455             mario3.visible=false;
456             mario4.visible=false;
457             mario5.visible=false;
458             mario6.visible=false;
459             mario7.visible=false;
460             mario8.visible=false;
461             mario9.visible=false;
462             Player1_txt.visible=false;
463             player1_txt.visible=false;
464             Movimientos_txt.visible=false;
465             movimientos_txt.visible=false;
466             tiempo1_txt.visible=false;
467             Tiempo1_txt.visible=false;
468             rendirse1_btn.visible=false;
469             //VAMOS A PONER VISIBLES EL NOMBRE DEL SIGUIENTE JUGADOR Y UN CONTADOR DE EL 3 AL 1 PARA MANDARLO AL FRAM
470             //Y MOSTRAREMOS EN PANTALLA EL NOMBRE EL SIGUIENTE JUGADOR
471             nextMario1_txt.text=jugadores[contPlayer];
472             nextMario_txt.visible=true;
473             nextMario1_txt.visible=true;
474             defeatmario_mc.visible=true;
475             timer.stop();
476             botonMario_btn.visible=true;
477         }
478     }
479 
480     rendirse1_btn.addEventListener(MouseEvent.CLICK, merindoM);
```

Si el contador de jugadores aun no iguala a nuestro número de jugadores entonces se pone en falso todos nuestros pedazos de imágenes también nuestras cajas de texto con el nombre los movimientos y el tiempo del jugador, y se pone en pantalla una imagen de derrota y se activa el botón para seguir con el siguiente jugador y se detiene el timer.

Básicamente esto anterior es lo que hacemos para cada una de nuestros puzles, solo cambian las variables.

# Fotograma 8

```
7   //TWEENS PARA LAS CAJAS DE TEXTO DONDE SE MUESTRAN LOS RESULTADOS
8   var Result_mc:Tween = new Tween (Result_txt, "x", Bounce.easeOut, 1, 182, 2, true);
9   var bestPlayer_mc:Tween = new Tween (bestPlayer_txt, "x", Elastic.easeOut, 1, 93, 2, true);
10  var BestPlayer_mc:Tween = new Tween (BestPlayer_txt, "x", Elastic.easeOut, 550, 273, 2, true);
11
12  var Pla_mc:Tween = new Tween (Pla_txt, "y", Regular.easeOut, 1, 108, 2, true);
13  var Pla1_mc:Tween = new Tween (Pla1_txt, "y", Regular.easeOut, 1, 149, 2, true);
14  var Pla2_mc:Tween = new Tween (Pla2_txt, "y", Regular.easeOut, 1, 198, 2, true);
15  var Pla3_mc:Tween = new Tween (Pla3_txt, "y", Regular.easeOut, 400, 245, 2, true);
16  var Pla4_mc:Tween = new Tween (Pla4_txt, "y", Regular.easeOut, 400, 289, 2, true);
17  var Pla5_mc:Tween = new Tween (Pla5_txt, "y", Regular.easeOut, 400, 331, 2, true);
18
19  var Tim_mc:Tween = new Tween (Tim_txt, "y", Regular.easeOut, 1, 108, 2, true);
20  var Tim1_mc:Tween = new Tween (Tim1_txt, "y", Regular.easeOut, 1, 149, 2, true);
21  var Tim2_mc:Tween = new Tween (Tim2_txt, "y", Regular.easeOut, 1, 198, 2, true);
22  var Tim3_mc:Tween = new Tween (Tim3_txt, "y", Regular.easeOut, 400, 245, 2, true);
23  var Tim4_mc:Tween = new Tween (Tim4_txt, "y", Regular.easeOut, 400, 289, 2, true);
24  var Tim5_mc:Tween = new Tween (Tim5_txt, "y", Regular.easeOut, 400, 331, 2, true);
25
26  var Mov_mc:Tween = new Tween (Mov_txt, "y", Regular.easeOut, 1, 108, 2, true);
27  var Mov1_mc:Tween = new Tween (Mov1_txt, "y", Regular.easeOut, 1, 149, 2, true);
28  var Mov2_mc:Tween = new Tween (Mov2_txt, "y", Regular.easeOut, 1, 198, 2, true);
29  var Mov3_mc:Tween = new Tween (Mov3_txt, "y", Regular.easeOut, 400, 245, 2, true);
30  var Mov4_mc:Tween = new Tween (Mov4_txt, "y", Regular.easeOut, 400, 289, 2, true);
31  var Mov5_mc:Tween = new Tween (Mov5_txt, "y", Regular.easeOut, 400, 331, 2, true);
```

Se ponen las variables de los tweens de cada texto que aparecerá en nuestro fotograma final de resultados

```
35  //VAMOS A MOSTRAR EL NOMBRE DEL JUGADOR CON MEJOR TIEI
36  //CONDICIONAREMOS PARA SABER SI TODOS LOS JUGADORES SI
37  if (TimeWin==10000){
38      BestPlayer_txt.text="Todos Se Rindieron";
39  }
40  else
41  {
42      BestPlayer_txt.text=PlayerWin;
43  }
```

Esta condición es para cuando al ultimo nuestra variable que guarda el tiempo récord tiene 10000 esto quiere decir que todos se rindieron y entonces mandara un mensaje de que todos se rindieron y no habrá un ganador, si no se cumple entonces se muestra lo que tenga guardado nuestra variable del jugador ganador.

```
44    //ESTADISTICAS DEL PRIMER JUGADOR
45    Pla1_txt.text=jugadores[0];
46    Tim1_txt.text=TimerArray[0];
47    Mov1_txt.text=MovArray[0];
48
49    //ESTADISTICAS DEL SEGUNDO JUGADOR
50    Pla2_txt.text=jugadores[1];
51    Tim2_txt.text=TimerArray[1];
52    Mov2_txt.text=MovArray[1];
53
54    //ESTADISTICAS DEL TERCER JUGADOR
55    if (numpl==3){
56        Pla3_txt.text=jugadores[2];
57        Tim3_txt.text=TimerArray[2];
58        Mov3_txt.text=MovArray[2];
59    }
```

Aquí se muestra lo que tenemos en cada uno de nuestros arrays, es para mostrar las estadísticas de los jugadores, cuando son mas de 2 jugadores entonces utilizamos una condición para darle un valor a nuestros textos y si esto no se cumple entonces no guarda nada y no hay problemas por datos nulos.

```
61    //ESTADISTICAS DEL CUARTO JUGADOR
62    if (numpl==4){
63        Pla3_txt.text=jugadores[2];
64        Tim3_txt.text=TimerArray[2];
65        Mov3_txt.text=MovArray[2];
66
67        Pla4_txt.text=jugadores[3];
68        Tim4_txt.text=TimerArray[3];
69        Mov4_txt.text=MovArray[3];
70    }
71    //ESTADISTICAS DEL QUINTO JUGADOR
72    if (numpl==5){
73        Pla3_txt.text=jugadores[2];
74        Tim3_txt.text=TimerArray[2];
75        Mov3_txt.text=MovArray[2];
76
77        Pla4_txt.text=jugadores[3];
78        Tim4_txt.text=TimerArray[3];
79        Mov4_txt.text=MovArray[3];
80
81        Pla5_txt.text=jugadores[4];
82        Tim5_txt.text=TimerArray[4];
83        Mov5_txt.text=MovArray[4];
84    }
85
```

Estas son las condiciones para cuando son 4 o 5 jugadores, hace lo mismo que el 3 y sirve también para que no se creen problemas por valores nulos.

```
87  //BOTON PARA REINICIAR EL JUEGO
88  regresar_btn.addEventListener(MouseEvent.CLICK, principio);
89
90  function principio(e:MouseEvent){
91      gotoAndStop(2);
92      cont=0;
93      contPlayer=0;
94  }
```

Botón para el botón de regresar, lo que hace es inicializar en 0 nuestras variables principales para poder volver a utilizar el juego sin tener que reiniciar nada.

```
96  //Función para guardar los datos en un txt
97  var ff:FileReference= new FileReference();
98  var texto:String;
99  texto="Players  Time  Movements"+"\n";
100 for (var i:int=0; i<numpl; i++){
101     texto=texto+jugadores[i]+" "+TimerArray[i]+" "+MovArray[i]+"\n";
102 }
103 ff.save(texto, "results.txt");
```

Esta función es para guardar nuestros datos en un documento de texto, el cual se llama “results.txt”, se utiliza un ciclo para poder guardar lo que tenemos en nuestros arrays almacenados. Cuando esto finaliza nos abre una ventana para guardar nuestro archivo.

# Conclusión.

En conclusión, trabajar con el lenguaje ActionScript 3.0 en el programa Animate es de gran aprendizaje para nosotros como estudiantes y para aquellos que quieran aprender a programar y diseñar juegos básicos.

Siendo uno de los primeros juegos para al menos uno de nosotros es de gran importancia la creación de este, pues además de empezar a crear este tipo de aplicaciones podemos aprender a diseñar lo que queremos, aparte de que la lógica de cada uno de los que lo realizan aumenta, mejorando sus habilidades.

Este proyecto fue un reto para nosotros, pues debíamos pensarle mucho para generar tanto los aleatorios de las imágenes, el movimiento de estas, el guardado de datos de cada jugador, etc. Todo esto nos llevó varias horas pensando, y tratando de hacer lo que queríamos, tratando de solucionar errores, tanto de sintaxis como de lógica, verificar que en todas las imágenes funcionara cada cosa, pero al final el resultado fue muy bueno, incluso cosas que se nos hacían más difícil llegaron a ser algo más sencillas al paso del proceso de creación del juego.

Fue un proceso a veces duro, a veces tedioso, pero se logró.