

# Universidad Politécnica de Victoria

## HERRAMIENTAS MULTIMEDIA

### DOCUMENTO TÉCNICO PARA LA REALIZACIÓN DE UN MEMORAMA CON ACTIONSCRIPT 3.0

#### Integrantes:

González Batista Rebeca

Saldívar Izaguirre Elvira Marisol

**Profesor:** Mario Humberto Rodríguez Chávez

Cd. Victoria, Tamaulipas a 21 de febrero de 2018

## Contenido

<b>1. Introducción</b>	3
1.1 Definición del tema del juego	3
1.2. Lógica implementada en el juego	3
<b>2. Desarrollo</b>	5
2.1 Portada	5
2.2 Número de jugadores	6
2.3 Nombres de jugadores	8
2.4 Instrucciones	10
2.5 Memorama	11
2.5.1 Variables iniciales	12
2.5.2 Ciclo para acomodar las cartas aleatoriamente	13
2.5.3 Tweens de tapas	14
2.5.4 Timers	15
2.5.5 Funciones de tapas como botones	23
2.6 Resultados	24
<b>3. Conclusiones</b>	28
<b>4. Anexos</b>	29

# 1. Introducción

## 1.1 Definición del tema del juego

Después de estar debatiendo entre nosotras qué temática usaríamos para el memorama, decidimos escoger a estos perritos:



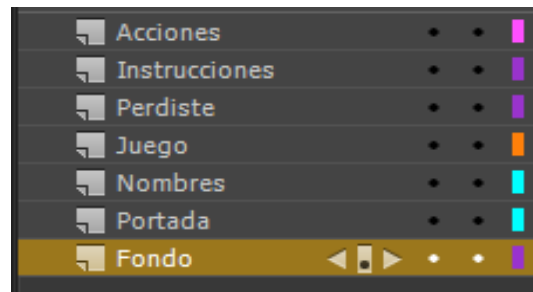
Originalmente, los rostros de los perritos se encontraban en un vector editable en Adobe Illustrator, lo cual aprovechamos para modificarlos a nuestro parecer y agregarles un marco de colores.

Decidimos esta temática porque quisimos hacerlo simple pero tierno.

## 1.2. Lógica implementada en el juego

Para la construcción de nuestro algoritmo, decidimos utilizar una serie de if's y un for. Los primeros para realizar las comprobaciones de las cartas (si eran pares o no, si se encontraban destapadas o no), medir los tiempos entre cada jugador y sus jugadas, reiniciar las jugadas, comprobar qué jugador había ganado, mostrar mensajes de "Perdiste" o "Ganaste" dependiendo de lo que sucediera, entre otras cosas. Los segundos para la asignación de posiciones aleatorias de las cartas.

Se trabajó en Animate CC. Para nuestra comodidad decidimos crear las siguientes capas:



*Acciones:* esta capa se utilizó únicamente para escribir los algoritmos necesarios para cada fotograma, por lo tanto, no se incluyeron elementos gráficos.

*Instrucciones:* aquí se encuentran únicamente las instrucciones del juego.

*Perdiste:* capa para las ventanas de "Ganaste" y "Perdiste".

*Juego:* en esta capa se encuentran todos los elementos gráficos del memorama, tales como las cartas, las tapas, los *textfield* de los puntos, los nombres y los tiempos, así como los elementos gráficos del *frame* de los resultados.

*Nombres:* se encuentran los elementos gráficos de las pantallas de Número de jugadores y Nombres de jugadores.

*Portada:* se encuentran todos los elementos gráficos de nuestra portada.

*Fondo:* aquí se colocaron todos los fondos utilizados dependiendo del *frame* del que se tratase.

## 2. Desarrollo

### 2.1 Portada



## Herramientas Multimedia

### Proyecto Unidad 1



Por: Rebeca Gonzalez Batista  
Elvira Marisol Saldivar Izaguirre

Profesor: M.S.I Mario H. Rodríguez Chávez

Esta portada se realizó de la siguiente manera:

1. Se colocaron los elementos gráficos que se fueran a utilizar, que en este caso son el logo de la universidad y la mancha de acuarela, y después se escribieron en tipografías de nuestra preferencia nuestros nombres, el del profesor, la materia y el tipo de trabajo que se estaba realizando, y para finalizar el botón de "Entrar".
2. En la parte de código, lo único que se hizo fue crear una función para el botón de "Entrar", la cuál sería ir y parar en el *frame* 2, el cuál corresponde a "¿Cuántas personas van a jugar?"

```
1  import flash.events.MouseEvent;
2
3  stop();
4  function memo (event:MouseEvent):void{
5      gotoAndStop(2);
6  }
7  entrarbtn.addEventListener(MouseEvent.CLICK, memo);
8
```

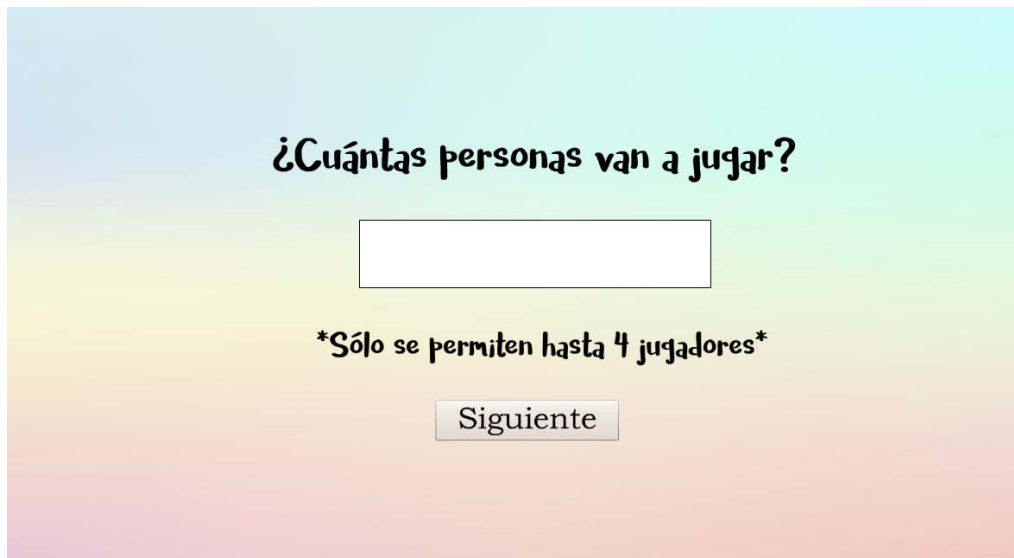
## 2.2 Número de jugadores



**¿Cuántas personas van a jugar?**

Siguiente

En este *frame* se ingresa el número de jugadores que van a participar en el memorama, teniendo un límite de 4 personas. Si se excede este límite, se mostrará un aviso y se dará oportunidad de ingresar la cantidad de personas a jugar de nuevo.



**¿Cuántas personas van a jugar?**

**\*Sólo se permiten hasta 4 jugadores\***

Siguiente

A continuación, se muestra el código correspondiente a este *frame*:

```
1  import flash.events.MouseEvent;
2
3
4  var jugadores: int = 0;
5  aviso.visible=false;
6  numjugtxt.restrict = "0-9";
7
8
9  function siguiente(event: MouseEvent): void {
10     jugadores = Number(numjugtxt.text);
11     if (jugadores > 4) {
12         aviso.visible=true;
13         numjugtxt.text="";
14     } else {
15         gotoAndStop(3);
16     }
17 }
18 sigbtn.addEventListener(MouseEvent.CLICK, siguiente);
```

Se declaró una variable **jugadores**, la visibilidad del aviso de que sólo 4 jugadores son permitidos se asignó **falsa** y se restringió el ingreso de caracteres a la caja de texto, permitiendo únicamente números.

Se creó una función llamada **siguiente** la cual fue asignada al botón **Siguiente**, y esta contiene la asignación de la variable **jugadores** a la caja de texto (en la que el usuario ingresa el número de jugadores), enseguida se utiliza el primer if para verificar que el número ingresado no sea mayor a 4, y si este es el caso, se hará visible el **aviso** donde especifica lo anterior y se limpiará la caja de texto. Si el número de jugadores es menor o igual a 4, pasará al siguiente *frame*.

## 2.3 Nombres de jugadores



A screenshot of a web application interface with a light blue and green gradient background. In the center, the text "Ingrese nombre:" is displayed in a bold, black, sans-serif font. To the right of this text is a white rectangular input field with a thin black border. Below the input field is a grey rectangular button with the word "Guardar" in a black, sans-serif font.

Como se lee en la pantalla, aquí se ingresan los nombres de los jugadores que vayan a participar. Cuando se hace clic al botón "Guardar", la caja de texto se limpia para poder ingresar el nombre del siguiente jugador (en caso de que la partida sea multijugadora).



A screenshot of a web application interface with a light blue and green gradient background. In the center, there is a grey rectangular button with the word "Jugar" in a black, sans-serif font.

Cuando el usuario termina de ingresar los nombres, aparece el botón "Jugar".



A continuación, se muestra el código correspondiente a este *frame*:

```

1  import flash.events.MouseEvent;
2
3  var puntosArr: Array = new Array();
4  var nombresArr: Array = new Array();
5  var tiempoArr:Array = new Array();
6  nombretxt.restrict = "A-Z.a-z";
7
8  playbtn.visible = false;
9  jugartxt.visible = false;
10
11 function guardarjug(event: MouseEvent): void {
12     nombresArr.push(String(nombretxt.text));
13     nombretxt.text = "";
14     if (nombresArr.length == (jugadores)) {
15         savebtn.visible = false;
16         guardar.visible = false;
17         ingrese.visible=false;
18         playbtn.visible = true;
19         jugartxt.visible = true;
20     }
21 }
22 savebtn.addEventListener(MouseEvent.CLICK, guardarjug);
23
24
25
26 function jugar(event: MouseEvent): void {
27     gotoAndStop(4);
28 }
29 playbtn.addEventListener(MouseEvent.CLICK, jugar);

```

*Línea 3-5:* en estas tres líneas se declaran los *Arrays* que se estarán utilizando durante el juego para almacenar los puntos, el nombre y el tiempo de cada jugador.

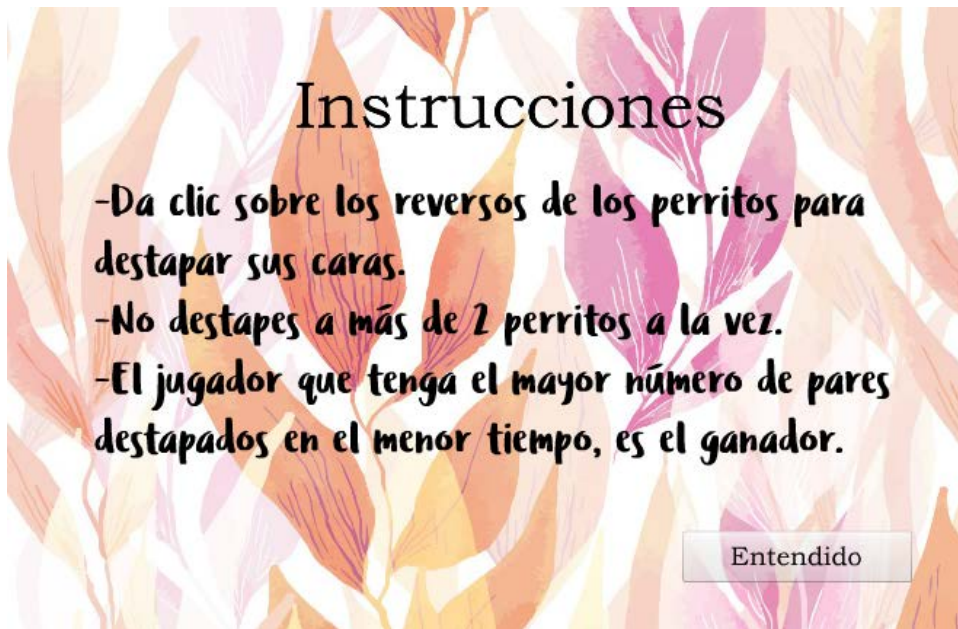
*Línea 6:* se restringe la caja de texto en donde se ingresará el nombre, permitiendo únicamente letras mayúsculas y minúsculas.

*Línea 8-9:* se oculta el botón y el texto de "Jugar".

*Línea 11-22:* función para guardar el nombre ingresado en su respectivo Array cada vez que el usuario haga clic en el botón "Guardar". Si la cantidad de los nombres guardados es equivalente al número de jugadores que se asignó desde un principio, el texto y la caja de texto en la que se ingresan los nombres, así como el botón de Guardar, desaparecen, y aparecen el botón y el texto y el botón de "Jugar".

*Línea 26-29:* función para pasar al siguiente *frame* cuando el usuario haga clic en el botón "Jugar".

## 2.4 Instrucciones

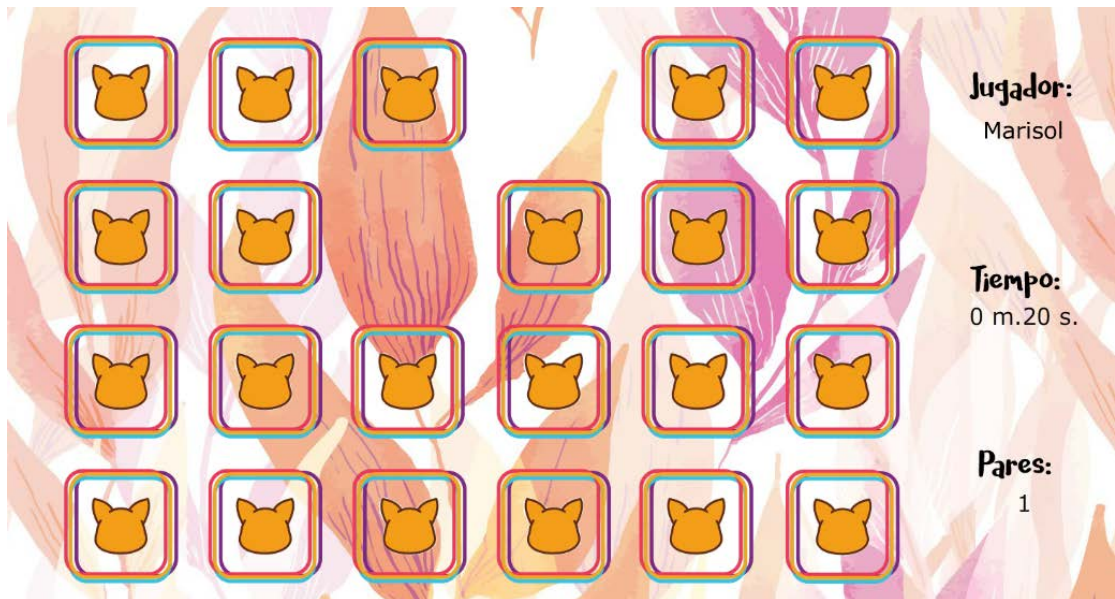


En este *frame* se muestra solamente las instrucciones del juego y un botón para proceder al juego.

```
1  import flash.events.MouseEvent;
2
3  function brb (event:MouseEvent):void{
4      gotoAndStop(5);
5  }
6  okibtn.addEventListener(MouseEvent.CLICK,brb);
```

Línea 3-6: función para pasar al siguiente *frame* (el del juego).

## 2.5 Memorama



La imagen de arriba muestra una jugada ya empezada en donde un par fue encontrado. Las posiciones de las cartas son aleatorias por cada partida. Cada turno tiene una duración de un minuto y medio, y el jugador que encuentre la mayor cantidad de pares gana.

A continuación, se explica el código de este *frame*, el cual es, prácticamente, el juego:

```
1  import fl.transitions.Tween;
2  import fl.transitions.TweenEvent;
3  import fl.transitions.easing.*;
4  import flash.events.Event;
5  import flash.events.MouseEvent;
6  import flash.utils.Timer;
7  import flash.events.TimerEvent;
```

*Línea 1-7:* librerías necesarias para los tweens y los timers que se utilizarán en este *frame*.

### 3.5.1 Variables iniciales

```

9      var j: int = 0;
10     nametxt.text = String(nombresArr[j]);
11     var puntos: int;
12

```

*Línea 9-11:* la variable **j** hace referencia al contador que se utilizará en nuestro ciclo for, en la línea 10 se imprime el nombre del jugador actual en la pantalla, y **puntos** hace referencia al contador de puntos del juego.

*Línea 15-38:* todas las caras de nuestras cartas se encuentran en un estado invisible. **p1, p2, p3,...,p1212** hacen referencia a los nombres de instancia de las caras de nuestras cartas.

```

15     p1.visible = false;
16     p2.visible = false;
17     p3.visible = false;
18     p4.visible = false;
19     p5.visible = false;
20     p6.visible = false;
21     p7.visible = false;
22     p8.visible = false;
23     p9.visible = false;
24     p10.visible = false;
25     p11.visible = false;
26     p12.visible = false;
27     p111.visible = false;
28     p22.visible = false;
29     p33.visible = false;
30     p44.visible = false;
31     p55.visible = false;
32     p66.visible = false;
33     p77.visible = false;
34     p88.visible = false;
35     p99.visible = false;
36     p1010.visible = false;
37     p1111.visible = false;
38     p1212.visible = false;

```

```

42     nplayertxt.visible = false;
43     loserwindow.visible = false;
44     winnerwindow.visible = false;
45     nextptxt.visible = false;
46     ganadortxt.visible = false;

```

*Línea 42-46:* Ventana de perdedor, ganador, y anuncio del siguiente jugador en modo invisible.

### 3.5.2 Ciclo para acomodar las cartas aleatoriamente

```

48
49 var perritosArr: Array = new Array(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p1
50
51 var tapasArr: Array = new Array(r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12, r
52

```

*Línea 49:* variable de tipo Array en la que se almacenarán las caras de las cartas de perritos.

*Línea 51:* variable de tipo Array en la que se almacenarán las tapas de las cartas.

```

73 for (var i: int = 0; i < perritosArr.length; i++) {
74     var aleatorio: int;
75     var equis: Number;
76     var ye: Number;
77
78     aleatorio = Math.random() * perritosArr.length;
79     equis = perritosArr[aleatorio].x;
80     ye = perritosArr[aleatorio].y;
81
82     perritosArr[aleatorio].x = perritosArr[i].x;
83     perritosArr[aleatorio].y = perritosArr[i].y;
84
85     perritosArr[i].x = equis;
86     perritosArr[i].y = ye;
87
88     r1.x = perritosArr[0].x;
89     r2.x = perritosArr[1].x;
90     r3.x = perritosArr[2].x;
91     r4.x = perritosArr[3].x;
92     r5.x = perritosArr[4].x;
93     r6.x = perritosArr[5].x;
94     r7.x = perritosArr[6].x;
95     r8.x = perritosArr[7].x;
96     r9.x = perritosArr[8].x;

```

*Línea 73:* dentro del ciclo declaramos una variable **i**, la cual nos servirá para ir contando las veces que se repiten los acomodados hasta que estos lleguen a un total de 24, que es la longitud de nuestro Array de las caras, o sea, la cantidad de cartas que tendremos.

*Línea 74:* variable **aleatorio** donde se almacenará el número aleatorio para las posiciones.

*Línea 75 y 76:* variables de la posición en los ejes x & y.

*Línea 78:* se le aplica el **math.random** a nuestra variable **aleatorio**.

*Línea 79 y 80:* el contenido de la variable **aleatorio** se les asigna a los ejes x & y de nuestro Array de las caras.



*Línea 82 y 83:* estos números **aleatorios** se asignarán a la posición en la que se encuentre nuestro ciclo for en el Array de las caras de nuestras cartas.

*Línea 85 y 86:* finalmente, vaciamos nuestras posiciones.

*Línea 88-136:* para que los botones de reverso vayan junto a su respectiva carta, les asignamos la misma posición dada anteriormente en los ejes x & y.

```

126         r14.y = perritosArr[13].y;
127         r15.y = perritosArr[14].y;
128         r16.y = perritosArr[15].y;
129         r17.y = perritosArr[16].y;
130         r18.y = perritosArr[17].y;
131         r19.y = perritosArr[18].y;
132         r20.y = perritosArr[19].y;
133         r21.y = perritosArr[20].y;
134         r22.y = perritosArr[21].y;
135         r23.y = perritosArr[22].y;
136         r24.y = perritosArr[23].y;
137
138     }

```

*Línea 138:* termina el ciclo.

### 3.5.3 Tweens de tapas

```

144     var rv1: Tween = new Tween(r1, "x", Regular.easeOut, -400, r1.x, 2, true);
145     var rv2: Tween = new Tween(r2, "x", Regular.easeOut, -400, r2.x, 2, true);
146     var rv3: Tween = new Tween(r3, "x", Regular.easeOut, -400, r3.x, 2, true);
147     var rv4: Tween = new Tween(r4, "x", Regular.easeOut, -400, r4.x, 2, true);
148     var rv5: Tween = new Tween(r5, "x", Regular.easeOut, -400, r5.x, 2, true);
149     var rv6: Tween = new Tween(r6, "x", Regular.easeOut, -400, r6.x, 2, true);
150     var rv7: Tween = new Tween(r7, "x", Regular.easeOut, -400, r7.x, 2, true);
151     var rv8: Tween = new Tween(r8, "x", Regular.easeOut, -400, r8.x, 2, true);
152     var rv9: Tween = new Tween(r9, "x", Regular.easeOut, -400, r9.x, 2, true);
153     var rv10: Tween = new Tween(r10, "x", Regular.easeOut, -400, r10.x, 2, true);
154     var rv11: Tween = new Tween(r11, "x", Regular.easeOut, -400, r11.x, 2, true);
155     var rv12: Tween = new Tween(r12, "x", Regular.easeOut, -400, r12.x, 2, true);
156     var rv13: Tween = new Tween(r13, "x", Regular.easeOut, -400, r13.x, 2, true);
157     var rv14: Tween = new Tween(r14, "x", Regular.easeOut, -400, r14.x, 2, true);
158     var rv15: Tween = new Tween(r15, "x", Regular.easeOut, -400, r15.x, 2, true);
159     var rv16: Tween = new Tween(r16, "x", Regular.easeOut, -400, r16.x, 2, true);
160     var rv17: Tween = new Tween(r17, "x", Regular.easeOut, -400, r17.x, 2, true);
161     var rv18: Tween = new Tween(r18, "x", Regular.easeOut, -400, r18.x, 2, true);
162     var rv19: Tween = new Tween(r19, "x", Regular.easeOut, -400, r19.x, 2, true);
163     var rv20: Tween = new Tween(r20, "x", Regular.easeOut, -400, r20.x, 2, true);
164     var rv21: Tween = new Tween(r21, "x", Regular.easeOut, -400, r21.x, 2, true);
165     var rv22: Tween = new Tween(r22, "x", Regular.easeOut, -400, r22.x, 2, true);
166     var rv23: Tween = new Tween(r23, "x", Regular.easeOut, -400, r23.x, 2, true);
167     var rv24: Tween = new Tween(r24, "x", Regular.easeOut, -400, r24.x, 2, true);
168

```

*Línea 144-167:* las cartas del reverso entran con **Tweens**. Si ponemos estas definiciones antes de nuestro ciclo for, no van a funcionar, ya que queremos que se posicionen en el lugar aleatorio que se les asignó.

### 3.5.4 Timers

```
174     var seg: int = 0;
175     var cont1: int;
176     var timer: Timer = new Timer(1000, cont1++);
177     var min: int = 0;
178
179
180     var sec: int;
181     var cont2: int;
182     var tim: Timer = new Timer(1000, cont2++);
183
184
185     timer.start();
186     timer.addEventListener(TimerEvent.TIMER, tiempo);
187
188
189     tim.addEventListener(TimerEvent.TIMER, time);
190
```

*Línea 174-177:* variables necesarias para el *timer* de nuestro juego. Donde **seg** son los segundos, **cont1** el contador de nuestro *timer*, **timer** es el nombre de nuestro *timer*, y **min** los minutos.

*Línea 180-182:* variables necesarias para el *timer* cuando hay cambio de jugador. Donde **sec** son los segundos, **cont2** el contador de nuestro *timer* y **tim** es el nombre de nuestro *timer*.

*Línea 185-186:* el *timer* inicia entrando al frame, y es llamado por la función de **tiempo**.

*Línea 189:* llamada a la función **time** con el *timer* del tiempo entre cambios de jugador.

```
192 function tiempo(tiempoevent: TimerEvent): void {  
193     seg++;  
194     tiempotxt.text = seg + " s.";  
195     tiempotxt.text = min + " m." + seg + " s.";  
196     if (puntos == 12) {  
197         winnerwindow.visible = true;  
198         ganadortxt.text = (nombresArr[j]);  
199         ganadortxt.visible = true;  
200         timer.stop();  
201         tim.start();  
202         if (j == (nombresArr.length - 1)) {  
203             puntosArr[j] = String(puntos);  
204             tiempoArr[j] = String(tiempotxt.text);  
205             timer.stop();  
206             gotoAndStop(6);  
207         }else{  
208             puntosArr[j] = String(puntos);  
209             tiempoArr[j] = String(tiempotxt.text);  
210             nextptxt.text = (nombresArr[j + 1]);  
211             nextptxt.visible = true;  
212             j++;  
213         }  
    }
```

*Línea 192:* inicia función **tiempo**.

*Línea 193-195:* los segundos van aumentando y se van imprimiendo en el *textfield* llamado **tiempotxt**, al igual que los minutos.

*Línea 196-207:* condición en la que se verifica si los puntos son iguales a 12, de ser cierto, aparece la ventana de ganador junto con el nombre del jugador actual, el tiempo se detiene y, como se terminó el juego para ese jugador, se activa el tiempo para pasar al siguiente jugador. En caso de que el jugador actual hubiese sido el último jugador, los puntos se guardan en su respectivo lugar del Array, así como el tiempo que hizo, y se pasa al siguiente *frame* que contiene los resultados. Si el jugador actual no es el último jugador, se guardan los puntos y el tiempo en sus respectivos Arrays y se imprime el nombre del siguiente jugador, y el contador de jugadores aumenta 1.



```

215     p1.visible = false;
216     p2.visible = false;
217     p3.visible = false;
218     p4.visible = false;
219     p5.visible = false;
220     p6.visible = false;
221     p7.visible = false;
222     p8.visible = false;
223     p9.visible = false;
224     p10.visible = false;
225     p11.visible = false;
226     p12.visible = false;
227     p111.visible = false;
228     p22.visible = false;
229     p33.visible = false;
230     p44.visible = false;
231     p55.visible = false;
232     p66.visible = false;
233     p77.visible = false;
234     p88.visible = false;
235     p99.visible = false;
236     p1010.visible = false;
237     p1111.visible = false;
238     p1212.visible = false;
239
240     r1.visible = true;
241     r2.visible = true;
242     r3.visible = true;
243     r4.visible = true;
244     r5.visible = true;
245     r6.visible = true;
246     r7.visible = true;
247     r8.visible = true;
248     r9.visible = true;
249     r10.visible = true;
250     r11.visible = true;
251     r12.visible = true;
252     r13.visible = true;
253     r14.visible = true;
254     r15.visible = true;
255     r16.visible = true;
256     r17.visible = true;
257     r18.visible = true;
258     r19.visible = true;
259     r20.visible = true;
260     r21.visible = true;
261     r22.visible = true;
262     r23.visible = true;
263     r24.visible = true;
264 }

```

*Línea 215-238:* las caras de las cartas vuelven a ser invisibles, ya que el juego se reiniciará.

*Línea 240-263:* las tapas de las cartas vuelven a ser visibles.

*Línea 264:* termina la condición de la línea 196.

```

266     if (min == 0 && seg == 60) {
267         seg = 0;
268         min++;
269     }

```

*Línea 266-269:* condición que verifica si los segundos llegaron a 60 y los minutos siguen en 0, de ser cierto, los segundos se regresarán a 0 y minutos aumentará 1.

```
271     if (min == 1 && seg == 31) {  
272         if (j == (nombresArr.length - 1)) {  
273             puntosArr[j] = String(puntos);  
274             tiempoArr[j] = String(tiempotxt.text);  
275             timer.stop();  
276             gotoAndStop(6);  
277         } else {  
278             if (puntos < 12) {  
279                 nplayertxt.visible = true;  
280                 loserwindow.visible = true;  
281             }  
282             puntosArr[j] = String(puntos);  
283             tiempoArr[j] = String(tiempotxt.text);  
284             nplayertxt.text = (nombresArr[j + 1]);  
285             timer.stop();  
286             j++;  
287             tim.start();
```

*Línea 271-287:* condición que verifica que se haya llegado al minuto y medio del juego, y de ser así, verifica que el contador de jugadores sea equivalente a la longitud del Array – 1 (el -1 se debe a que el contador y las posiciones del Array comienzan en 0), y de ser así, se guardan los puntos y el tiempo en sus respectivos Arrays, el tiempo se detiene y se pasa al siguiente *frame*. Si aún no es el último jugador, se verifica si los puntos obtenidos son menores a 12 (sea 12 el número de pares totales), y de ser cierto, aparece la ventana de perdedor junto con el nombre del jugador, se guardan los respectivos puntos y tiempo en sus Arrays y se imprime el nombre del siguiente jugador, se detiene el tiempo, se suma 1 al contador e inicia el *timer* para el cambio de jugador.

```
308 p99.visible = false;
309 p1010.visible = false;
310 p1111.visible = false;
311 p1212.visible = false;
312
313 r1.visible = true;
314 r2.visible = true;
315 r3.visible = true;
316 r4.visible = true;
317 r5.visible = true;
318 r6.visible = true;
319 r7.visible = true;
320 r8.visible = true;
321 r9.visible = true;
322 r10.visible = true;
323 r11.visible = true;
324 r12.visible = true;
325 r13.visible = true;
326 r14.visible = true;
327 r15.visible = true;
328 r16.visible = true;
329 r17.visible = true;
330 r18.visible = true;
331 r19.visible = true;
332 r20.visible = true;
333 r21.visible = true;
334 r22.visible = true;
335 r23.visible = true;
336 r24.visible = true;
337
338 }
339 }
340 }
```

*Línea 288-311:* las caras de las cartas vuelven a ser invisibles porque el juego se reiniciará.

*Línea 313-336:* las tapas de las cartas vuelven a ser visibles.

*Línea 338:* fin de la condición que verifica si el contador de jugadores es igual a la longitud del Array de nombres – 1.

*Línea 339:* fin de la condición que verifica si ya se llegó al límite de tiempo de un minuto y medio.

*Línea 340:* fin de la función del **tiempo**.

```
352 function time(tiempoevent: TimerEvent) {  
353     sec++;  
354     if (sec > 5) {  
355         if (j == (nombresArr.length)) {  
356             timer.stop();  
357             gotoAndStop(6);  
358         } else {  
359             nametxt.text = String(nombresArr[j]);  
360             tim.stop();  
361             nplayertxt.visible = false;  
362             loserwindow.visible = false;  
363             winnerwindow.visible = false;  
364             ganadortxt.visible = false;  
365             nextptxt.visible = false;  
366             seg = 0;  
367             min = 0;  
368             sec = 0;  
369             timer.start();  
370             puntos = 0;  
371             parestxt.text = String(puntos);  
372         }  
373     }  
374 }
```

*Línea 352:* inicio de la función **time**, la cual se utiliza para el tiempo de espera entre cada jugador.

*Línea 353:* cuando se llama a esta función, la variable **sec** comienza a incrementarse.

*Línea 354-373:* cuando el tiempo de espera llega a 5 segundos, se verifica que el contador de jugadores sea igual a la longitud del Array de los nombres, y de ser así, se detiene el tiempo y se pasa al siguiente *frame*, ya que esto significará que es el último jugador. Si no se trata del último jugador, entonces se imprime el nombre del jugador que sigue, se detiene este *timery* se ocultan las ventanas de "Ganador" o "Perdedor" dependiendo de cual sea la que esté visible, se reinician los segundos, los minutos y la variable **sec** para que se encuentre en 0 para su próxima llamada. El *timer* principal comienza, los puntos se reinician y se guardan los puntos actuales en su lugar correspondiente en el Array.

*Línea 374:* fin de la función **time**.

```

381  var parl: int;
382  var contpl: int;
383  var timl: Timer = new Timer(1000, contpl++);
384

```

*Línea 381-383:* declaración de las variables para el *timer* que se ocupará para el tiempo que duran las cartas destapadas.

```

386  function tiempocards(tiempoevent: TimerEvent): void {
387      parl++;
388      if ((parl > 3) && (cartas > 1)) {
389          if ((p1.visible == true) && (p111.visible == true)) {
390              p1.visible = false;
391              p111.visible = false;
392              puntos++;
393              parestxt.text = String(puntos);
394              timl.stop();
395              cartas = 0;
396          } else {
397              if ((p1.visible == true) && (p111.visible == false)) {
398                  r1.visible = true;
399                  p1.visible = false;
400              }
401              if ((p1.visible == false) && (p111.visible == true)) {
402                  r13.visible = true;
403                  p111.visible = false;
404              }
405              cartas = 0;
406          }
407          if ((p2.visible == true) && (p22.visible == true)) {
408              p2.visible = false;
409              p22.visible = false;
410              puntos++;
411              parestxt.text = String(puntos);
412              timl.stop();
413              cartas = 0;
414          } else {
415              if ((p2.visible == true) && (p22.visible == false)) {
416                  r2.visible = true;
417                  p2.visible = false;
418              }
419              if ((p2.visible == false) && (p22.visible == true)) {

```

*Línea 386:* inicio de la función **tiempocards**, la cual es utilizada para comparar las cartas y saber si son pares o no.

*Línea 387:* el contador **par1** incrementa.

*Línea 388:* se realiza una verificación de que, si el contador **par1** ha superado los 3 segundos, y ya se dieron clic a 2 cartas (la variable **cartas** será definida más abajo), entonces realiza lo siguiente.

*Línea 389-396:* condición para verificar que ambas cartas que forman un par estén visibles, y de ser así, las desaparece y suma 1 al contador de puntos, imprime los puntos que lleva en la pantalla, detiene este contador de tiempo llamado **tim1** y reinicia la variable **cartas**.

*Línea 396-406:* Si la condición de las líneas 389-396 no se cumplen, se verifica que, si una carta de este par esté visible y la otra no, y de ser cierto, las vuelve a tapar con sus respectivas reversas, y se reinicia la variable **cartas**.

```
628         tim1.stop();
629
630         cartas = 0;
631     } else {
632         if ((p12.visible == true) && (p1212.visible == false)) {
633             r12.visible = true;
634             p12.visible = false;
635
636         }
637         if ((p12.visible == false) && (p1212.visible == true)) {
638             r24.visible = true;
639             p1212.visible = false;
640         }
641
642         cartas = 0;
643     }
644
645 }
646
647 }
648 tim1.addEventListener(TimerEvent.TIMER, tiempocards);
```

*Línea 407-643:* Las mismas condiciones que se verifican desde la línea 389 hasta la línea 406 se repiten, pero para cada par de cartas.

*Línea 645:* fin de la condición que verifica si el tiempo de espera entre cada par de cartas volteadas se ha agotado.

*Línea 647:* fin de la función **tiempocards**.

*Línea 648:* llamada a esta función con el *timer* **tim1**.

### 3.5.5 Funciones de tapas como botones

```

651  var cartas: int = 0;
652
653  function rev1(event: MouseEvent): void {
654      p1.visible = true;
655      r1.visible = false;
656      cartas++;
657      if (cartas > 1) {
658          tim1.start();
659      }
660      trace(puntos);
661  }
662  r1.addEventListener(MouseEvent.CLICK, rev1);
663
664  function rev2(event: MouseEvent): void {
665      p2.visible = true;
666      r2.visible = false;
667      cartas++;
668      if (cartas > 1) {
669          tim1.start();
670      }
671      trace(puntos);
672  }
673  r2.addEventListener(MouseEvent.CLICK, rev2);
674
675  function rev3(event: MouseEvent): void {
676      p3.visible = true;
677      r3.visible = false;
678      cartas++;

```

*Línea 651:* variable que se utiliza como un contador de clics (permitidos son 2).

*Línea 653-662:* función para el botón de la tapa 1. Al hacer clic en la tapa (en este caso **r1**), la cara de la carta que esté debajo se hace visible (en este caso **p1**). El contador de clics aumenta 1, y si este ha llegado a un valor superior a 1, inicia la función **tim1** (detallada anteriormente), y realiza un trace de los puntos que lleva acumulados el jugador.

```

931      r22.visible = false;
932      cartas++;
933      if (cartas > 1) {
934          tim1.start();
935      }
936      trace(puntos);
937  }
938  r22.addEventListener(MouseEvent.CLICK, rev22);
939
940  function rev23(event: MouseEvent): void {
941      p1111.visible = true;
942      r23.visible = false;
943      cartas++;
944      if (cartas > 1) {
945          tim1.start();
946      }
947      trace(puntos);
948  }
949  r23.addEventListener(MouseEvent.CLICK, rev23);
950
951  function rev24(event: MouseEvent): void {
952      p1212.visible = true;
953      r24.visible = false;
954      cartas++;
955      if (cartas > 1) {
956          tim1.start();
957      }
958      trace(puntos);
959  }
960  r24.addEventListener(MouseEvent.CLICK, rev24);
961
962  function rev25(event: MouseEvent): void {
963      p1313.visible = true;
964      r25.visible = false;
965      cartas++;
966      if (cartas > 1) {
967          tim1.start();
968      }
969      trace(puntos);
970  }
971  r25.addEventListener(MouseEvent.CLICK, rev25);
972
973  function rev26(event: MouseEvent): void {
974      p1414.visible = true;
975      r26.visible = false;
976      cartas++;
977      if (cartas > 1) {
978          tim1.start();
979      }
980      trace(puntos);
981  }
982  r26.addEventListener(MouseEvent.CLICK, rev26);
983
984  function rev27(event: MouseEvent): void {
985      p1515.visible = true;
986      r27.visible = false;
987      cartas++;
988      if (cartas > 1) {
989          tim1.start();
990      }
991      trace(puntos);
992  }
993  r27.addEventListener(MouseEvent.CLICK, rev27);
994
995  function rev28(event: MouseEvent): void {
996      p1616.visible = true;
997      r28.visible = false;
998      cartas++;
999      if (cartas > 1) {
1000          tim1.start();
1001      }
1002      trace(puntos);
1003  }
1004  r28.addEventListener(MouseEvent.CLICK, rev28);
1005
1006  function rev29(event: MouseEvent): void {
1007      p1717.visible = true;
1008      r29.visible = false;
1009      cartas++;
1010      if (cartas > 1) {
1011          tim1.start();
1012      }
1013      trace(puntos);
1014  }
1015  r29.addEventListener(MouseEvent.CLICK, rev29);
1016
1017  function rev30(event: MouseEvent): void {
1018      p1818.visible = true;
1019      r30.visible = false;
1020      cartas++;
1021      if (cartas > 1) {
1022          tim1.start();
1023      }
1024      trace(puntos);
1025  }
1026  r30.addEventListener(MouseEvent.CLICK, rev30);
1027
1028  function rev31(event: MouseEvent): void {
1029      p1919.visible = true;
1030      r31.visible = false;
1031      cartas++;
1032      if (cartas > 1) {
1033          tim1.start();
1034      }
1035      trace(puntos);
1036  }
1037  r31.addEventListener(MouseEvent.CLICK, rev31);
1038
1039  function rev32(event: MouseEvent): void {
1040      p2020.visible = true;
1041      r32.visible = false;
1042      cartas++;
1043      if (cartas > 1) {
1044          tim1.start();
1045      }
1046      trace(puntos);
1047  }
1048  r32.addEventListener(MouseEvent.CLICK, rev32);
1049
1050  function rev33(event: MouseEvent): void {
1051      p2121.visible = true;
1052      r33.visible = false;
1053      cartas++;
1054      if (cartas > 1) {
1055          tim1.start();
1056      }
1057      trace(puntos);
1058  }
1059  r33.addEventListener(MouseEvent.CLICK, rev33);
1060
1061  function rev34(event: MouseEvent): void {
1062      p2222.visible = true;
1063      r34.visible = false;
1064      cartas++;
1065      if (cartas > 1) {
1066          tim1.start();
1067      }
1068      trace(puntos);
1069  }
1070  r34.addEventListener(MouseEvent.CLICK, rev34);
1071
1072  function rev35(event: MouseEvent): void {
1073      p2323.visible = true;
1074      r35.visible = false;
1075      cartas++;
1076      if (cartas > 1) {
1077          tim1.start();
1078      }
1079      trace(puntos);
1080  }
1081  r35.addEventListener(MouseEvent.CLICK, rev35);
1082
1083  function rev36(event: MouseEvent): void {
1084      p2424.visible = true;
1085      r36.visible = false;
1086      cartas++;
1087      if (cartas > 1) {
1088          tim1.start();
1089      }
1090      trace(puntos);
1091  }
1092  r36.addEventListener(MouseEvent.CLICK, rev36);
1093
1094  function rev37(event: MouseEvent): void {
1095      p2525.visible = true;
1096      r37.visible = false;
1097      cartas++;
1098      if (cartas > 1) {
1099          tim1.start();
1100      }
1101      trace(puntos);
1102  }
1103  r37.addEventListener(MouseEvent.CLICK, rev37);
1104
1105  function rev38(event: MouseEvent): void {
1106      p2626.visible = true;
1107      r38.visible = false;
1108      cartas++;
1109      if (cartas > 1) {
1110          tim1.start();
1111      }
1112      trace(puntos);
1113  }
1114  r38.addEventListener(MouseEvent.CLICK, rev38);
1115
1116  function rev39(event: MouseEvent): void {
1117      p2727.visible = true;
1118      r39.visible = false;
1119      cartas++;
1120      if (cartas > 1) {
1121          tim1.start();
1122      }
1123      trace(puntos);
1124  }
1125  r39.addEventListener(MouseEvent.CLICK, rev39);
1126
1127  function rev40(event: MouseEvent): void {
1128      p2828.visible = true;
1129      r40.visible = false;
1130      cartas++;
1131      if (cartas > 1) {
1132          tim1.start();
1133      }
1134      trace(puntos);
1135  }
1136  r40.addEventListener(MouseEvent.CLICK, rev40);
1137
1138  function rev41(event: MouseEvent): void {
1139      p2929.visible = true;
1140      r41.visible = false;
1141      cartas++;
1142      if (cartas > 1) {
1143          tim1.start();
1144      }
1145      trace(puntos);
1146  }
1147  r41.addEventListener(MouseEvent.CLICK, rev41);
1148
1149  function rev42(event: MouseEvent): void {
1150      p3030.visible = true;
1151      r42.visible = false;
1152      cartas++;
1153      if (cartas > 1) {
1154          tim1.start();
1155      }
1156      trace(puntos);
1157  }
1158  r42.addEventListener(MouseEvent.CLICK, rev42);
1159
1160  function rev43(event: MouseEvent): void {
1161      p3131.visible = true;
1162      r43.visible = false;
1163      cartas++;
1164      if (cartas > 1) {
1165          tim1.start();
1166      }
1167      trace(puntos);
1168  }
1169  r43.addEventListener(MouseEvent.CLICK, rev43);
1170
1171  function rev44(event: MouseEvent): void {
1172      p3232.visible = true;
1173      r44.visible = false;
1174      cartas++;
1175      if (cartas > 1) {
1176          tim1.start();
1177      }
1178      trace(puntos);
1179  }
1180  r44.addEventListener(MouseEvent.CLICK, rev44);
1181
1182  function rev45(event: MouseEvent): void {
1183      p3333.visible = true;
1184      r45.visible = false;
1185      cartas++;
1186      if (cartas > 1) {
1187          tim1.start();
1188      }
1189      trace(puntos);
1190  }
1191  r45.addEventListener(MouseEvent.CLICK, rev45);
1192
1193  function rev46(event: MouseEvent): void {
1194      p3434.visible = true;
1195      r46.visible = false;
1196      cartas++;
1197      if (cartas > 1) {
1198          tim1.start();
1199      }
1200      trace(puntos);
1201  }
1202  r46.addEventListener(MouseEvent.CLICK, rev46);
1203
1204  function rev47(event: MouseEvent): void {
1205      p3535.visible = true;
1206      r47.visible = false;
1207      cartas++;
1208      if (cartas > 1) {
1209          tim1.start();
1210      }
1211      trace(puntos);
1212  }
1213  r47.addEventListener(MouseEvent.CLICK, rev47);
1214
1215  function rev48(event: MouseEvent): void {
1216      p3636.visible = true;
1217      r48.visible = false;
1218      cartas++;
1219      if (cartas > 1) {
1220          tim1.start();
1221      }
1222      trace(puntos);
1223  }
1224  r48.addEventListener(MouseEvent.CLICK, rev48);
1225
1226  function rev49(event: MouseEvent): void {
1227      p3737.visible = true;
1228      r49.visible = false;
1229      cartas++;
1230      if (cartas > 1) {
1231          tim1.start();
1232      }
1233      trace(puntos);
1234  }
1235  r49.addEventListener(MouseEvent.CLICK, rev49);
1236
1237  function rev50(event: MouseEvent): void {
1238      p3838.visible = true;
1239      r50.visible = false;
1240      cartas++;
1241      if (cartas > 1) {
1242          tim1.start();
1243      }
1244      trace(puntos);
1245  }
1246  r50.addEventListener(MouseEvent.CLICK, rev50);
1247
1248  function rev51(event: MouseEvent): void {
1249      p3939.visible = true;
1250      r51.visible = false;
1251      cartas++;
1252      if (cartas > 1) {
1253          tim1.start();
1254      }
1255      trace(puntos);
1256  }
1257  r51.addEventListener(MouseEvent.CLICK, rev51);
1258
1259  function rev52(event: MouseEvent): void {
1260      p4040.visible = true;
1261      r52.visible = false;
1262      cartas++;
1263      if (cartas > 1) {
1264          tim1.start();
1265      }
1266      trace(puntos);
1267  }
1268  r52.addEventListener(MouseEvent.CLICK, rev52);
1269
1270  function rev53(event: MouseEvent): void {
1271      p4141.visible = true;
1272      r53.visible = false;
1273      cartas++;
1274      if (cartas > 1) {
1275          tim1.start();
1276      }
1277      trace(puntos);
1278  }
1279  r53.addEventListener(MouseEvent.CLICK, rev53);
1280
1281  function rev54(event: MouseEvent): void {
1282      p4242.visible = true;
1283      r54.visible = false;
1284      cartas++;
1285      if (cartas > 1) {
1286          tim1.start();
1287      }
1288      trace(puntos);
1289  }
1290  r54.addEventListener(MouseEvent.CLICK, rev54);
1291
1292  function rev55(event: MouseEvent): void {
1293      p4343.visible = true;
1294      r55.visible = false;
1295      cartas++;
1296      if (cartas > 1) {
1297          tim1.start();
1298      }
1299      trace(puntos);
1300  }
1301  r55.addEventListener(MouseEvent.CLICK, rev55);
1302
1293  function rev56(event: MouseEvent): void {
1304      p4444.visible = true;
1305      r56.visible = false;
1306      cartas++;
1307      if (cartas > 1) {
1308          tim1.start();
1309      }
1310      trace(puntos);
1311  }
1312  r56.addEventListener(MouseEvent.CLICK, rev56);
1313
1314  function rev57(event: MouseEvent): void {
1315      p4545.visible = true;
1316      r57.visible = false;
1317      cartas++;
1318      if (cartas > 1) {
1319          tim1.start();
1320      }
1321      trace(puntos);
1322  }
1323  r57.addEventListener(MouseEvent.CLICK, rev57);
1324
1325  function rev58(event: MouseEvent): void {
1326      p4646.visible = true;
1327      r58.visible = false;
1328      cartas++;
1329      if (cartas > 1) {
1330          tim1.start();
1331      }
1332      trace(puntos);
1333  }
1334  r58.addEventListener(MouseEvent.CLICK, rev58);
1335
1336  function rev59(event: MouseEvent): void {
1337      p4747.visible = true;
1338      r59.visible = false;
1339      cartas++;
1340      if (cartas > 1) {
1341          tim1.start();
1342      }
1343      trace(puntos);
1344  }
1345  r59.addEventListener(MouseEvent.CLICK, rev59);
1346
1347  function rev60(event: MouseEvent): void {
1348      p4848.visible = true;
1349      r60.visible = false;
1350      cartas++;
1351      if (cartas > 1) {
1352          tim1.start();
1353      }
1354      trace(puntos);
1355  }
1356  r60.addEventListener(MouseEvent.CLICK, rev60);
1357
1358  function rev61(event: MouseEvent): void {
1359      p4949.visible = true;
1360      r61.visible = false;
1361      cartas++;
1362      if (cartas > 1) {
1363          tim1.start();
1364      }
1365      trace(puntos);
1366  }
1367  r61.addEventListener(MouseEvent.CLICK, rev61);
1368
1369  function rev62(event: MouseEvent): void {
1370      p5050.visible = true;
1371      r62.visible = false;
1372      cartas++;
1373      if (cartas > 1) {
1374          tim1.start();
1375      }
1376      trace(puntos);
1377  }
1378  r62.addEventListener(MouseEvent.CLICK, rev62);
1379
1380  function rev63(event: MouseEvent): void {
1381      p5151.visible = true;
1382      r63.visible = false;
1383      cartas++;
1384      if (cartas > 1) {
1385          tim1.start();
1386      }
1387      trace(puntos);
1388  }
1389  r63.addEventListener(MouseEvent.CLICK, rev63);
1390
1391  function rev64(event: MouseEvent): void {
1392      p5252.visible = true;
1393      r64.visible = false;
1394      cartas++;
1395      if (cartas > 1) {
1396          tim1.start();
1397      }
1398      trace(puntos);
1399  }
1400  r64.addEventListener(MouseEvent.CLICK, rev64);
1401
1402  function rev65(event: MouseEvent): void {
1403      p5353.visible = true;
1404      r65.visible = false;
1405      cartas++;
1406      if (cartas > 1) {
1407          tim1.start();
1408      }
1409      trace(puntos);
1410  }
1411  r65.addEventListener(MouseEvent.CLICK, rev65);
1412
1413  function rev66(event: MouseEvent): void {
1414      p5454.visible = true;
1415      r66.visible = false;
1416      cartas++;
1417      if (cartas > 1) {
1418          tim1.start();
1419      }
1420      trace(puntos);
1421  }
1422  r66.addEventListener(MouseEvent.CLICK, rev66);
1423
1424  function rev67(event: MouseEvent): void {
1425      p5555.visible = true;
1426      r67.visible = false;
1427      cartas++;
1428      if (cartas > 1) {
1429          tim1.start();
1430      }
1431      trace(puntos);
1432  }
1433  r67.addEventListener(MouseEvent.CLICK, rev67);
1434
1435  function rev68(event: MouseEvent): void {
1436      p5656.visible = true;
1437      r68.visible = false;
1438      cartas++;
1439      if (cartas > 1) {
1440          tim1.start();
1441      }
1442      trace(puntos);
1443  }
1444  r68.addEventListener(MouseEvent.CLICK, rev68);
1445
1446  function rev69(event: MouseEvent): void {
1447      p5757.visible = true;
1448      r69.visible = false;
1449      cartas++;
1450      if (cartas > 1) {
1451          tim1.start();
1452      }
1453      trace(puntos);
1454  }
1455  r69.addEventListener(MouseEvent.CLICK, rev69);
1456
1457  function rev70(event: MouseEvent): void {
1458      p5858.visible = true;
1459      r70.visible = false;
1460      cartas++;
1461      if (cartas > 1) {
1462          tim1.start();
1463      }
1464      trace(puntos);
1465  }
1466  r70.addEventListener(MouseEvent.CLICK, rev70);
1467
1468  function rev71(event: MouseEvent): void {
1469      p5959.visible = true;
1470      r71.visible = false;
1471      cartas++;
1472      if (cartas > 1) {
1473          tim1.start();
1474      }
1475      trace(puntos);
1476  }
1477  r71.addEventListener(MouseEvent.CLICK, rev71);
1478
1479  function rev72(event: MouseEvent): void {
1480      p6060.visible = true;
1481      r72.visible = false;
1482      cartas++;
1483      if (cartas > 1) {
1484          tim1.start();
1485      }
1486      trace(puntos);
1487  }
1488  r72.addEventListener(MouseEvent.CLICK, rev72);
1489
1490  function rev73(event: MouseEvent): void {
1491      p6161.visible = true;
1492      r73.visible = false;
1493      cartas++;
1494      if (cartas > 1) {
1495          tim1.start();
1496      }
1497      trace(puntos);
1498  }
1499  r73.addEventListener(MouseEvent.CLICK, rev73);
1500
1501  function rev74(event: MouseEvent): void {
1502      p6262.visible = true;
1503      r74.visible = false;
1504      cartas++;
1505      if (cartas > 1) {
1506          tim1.start();
1507      }
1508      trace(puntos);
1509  }
1510  r74.addEventListener(MouseEvent.CLICK, rev74);
1511
1512  function rev75(event: MouseEvent): void {
1513      p6363.visible = true;
1514      r75.visible = false;
1515      cartas++;
1516      if (cartas > 1) {
1517          tim1.start();
1518      }
1519      trace(puntos);
1520  }
1521  r75.addEventListener(MouseEvent.CLICK, rev75);
1522
1523  function rev76(event: MouseEvent): void {
1524      p6464.visible = true;
1525      r76.visible = false;
1526      cartas++;
1527      if (cartas > 1) {
1528          tim1.start();
1529      }
1530      trace(puntos);
1531  }
1532  r76.addEventListener(MouseEvent.CLICK, rev76);
1533
1534  function rev77(event: MouseEvent): void {
1535      p6565.visible = true;
1536      r77.visible = false;
1537      cartas++;
1538      if (cartas > 1) {
1539          tim1.start();
1540      }
1541      trace(puntos);
1542  }
1543  r77.addEventListener(MouseEvent.CLICK, rev77);
1544
1545  function rev78(event: MouseEvent): void {
1546      p6666.visible = true;
1547      r78.visible = false;
1548      cartas++;
1549      if (cartas > 1) {
1550          tim1.start();
1551      }
1552      trace(puntos);
1553  }
1554  r78.addEventListener(MouseEvent.CLICK, rev78);
1555
1556  function rev79(event: MouseEvent): void {
1557      p6767.visible = true;
1558      r79.visible = false;
1559      cartas++;
1560      if (cartas > 1) {
1561          tim1.start();
1562      }
1563      trace(puntos);
1564  }
1565  r79.addEventListener(MouseEvent.CLICK, rev79);
1566
1567  function rev80(event: MouseEvent): void {
1568      p6868.visible = true;
1569      r80.visible = false;
1570      cartas++;
1571      if (cartas > 1) {
1572          tim1.start();
1573      }
1574      trace(puntos);
1575  }
1576  r80.addEventListener(MouseEvent.CLICK, rev80);
1577
1578  function rev81(event: MouseEvent): void {
1579      p6969.visible = true;
1580      r81.visible = false;
1581      cartas++;
1582      if (cartas > 1) {
1583          tim1.start();
1584      }
1585      trace(puntos);
1586  }
1587  r81.addEventListener(MouseEvent.CLICK, rev81);
1588
1589  function rev82(event: MouseEvent): void {
1589      p7070.visible = true;
1590      r82.visible = false;
1591      cartas++;
1592      if (cartas > 1) {
1593          tim1.start();
1594      }
1595      trace(puntos);
1596  }
1597  r82.addEventListener(MouseEvent.CLICK, rev82);
1598
1599  function rev83(event: MouseEvent): void {
1599      p7171.visible = true;
1600      r83.visible = false;
1601      cartas++;
1602      if (cartas > 1) {
1603          tim1.start();
1604      }
1605      trace(puntos);
1606  }
1607  r83.addEventListener(MouseEvent.CLICK, rev83);
1608
1609  function rev84(event: MouseEvent): void {
1609      p7272.visible = true;
1610      r84.visible = false;
1611      cartas++;
1612      if (cartas > 1) {
1613          tim1.start();
1614      }
1615      trace(puntos);
1616  }
1617  r84.addEventListener(MouseEvent.CLICK, rev84);
1618
1619  function rev85(event: MouseEvent): void {
1619      p7373.visible = true;
1620      r85.visible = false;
1621      cartas++;
1622      if (cartas > 1) {
1623          tim1.start();
1624      }
1625      trace(puntos);
1626  }
1627  r85.addEventListener(MouseEvent.CLICK, rev85);
1628
1629  function rev86(event: MouseEvent): void {
1629      p7474.visible = true;
1630      r86.visible = false;
1631      cartas++;
1632      if (cartas > 1) {
1633          tim1.start();
1634      }
1635      trace(puntos);
1636  }
1637  r86.addEventListener(MouseEvent.CLICK, rev86);
1638
1639  function rev87(event: MouseEvent): void {
1639      p7575.visible = true;
1640      r87.visible = false;
1641      cartas++;
1642      if (cartas > 1) {
1643          tim1.start();
1644      }
1645      trace(puntos);
1646  }
1647  r87.addEventListener(MouseEvent.CLICK, rev87);
1648
1649  function rev88(event: MouseEvent): void {
1649      p7676.visible = true;
1650      r88.visible = false;
1651      cartas++;
1652      if (cartas > 1) {
1653          tim1.start();
1654      }
1655      trace(puntos);
1656  }
1657  r88.addEventListener(MouseEvent.CLICK, rev88);
1658
1659  function rev89(event: MouseEvent): void {
1659      p7777.visible = true;
1660      r89.visible = false;
1661      cartas++;
1662      if (cartas > 1) {
1663          tim1.start();
1664      }
1665      trace(puntos);
1666  }
1667  r89.addEventListener(MouseEvent.CLICK, rev89);
1668
1669  function rev90(event: MouseEvent): void {
1669      p7878.visible = true;
1670      r90.visible = false;
1671      cartas++;
1672      if (cartas > 1) {
1673          tim1.start();
1674      }
1675      trace(puntos);
1676  }
1677  r90.addEventListener(MouseEvent.CLICK, rev90);
1678
1679  function rev91(event: MouseEvent): void {
1679      p7979.visible = true;
1680      r91.visible = false;
1681      cartas++;
1682      if (cartas > 1) {
1683          tim1.start();
1684      }
1685      trace(puntos);
1686  }
1687  r91.addEventListener(MouseEvent.CLICK, rev91);
1688
1689  function rev92(event: MouseEvent): void {
1689      p8080.visible = true;
1690      r92.visible = false;
1691      cartas++;
1692      if (cartas > 1) {
1693          tim1.start();
1694      }
1695      trace(puntos);
1696  }
1697  r92.addEventListener(MouseEvent.CLICK, rev92);
1698
1699  function rev93(event: MouseEvent): void {
1699      p8181.visible = true;
1700      r93.visible = false;
1701      cartas++;
1702      if (cartas > 1) {
1703          tim1.start();
1704      }
1705      trace(puntos);
1706  }
1707  r93.addEventListener(MouseEvent.CLICK, rev93);
1708
1709  function rev94(event: MouseEvent): void {
1709      p8282.visible = true;
1710      r94.visible = false;
1711      cartas++;
1712      if (cartas > 1) {
1713          tim1.start();
1714      }
1715      trace(puntos);
1716  }
1717  r94.addEventListener(MouseEvent.CLICK, rev94);
1718
1719  function rev95(event: MouseEvent): void {
1719      p8383.visible = true;
1720      r95.visible = false;
1721      cartas++;
1722      if (cartas > 1) {
1723          tim1.start();
1724      }
1725      trace(puntos);
1726  }
1727  r95.addEventListener(MouseEvent.CLICK, rev95);
1728
1729  function rev96(event: MouseEvent): void {
1729      p8484.visible = true;
1730      r96.visible = false;
1731      cartas++;
1732      if (cartas > 1) {
1733          tim1.start();
1734      }
1735      trace(puntos);
1736  }
1737  r96.addEventListener(MouseEvent.CLICK, rev96);
1738
1739  function rev97(event: MouseEvent): void {
1739      p8585.visible = true;
1740      r97.visible = false;
1741      cartas++;
1742      if (cartas > 1) {
1743          tim1.start();
1744      }
1745      trace(puntos);
1746  }
1747  r97.addEventListener(MouseEvent.CLICK, rev97);
1748
1749  function rev98(event: MouseEvent): void {
1749      p8686.visible = true;
1750      r98.visible = false;
1751      cartas++;
1752      if (cartas > 1) {
1753          tim1.start();
1754      }
1755      trace(puntos);
1756  }
1757  r98.addEventListener(MouseEvent.CLICK, rev98);
1758
1759  function rev99(event: MouseEvent): void {
1759      p8787.visible = true;
1760      r99.visible = false;
1761      cartas++;
1762      if (cartas > 1) {
1763          tim1.start();
1764      }
1765      trace(puntos);
1766  }
1767  r99.addEventListener(MouseEvent.CLICK, rev99);
1768
1769  function rev100(event: MouseEvent): void {
1769      p8888.visible = true;
1770      r100.visible = false;
1771      cartas++;
1772      if (cartas > 1) {
1773          tim1.start();
1774      }
1775      trace(puntos);
1776  }
1777  r100.addEventListener(MouseEvent.CLICK, rev100);
1778
1779  function rev101(event: MouseEvent): void {
1779      p8989.visible = true;
1780      r101.visible = false;
1781      cartas++;
1782      if (cartas > 1) {
1783          tim1.start();
1784      }
1785      trace(puntos);
1786  }
1787  r101.addEventListener(MouseEvent.CLICK, rev101);
1788
1789  function rev102(event: MouseEvent): void {
1789      p9090.visible = true;
1790      r102.visible = false;
1791      cartas++;
1792      if (cartas > 1) {
1793          tim1.start();
1794      }
1795      trace(puntos);
1796  }
1797  r102.addEventListener(MouseEvent.CLICK, rev102);
1798
1799  function rev103(event: MouseEvent): void {
1799      p9191.visible = true;
1800      r103.visible = false;
1801      cartas++;
1802      if (cartas > 1) {
1803          tim1.start();
1804      }
1805      trace(puntos);
1806  }
1807  r103.addEventListener(MouseEvent.CLICK, rev103);
1808
1809  function rev104(event: MouseEvent): void {
1809      p9292.visible = true;
1810      r104.visible = false;
1811      cartas++;
1812      if (cartas > 1) {
1813          tim1.start();
1814      }
1815      trace(puntos);
1816  }
1817  r104.addEventListener(MouseEvent.CLICK, rev104);
1818
1819  function rev105(event: MouseEvent): void {
1819      p9393.visible = true;
1820      r105.visible = false;
1821      cartas++;
1822      if (cartas > 1) {
1823          tim1.start();
1824      }
1825      trace(puntos);
1826  }
1827  r105.addEventListener(MouseEvent.CLICK, rev105);
1828
1829  function rev106(event: MouseEvent): void {
1829      p9494.visible = true;
1830      r106.visible = false;
1831      cartas++;
1832      if (cartas > 1) {
1833          tim1.start();
1834      }
1835      trace(puntos);
1836  }
1837  r106.addEventListener(MouseEvent.CLICK, rev106);
1838
1839  function rev107(event: MouseEvent): void {
1839      p9595.visible = true;
1840      r107.visible = false;
1841      cartas++;
1842      if (cartas > 1) {
1843          tim1.start();
1844      }
1845      trace(puntos);
1846  }
1847  r107.addEventListener(MouseEvent.CLICK, rev107);
1848
1849  function rev108(event: MouseEvent): void {
1849      p9696.visible = true;
1850      r108.visible = false;
1851      cartas++;
1852      if (cartas > 1) {
1853          tim1.start();
1854      }
1855      trace(puntos);
1856  }
1857  r108.addEventListener(MouseEvent.CLICK, rev108);
1858
1859  function rev109(event: MouseEvent): void {
1859      p9797.visible = true;
1860      r109.visible = false;
1861      cartas++;
1862      if (cartas > 1) {
1863          tim1.start();
1864      }
1865      trace(puntos);
1866  }
1867  r109.addEventListener(MouseEvent.CLICK, rev109);
1868
1869  function rev110(event: MouseEvent): void {
1869      p9898.visible = true;
1870      r110.visible = false;
1871      cartas++;
1872      if (cartas > 1) {
1873          tim1.start();
1874      }
1875      trace(puntos);
1876  }
1877  r110.addEventListener(MouseEvent.CLICK, rev110);
1878
1879  function rev111(event: MouseEvent): void {
1879      p9999.visible = true;
1880      r111.visible = false;
1881      cartas++;
1882      if (cartas > 1) {
1883          tim1.start();
1884      }
1885      trace(puntos);
1886  }
1887  r111.addEventListener(MouseEvent.CLICK, rev111);
1888
1889  function rev112(event: MouseEvent): void {
1889      p10000.visible = true;
1890      r112.visible = false;
1891      cartas++;
1892      if (cartas > 1) {
1893          tim1.start();
1894      }
1895      trace(puntos);
1896  }
1897  r112.addEventListener(MouseEvent.CLICK, rev112);
1898
1899  function rev113(event: MouseEvent): void {
1899      p10101.visible = true;
1900      r113.visible = false;
1901      cartas++;
1902      if (cartas > 1) {
1903          tim1.start();
1904      }
1905      trace(puntos);
1906  }
1907  r113.addEventListener(MouseEvent.CLICK, rev113);
1908
1909  function rev114(event: MouseEvent): void {
1909      p10202.visible = true;
1910      r114.visible = false;
1911      cartas++;
1912      if (cartas > 1) {
1913          tim1.start();
1914      }
1915      trace(puntos);
1916  }
1917  r114.addEventListener(MouseEvent.CLICK, rev114);
1918
```

## 2.6 Resultados



Jugador	Pares encontrados	Tiempo
Sebas	12	0 m.43 s.
Marisol	12	0 m.50 s.

Felicidades a Marisol por haber encontrado la mayor cantidad de pares

En este *frame*, se imprimen los resultados de la partida recién finalizada. A resultados nos referimos a nombres, puntos y tiempos de cada jugador.

Se cuenta con un mensaje de felicitaciones para el jugador que más puntos haya obtenido, y un botón de exportar para exportar los resultados a un *.txt* si uno así lo desea.

A continuación, el código de este frame:

```
1  import flash.net.FileReference;
2  import flash.events.MouseEvent;
3
4  var txt: FileReference = new FileReference();
5  var unjuga: Array = new Array();
6  var dosjuga: Array = new Array();
7  var tresjuga: Array = new Array();
8  var cuatrojuga: Array = new Array();
9
```

*Línea 4-8:* **txt** es nuestra variable utilizada para exportar a un documento *.txt*. Los Arrays **unjuga**, **dosjuga**, **tresjuga**, **cuatrojuga** hacen referencia a la cantidad de jugadores permitidos, dependiendo de los jugadores que participen será el Array que se utilice, y son para almacenar los datos de la partida.



```
10 if (nombresArr.length == 1) {
11     player1.text = String(nombresArr[0]);
12     score1.text = String(puntosArr[0]);
13     tiempo1.text = String(tiempoArr[0]);
14     unjuga.push(nombresArr[0]);
15     unjuga.push(puntosArr[0]);
16     unjuga.push(tiempoArr[0]);
17     exportarbtn.addEventListener(MouseEvent.CLICK, exportar1);
18 }
19
20 if (nombresArr.length == 2) {
21     player1.text = String(nombresArr[0]);
22     player2.text = String(nombresArr[1]);
23     score1.text = String(puntosArr[0]);
24     score2.text = String(puntosArr[1]);
25     tiempo1.text = String(tiempoArr[0]);
26     tiempo2.text = String(tiempoArr[1]);
27     if (puntosArr[0] > puntosArr[1]) {
28         congrats.text = "Felicidades a " + nombresArr[0] + " por haber encontrado la
29     } else {
30         congrats.text = "Felicidades a " + nombresArr[1] + " por haber encontrado la
31     }
32     dosjuga.push(nombresArr[0]);
33     dosjuga.push(puntosArr[0]);
34     dosjuga.push(tiempoArr[0]);
35     dosjuga.push(nombresArr[1]);
36     dosjuga.push(puntosArr[1]);
37     dosjuga.push(tiempoArr[1]);
38     exportarbtn.addEventListener(MouseEvent.CLICK, exportar2);
39 }
```

*Línea 10-18:* condición que se cumple solamente si la longitud del Array es igual 1, o sea, si la partida fue de un solo jugador. Se imprime su nombre, su puntaje y el tiempo en el que tardó en terminar el juego. Después se añaden los datos anteriores al Array **unjuga** y se llama a la función **exportar1**.

*Línea 20-39:* se realiza exactamente lo mismo que la condición pasada, solamente que esta se cumple si la cantidad de jugadores es 2, y el Array utilizado para almacenar los datos de los jugadores es **dosjuga**. En este caso, se muestra un mensaje de quién fue el mejor jugador, comparando los puntajes. Se llama a la función **exportar2**.

```

41 if (nombresArr.length == 3) {
42     player1.text = String(nombresArr[0]);
43     player2.text = String(nombresArr[1]);
44     player3.text = String(nombresArr[2]);
45     score1.text = String(puntosArr[0]);
46     score2.text = String(puntosArr[1]);
47     score3.text = String(puntosArr[2]);
48     tiempo1.text = String(tiempoArr[0]);
49     tiempo2.text = String(tiempoArr[1]);
50     tiempo3.text = String(tiempoArr[2]);
51     if ((puntosArr[0] > puntosArr[1]) && (puntosArr[0] > puntosArr[2])) {
52         congrats.text = "Felicidades a " + nombresArr[0] + " por haber encontrado la
53     }
54     if ((puntosArr[1] > puntosArr[0]) && (puntosArr[1] > puntosArr[2])) {
55         congrats.text = "Felicidades a " + nombresArr[1] + " por haber encontrado la
56     }
57     if ((puntosArr[2] > puntosArr[0]) && (puntosArr[2] > puntosArr[1])) {
58         congrats.text = "Felicidades a " + nombresArr[2] + " por haber encontrado la
59     }
60     tresjuga.push(nombresArr[0]);
61     tresjuga.push(puntosArr[0]);
62     tresjuga.push(tiempoArr[0]);
63     tresjuga.push(nombresArr[1]);
64     tresjuga.push(puntosArr[1]);
65     tresjuga.push(tiempoArr[1]);
66     tresjuga.push(nombresArr[2]);
67     tresjuga.push(puntosArr[2]);
68     tresjuga.push(tiempoArr[2]);
69     exportarbtn.addEventListener(MouseEvent.CLICK, exportar3);
70 }

```

*Línea 41-70:* se realiza exactamente lo mismo que la condición pasada, solamente que esta se cumple si la cantidad de jugadores es 3, y el Array utilizado para almacenar los datos de los jugadores es **tresjuga**. En este caso, se muestra un mensaje de quién fue el mejor jugador, comparando los puntajes. Se llama a la función **exportar3**.

```

72 if (nombresArr.length == 4) {
73     player1.text = String(nombresArr[0]);
74     player2.text = String(nombresArr[1]);
75     player3.text = String(nombresArr[2]);
76     player4.text = String(nombresArr[3]);
77     score1.text = String(puntosArr[0]);
78     score2.text = String(puntosArr[1]);
79     score3.text = String(puntosArr[2]);
80     score4.text = String(puntosArr[3]);
81     tiempo1.text = String(tiempoArr[0]);
82     tiempo2.text = String(tiempoArr[1]);
83     tiempo3.text = String(tiempoArr[2]);
84     tiempo4.text = String(tiempoArr[3]);
85     if ((puntosArr[0] > puntosArr[1]) && (puntosArr[0] > puntosArr[2]) && (puntosArr[0] > puntosArr[3])) {
86         congrats.text = "Felicitades a " + nombresArr[0] + " por haber encontrado la
87     }
88     if ((puntosArr[1] > puntosArr[0]) && (puntosArr[1] > puntosArr[2]) && (puntosArr[1] > puntosArr[3])) {
89         congrats.text = "Felicitades a " + nombresArr[1] + " por haber encontrado la
90     }
91     if ((puntosArr[2] > puntosArr[0]) && (puntosArr[2] > puntosArr[1]) && (puntosArr[2] > puntosArr[3])) {
92         congrats.text = "Felicitades a " + nombresArr[2] + " por haber encontrado la
93     }
94     if ((puntosArr[3] > puntosArr[0]) && (puntosArr[3] > puntosArr[1]) && (puntosArr[3] > puntosArr[2])) {
95         congrats.text = "Felicitades a " + nombresArr[3] + " por haber encontrado la
96     }
97     cuatrojuga.push(nombresArr[0]);
98     cuatrojuga.push(puntosArr[0]);
99     cuatrojuga.push(tiempoArr[0]);
100    cuatrojuga.push(nombresArr[1]);
101    cuatrojuga.push(puntosArr[1]);
102    cuatrojuga.push(tiempoArr[1]);
103    cuatrojuga.push(nombresArr[2]);
104    cuatrojuga.push(puntosArr[2]);
105    cuatrojuga.push(tiempoArr[2]);
106    cuatrojuga.push(nombresArr[3]);
107    cuatrojuga.push(puntosArr[3]);
108    cuatrojuga.push(tiempoArr[3]);
109    exportarbtn.addEventListener(MouseEvent.CLICK, exportar4);
110 }
111
112 function exportar1(event: MouseEvent): void {
113     txt.save(unjuga, "Puntajes.txt");
114 }
115
116 function exportar2(event: MouseEvent): void {
117     txt.save(dosjuga, "Puntajes.txt");
118 }
119
120 function exportar3(event: MouseEvent): void {
121     txt.save(tresjuga, "Puntajes.txt");
122 }
123
124 function exportar4(event: MouseEvent): void {
125     txt.save(cuatrojuga, "Puntajes.txt");
126 }

```

*Línea 72-110:* se realiza exactamente lo mismo que la condición pasada, solamente que esta se cumple si la cantidad de jugadores es 4, y el Array utilizado para almacenar los datos de los jugadores es **cuatrojuga**. En este caso, se muestra un mensaje de quién fue el mejor jugador, comparando los puntajes. Se llama a la función **exportar4**.

*Línea 112-126:* funciones por cada Array para exportar los resultados.

### 3. Conclusiones

Con este proyecto nos pudimos dar cuenta de que a simple vista parece demasiado complicado, pero si uno se pone a pensar en maneras más fáciles y óptimas para realizar el trabajo no resulta tan difícil como parecía. Se repitieron muchas funciones y condiciones, lo cual convirtió este trabajo tedioso en cuanto a código, pero no complicado.

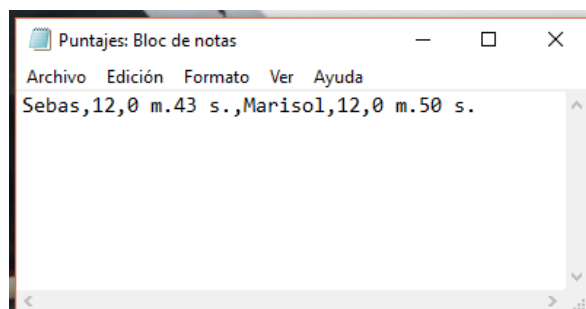
## 4. Anexos



a. Pantalla cuando un jugador pierde



b. Pantalla cuando un jugador gana



c. Documento exportado con resultados