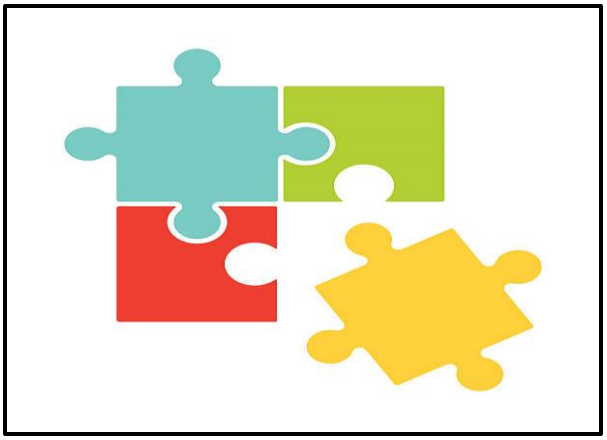
Universidad Politécnica de Victoria

Unidad 1

Proyecto: *Puzzle*



Manual técnico

Integrantes:

- Andrés Fidel García González
  Matrícula: 1730580
- Carlos Daniel Zúñiga González
  Matrícula: 1730142

Materia: Herramientas multimedia

Profesor: Mario Humberto Rodríguez Chávez

Carrera: Ingeniería en tecnologías de la información

Grupo: HM-ITI-07122

Cd. Victoria, Tamaulipas

INDICE

Introducción

El puzzle (rompecabezas) es un juego tradicional que ayuda a agilizar la memoria de las personas, este juego tiene como objetivo el formar una imagen que está dividida en partes aleatorias y esta debe ser formada hasta llegar a su estado original.

A través de este documento técnico hablaremos acerca de nuestro proyecto que fue el desarrollo de un juego de categoría puzzle. A continuación, explicaremos como fue el desarrollo y diseño que se implementaron para darle vida a este proyecto y los métodos lógicos que aplicamos para desarrollar los eventos que tiene cada parte del código.

Objetivos

La temática principal de este proyecto es la realización de un rompecabezas de diferentes tipos de niveles donde los jugadores puedan seleccionar e iniciar sus partidas. El juego del puzzle consta de 4 rompecabezas para armar donde se compiten entre dos jugadores para saber quién tiene la capacidad de armar el puzzle con el menor número de movimientos posibles. También existe la función para rendirse en caso de que un jugador ya no quiera continuar con la partida y darle paso al próximo jugador.

El objetivo principal de este proyecto fue el aplicar los conocimientos obtenido a lo largo de la primera unidad y tratar de utilizar métodos lógicos para cumplir con los requisitos que se requieran para crear el rompecabezas. Realizar todo el proceso trabajando en equipo y obtener retroalimentación de información importante entre nosotros.

Escenario

Frame 1

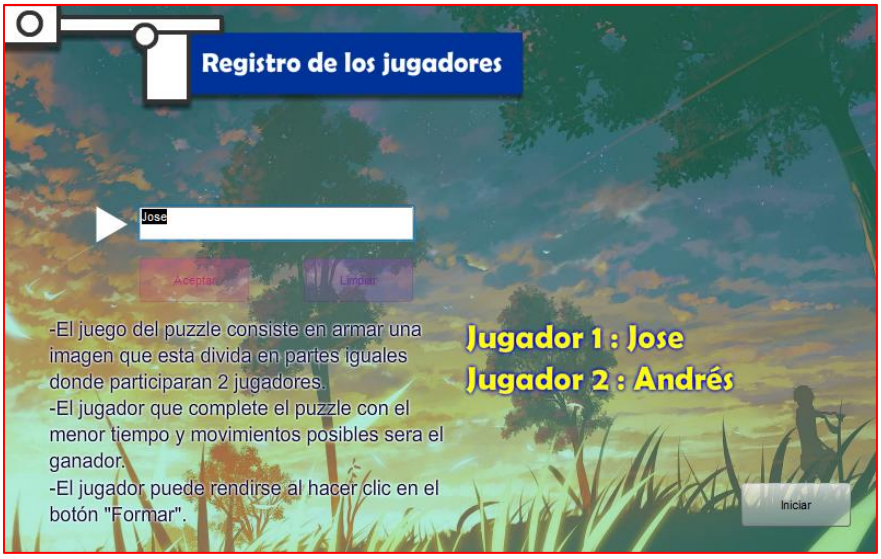
En el fotograma 1 se ubica la portada con diseño y movimiento, mostrando el título de nuestro proyecto junto con los datos importantes del equipo. En la parte inferior central se encuentra el botón para comenzar con el juego. En la portada utilizamos un diseño que combinara con la información mostrada de manera que el usuario pueda entender sin problemas sobre lo que se indica. Se utilizaron diferentes tipos de propiedades para el diseño de letras como el sombreado y la luminosidad.

Frame 2

Se muestra la parte del registro para los competidores del puzzle. El jugador 1 y 2 introducirán un nombre valido para la clasificación del puntaje que tendrá al final de la partida. Debajo del registro se encuentran las reglas a seguir de los jugadores y explican el objetivo para ganar el juego.

Al haberse registrado todos los jugadores, aparecerán sus nombres acompañado de su número de jugador para la partida y en la parte inferior derecha se activará un el botón para Iniciar con el siguiente paso del juego.

Se utilizaron diferentes tipos de colores llamativos para darle vida al fotograma de registros, se utilizaron diferentes propiedades como el sombreado, luminosidad y la aplicación de movimientos al iniciar con el fotograma.



Frame 3

Muestra el menú de selección para el puzzle donde cada una de las imágenes están clasificada por su nivel de dificultad mediante el número de estrellas que posee cada imagen.

El nivel de cada imagen está clasificado según el número de partes divididas que tiene cada puzzle y si el diseño de la imagen dificulta en cómo será formado a la hora de iniciar con el juego.

La selección incluye dos flechas donde los jugadores pueden interactuar y cambiar la imagen, en dirección de izquierda o derecha, y escoger el puzzle haciendo clic en la imagen para dar inicio a la partida.

Frame 4, 5, 6 y 7

Se muestra el nombre del jugador, el tiempo y el número de movimiento para el jugador actual y en la parte inferior derecha se encuentra la forma en que se debe de formar el puzzle.

Al momento de seleccionar la imagen para el puzzle, iremos al fotograma 4 donde se dará inicia a la partida con el jugador 1. Antes de iniciar se muestra una ventana de cuenta regresiva de 3 segundos para empezar con la partida. Una vez iniciado el juego, correrá el tiempo y el juego jugador tendrá el permiso de mover las piezas posibles que se encuentren alrededor el vacío.

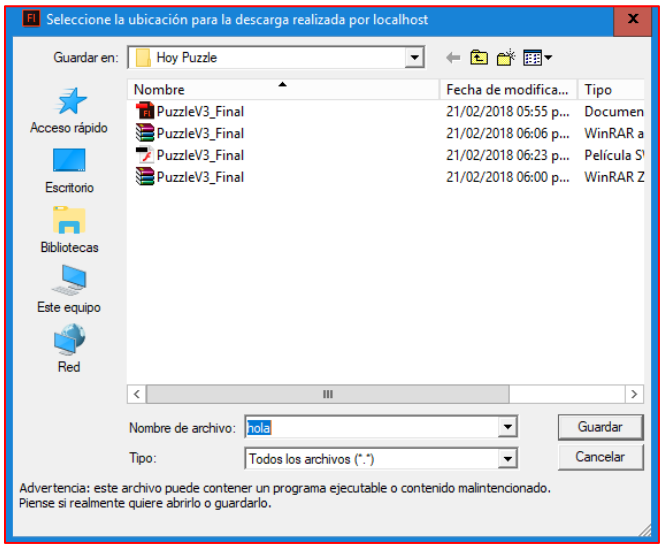
En caso de que el jugador 1 haya formado el rompecabezas, el rompecabezas volverá a desarmarse aleatoriamente y el jugador 2 tendrá el permiso de comenzar con su partida siguiendo los mismos pasos del jugador 1. Si uno de los jugadores se rinde y da clic en el botón “Formar” este perderá automáticamente y regresará a su estado original la imagen.

Si todos los jugadores terminaron su partida, entonces el juego se ha terminado.

Frame 8

Al finalizar la partida entre los dos jugadores la película se dirigirá al siguiente fotograma donde se mostrará los resultados finales que obtuvieron cada uno de los jugadores. En la ventana se observara el nombre de cada jugador acompañado con sus números de movimientos y el tiempo en que se tardó en armar el rompecabezas.

En ese mismo fotograma nos aparecerá también una ventana que nos permite guardar los registros de la partida en un documento txt guardado con el nombre de “hola” para saber cuáles fueron los jugadores que jugaron una partida en el juego del rompecabezas.



Scripts del juego

Los scripts del juego fueron desarrollados en la primera capa llamada “Actions” del archivo flash.

Librerías utilizadas en los scripts

Se utilizaron una variedad de paquetes para la funcionalidad del juego. Se implementaron para los eventos del mouse, la utilización de movimientos con Tween y para que nos permitiera guardar información mediante un archivo tipo txt.

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

Fotograma 1

Se pausa el fotograma actual para poder programar en su entorno.

```
7   //Parar el fotograma.
8   stop();
```

Se declaran variables para usar los efectos de movimientos Tween en cada letra de la palabra “PUZZLE” para tener una presentación agradable al iniciar la película.

Asignación de un movimiento Tween horizontal de derecha a izquierda en el texto material_txt.

Se le asigna un movimiento Tween al botón “Comenzar” en dirección vertical hacia arriba.

```
11  //Efectos de movimientos utilizando Tween.
12  var flm1:Tween = new Tween(p_mc,"z",Back.easeOut,-700, 0,3,true);
13  var flm2:Tween = new Tween(u_mc,"z",Back.easeOut,700, 0,3,true);
14  var flm3:Tween = new Tween(z_mc,"z",Back.easeOut,-700, 0,3,true);
15  var flm4:Tween = new Tween(zz_mc,"z",Back.easeOut,700, 0,3,true);
16  var flm5:Tween = new Tween(l_mc,"z",Back.easeOut,-700, 0,3,true);
17  var flm6:Tween = new Tween(e_mc,"z",Back.easeOut,700, 0,3,true);
18  var flm7:Tween = new Tween(materia_txt,"x",Regular.easeIn,700,218,2,true);
19  var flm8:Tween = new Tween(comenzar_btn,"y",Regular.easeIn,700,418,2,true);
```

Función que nos ayuda a ir al siguiente fotograma donde se encuentra la parte del registro de los jugadores. Esta función se activará cuando el usuario de clic en el botón Comenzar en la película.

```
21    //Funcion para ir al siguiente fotograma.
22    function comenzar(mouse:MouseEvent):void{
23        gotoAndStop(2);
24    }
25    //Activación del evento al hacer clic en el boton Comenzar.
26    comenzar_btn.addEventListener(MouseEvent.CLICK,comenzar);
```

Fotograma 2

Se declara la variable f2m1 en la línea 7 para la utilización del Tween en el título que tiene como instancia el nombre de registroJ_lab. Tiene un movimiento horizontal de izquierda a derecha.

En la línea 10 se declara un contador iniciando con el valor 1, este se utilizará para identificar el número del jugador actual en los registros.

En la línea 13 hasta la 15 se inicializa los arrays principales para almacenar datos de cada jugador:

El array players para almacenar el nombre de cada jugador que participara en el juego, otros arrays para guardar el tiempo y el número de movimientos.

```
7     var f2m1:Tween = new Tween(registroJ_lab,"x",Back.easeOut,-600,238.50,3,true);
8
9     //Declaracion de la variable para definir el numero del jugador
10    var n:int=1;
11
12    //Declaración del arreglo para almacenar el nombre los jugadores.
13    var players:Array=new Array();
```

En el fotograma 21 se desarrolla una función con el nombre “login” y se utiliza el evento del mouse, esta función es la encargada de almacenar el nombre del jugador al hacer clic en el botón “Aceptar”. El evento de la función será activada al hacer clic en el botón “Aceptar” con instancia de “aceptar_btn” como se muestra en la línea 50.

En el fotograma 23 se hace una validación para los datos registrados por el jugador en una condición donde solo permite valores tipo string y no números positivos ni negativos o registros vacíos, en caso de cumplir con la condición, de la línea 25 hasta la 27 muestra un mensaje con Tween diciéndonos que el registro no es válido, y después la casilla de registro con instancia “jugadores_txt” es vaciada para que el mismo jugador intente registrar otro nombre. En caso de que

la condición no se cumpla, continuara con las siguientes líneas evitando las líneas anteriores.

En la línea 39 muestra una condición donde limita el registro hasta 2 jugadores. En caso de cumplir con la condición desaparecerán la información mostrada en la película y los botones de Aceptar y Limpiar serán bloqueado y se activara el botón de “Iniciar”. Al cumplirse la condición, en la línea 47 muestra mediante un texto dinámico “nombres_lab” los registros de los jugadores.

```
21  function login(mouse:MouseEvent):void{
22      //Condición que esta validando los valores, no se permiten numeros.
23      if(Number(jugadores_txt.text) < 0 || Number(jugadores_txt.text) >= 0)
24      {
25          var f2m2:Tween = new Tween(aviso_lab,"x",Regular.easeOut,600,418,2,true); //Mensaje de errores
26         aviso_lab.text="Error, Registro no valido \nVuelve a intentar... ";
27          jugadores_txt.text="";
28      }
29      else
30      {//En caso de no haber errores, se registrará el nombre del jugador en el array "players".
31      n++;
32      aviso_lab.text="";
33      numero_lab.text="Ingresa tu nombre jugador "+n+" :";
34      players.push(jugadores_txt.text);
35      }
36      //Vaciando el nombre ingresado.
37      jugadores_txt.text="";
38      //Validando el limite de jugadores
39      if(n>2){
40          jugadores_txt.text="";
41          aceptar_btn.enabled=false;
42          limpiar_btn.enabled=false;
43          numero_lab.visible=false;
44          sig_btn.enabled=true;
45
46          //Mostrando a los jugadores ingresados
47          nombres_lab.text="Jugador 1 : "+players[0]+"\nJugador 2 : "+players[1];
48      }
49  }
50  aceptar_btn.addEventListener(MouseEvent.CLICK,login);
```

Función llamada “limpiar” que nos permite limpiar en texto de la casilla “jugadores_txt” donde se registran los jugadores. Se activará esta función al hacer clic en el botón Limpiar que tiene como instancia “limpiar_btn”.

```
52  //Funcion para limpiar el registro
53  function limpiar(mouse:MouseEvent):void{
54      jugadores_txt.text="";
55  }
56  limpiar_btn.addEventListener(MouseEvent.CLICK,limpiar);
```

Función con el nombre de “siguiente” que nos permite cambiar al fotograma 3. Esta función se activara al hacer clic en el botón Iniciar que tiene como instancia “sig_btn”.

```
58  function siguiente(mouse:MouseEvent):void{
59      gotoAndStop(3);
60  }
61  sig_btn.addEventListener(MouseEvent.CLICK,siguiente);
```

Frame 3

En esta parte del juego se encuentra la elección del puzzle para comenzar a jugar.

Se pausa el fotograma numero 3.

```
6   //Parar el fotograma.
7   stop();
```

Al iniciar el fotograma 3, se muestra con movimiento Tween la primera imagen junto a su estrella de forma predeterminada.

Muestra solo la primera estrella que tiene como instancia n1_mc y la demás están desactivadas.

```
10  n1_mc.visible=true;
11  n2_mc.visible=false;
12  n3_mc.visible=false;
13  n4_mc.visible=false;
14  star1.visible=true;
15  star2.visible=false;
16  star3.visible=false;
```

Inicialización de la variable “e” con valor de 1, esta variable se utilizara para saber cuál es la imagen que se está mostrando en la selección de puzzles. El valor se estará modificando en el proceso de las funciones “derecha” e “izquierda”.

```
15  var e:int=1;
```

En la línea 20 está la declaración de la variable “select” para trabajar con Tweens.

En la línea 21 hasta la 23 se implementan movimiento al título, la primer imagenen que aparece por predeterminado y una estrella dorada con ayuda del Tween de la variable “select”.

```
19  //Declarando la variable para usar el Tween.
20  var select:Tween;
21  select = new Tween(selecciones_lab,"y",Back.easeOut,-400,51,2,true);
22  select = new Tween(n1_mc,"y",Regular.easeIn,485,240,1,true);
23  select = new Tween(star1,"z",Regular.easeOut,-500,0,1,true);
24
```

En la línea 26 se inicia una función llamada “derecha” que tiene como objetivo cambiar de imagen de izquierda a derecha. Esta función se activa cuando el

usuario da clic en el icono de la flecha izquierda que tiene como instancia “der_btn”.

La función contiene un conjunto de condiciones las cuales se cumplirán de acuerdo al valor que contenga la variable “e” que se estará incrementando cada vez que el usuario de clic al icono. Cada condición muestra una imagen con movimiento y al hacer clic en una de ellas empezara una partida, las imágenes están acompañada con sus estrellas las cuales se estarán mostrando de acuerdo a la dificultad del puzzle.

En la línea 29 al cumplirse la condición se mostrara la imagen solicitada y se activaran 2 estrellas.

```
25  //Función para cambiar el nivel de izquierda a derecha.
26  function derecha(mouse:MouseEvent):void{
27      e++;
28      //Condiciones donde muestran las imagenes y su dificultada dependiendo
29      if(e==2)
30      {
31          select = new Tween(n2_mc,"y",Regular.easeIn,485,240,1,true);
32          n1_mc.visible=false;
33          n2_mc.visible=true;
34          n3_mc.visible=false;
35          n4_mc.visible=false;
36          izq_btn.visible=true;
37          star1.visible=true;
38          star2.visible=true;
39          star3.visible=false;
40          select = new Tween(star1,"z",Regular.easeOut,-500,0,1,true);
41          select = new Tween(star2,"z",Regular.easeOut,-500,0,1,true);
42      }
```

En la línea 44 al cumplirse la condición se mostrara la imagen solicitada y se activaran 2 estrellas.

```
44      if(e==3)
45      {
46          select = new Tween(n3_mc,"y",Regular.easeIn,485,240,1,true);
47          n1_mc.visible=false;
48          n2_mc.visible=false;
49          n3_mc.visible=true;
50          n4_mc.visible=false;
51          star1.visible=true;
52          star2.visible=true;
53          star3.visible=false;
54          select = new Tween(star1,"z",Regular.easeOut,-500,0,1,true);
55          select = new Tween(star2,"z",Regular.easeOut,-500,0,1,true);
56      }
```

En la línea 58 al cumplirse la condición se mostrara la imagen solicitada y se activaran 3 estrellas.

```
58      if(e==4)
59      {
60          select = new Tween(n4_mc,"y",Regular.easeIn,485,240,1,true);
61          n1_mc.visible=false;
62          n2_mc.visible=false;
63          n3_mc.visible=false;
64          n4_mc.visible=true;
65          der_btn.visible=false;
66          star1.visible=true;
67          star2.visible=true;
68          star3.visible=true;
69          select = new Tween(star1,"z",Regular.easeOut,-500,0,1,true);
70          select = new Tween(star2,"z",Regular.easeOut,-500,0,1,true);
71          select = new Tween(star3,"z",Regular.easeOut,-500,0,1,true);
72      }
73      }
74
75  der_btn.addEventListener(MouseEvent.CLICK,derecha);
```

Se inicia una función llamada “izquierda” que tiene como objetivo cambiar de imagen de derecha a izquierda. Esta función se activa cuando el usuario de clic en el icono de la flecha izquierda que tiene como instancia “izq_btn”.

La función contiene un conjunto de condiciones las cuales se cumplirán de acuerdo al valor que contenga la variable “e” que se estará excrementando cada vez que el usuario de clic al icono. Cada condición muestra una imagen con movimiento y al hacer clic en una de ellas empezara una partida, las imágenes están acompañada con sus estrellas las cuales se estarán mostrando de acuerdo a la dificultad del puzzle.

En la línea 81 muestra la condición que al cumplirse mostrara la imagen solicitada de acuerdo al contador “e” y se activaran 2 estrellas.

```
77  //Función para mover la seleccion de imagen de derecha a izquierda
78  function izquierda(mouse:MouseEvent):void{
79      e--;
80      //Condiciones donde muestran las imagenes y su dificultada dependiendo
81      if(e==3)
82      {
83          select = new Tween(n3_mc,"y",Regular.easeIn,485,240,1,true);
84          n1_mc.visible=false;
85          n2_mc.visible=false;
86          n3_mc.visible=true;
87          n4_mc.visible=false;
88          der_btn.visible=true;
89          star1.visible=true;
90          star2.visible=true;
91          star3.visible=false;
92          select = new Tween(star1,"z",Regular.easeOut,-500,0,1,true);
93          select = new Tween(star2,"z",Regular.easeOut,-500,0,1,true);
94      }
```

En la línea 96 al cumplirse la condición mostrara la imagen solicitada y se activaran 2 estrellas.

```
96      if(e==2)
97      {
98          select = new Tween(n2_mc,"y",Regular.easeIn,485,240,1,true);
99          n1_mc.visible=false;
100         n2_mc.visible=true;
101         n3_mc.visible=false;
102         n4_mc.visible=false;
103         der_btn.visible=true;
104         star1.visible=true;
105         star2.visible=true;
106         star3.visible=false;
107         select = new Tween(star1,"z",Regular.easeOut,-500,0,1,true);
108         select = new Tween(star2,"z",Regular.easeOut,-500,0,1,true);
109     }
```

En la línea 111 al cumplirse la condición mostrara la imagen solicitada y se activaran 1 estrella.

```
111         if(e==1)
112         {
113             select = new Tween(n1_mc,"y",Regular.easeIn,485,240,1,true);
114             n1_mc.visible=true;
115             n2_mc.visible=false;
116             n3_mc.visible=false;
117             n4_mc.visible=false;
118             izq_btn.visible=false;
119             star1.visible=true;
120             star2.visible=false;
121             star3.visible=false;
122             select = new Tween(star1,"z",Regular.easeOut,-500,0,1,true);
123         }
124         }
125 izq_btn.addEventListener(MouseEvent.CLICK,izquierda);
```

Funciones que tienen un único proceso, llevarnos a su respectico fotograma donde muestra su partida con el puzzle seleccionado.

Dependiendo de la imagen que se haya elegido nos llevara a ese nivel activando su respectivo evento. Cada imagen tiene su instancia: n1_mc, n2_mc, n3_mc y n4_mc.

```
72  //Función para ir al puzzle numero 1
73  function nivelUno(mouse:MouseEvent):void{
74      gotoAndStop(4);
75  }
76  n1_mc.addEventListener(MouseEvent.CLICK,nivelUno);
77
78  //Función para ir al puzzle numero 2
79  function nivelDos(mouse:MouseEvent):void{
80      gotoAndStop(5);
81  }
82  n2_mc.addEventListener(MouseEvent.CLICK,nivelDos);
83
84  //Función para ir al puzzle numero 3
85  function nivelTres(mouse:MouseEvent):void{
86      gotoAndStop(6);
87  }
88  n3_mc.addEventListener(MouseEvent.CLICK,nivelTres);
89
90  //Función para ir al puzzle numero 4
91  function nivelCuatro(mouse:MouseEvent):void{
92      gotoAndStop(7);
93  }
94
95  n4_mc.addEventListener(MouseEvent.CLICK,nivelCuatro);
```

Frame 4

En el fotograma 4 se muestra el juego en si, que es el rompecabezas de 3x3 donde se tiene que acomodar los cuadros hasta que quede como la imagen original.

¿Cómo funciona?

El funcionamiento del juego consiste en primero guardar las posisiones donde estas los cuadros de la imagen, y la ultima posision se calcula con la penultima.

Es una matriz de 9x3 donde las primeras dos columnas son las posisiones de x y y, y la tercera columna guarda un valor numerico ya sea 0 o 1 donde 0 significia que esa posision esta siendo ocupada por un cuadro de la imagen y 1 significa que hay un cuadro de la imagen en esa posision.

```
var pos:Array = new Array();
pos[0] = [s_F1.x,s_F1.y,1];
pos[1] = [s_F2.x,s_F2.y,1];
pos[2] = [s_F3.x,s_F3.y,1];
pos[3] = [s_F4.x,s_F4.y,1];
pos[4] = [s_F5.x,s_F5.y,1];
pos[5] = [s_F6.x,s_F6.y,1];
pos[6] = [s_F7.x,s_F7.y,1];
pos[7] = [s_F8.x,s_F8.y,1];
pos[8] = [s_F8.x+s_F8.width,s_F8.y,0];
```

Despues los cuadros de la imagen se ordenan de forma aleatoria en las posisiones ya guardadas.

```
while(ran.length < 8){
  //Se asigna un numero aleatorio
  num = (Math.random()*8);
  //La bandera se asigna como falsa
  b = false;
  //En un ciclo que va de 0 hasta la longitud d
  for(xd = 0; xd < ran.length; xd++){
    //Si es igual a algun elemento del arreglo
    if(ran[xd] == num){
        //se activa la bandera, se le asigna tr
        b = true;
        break;
    }
  }
  //Si la bandera esta en falso
  if(b == false){
    //Se captura el numero aleatorio en el arre
    ran[ran.length] = num;
  }
}
//Ya que se tiene numeros aleatorios del 0 al 8
for(xd = 0; xd < ran.length; xd++){
    this["s_F"+(xd+1)].x = pos[ran[xd]][0];
    this["s_F"+(xd+1)].y = pos[ran[xd]][1];
    pos[xd][2] = 1;
}
pos[8][2] = 0;
}
```

Todos los cuadros se añaden a una sola funcion de evento de raton.

```
s_F1.addEventListener(MouseEvent.CLICK,onClick);
s_F2.addEventListener(MouseEvent.CLICK,onClick);
s_F3.addEventListener(MouseEvent.CLICK,onClick);
s_F4.addEventListener(MouseEvent.CLICK,onClick);
s_F5.addEventListener(MouseEvent.CLICK,onClick);
s_F6.addEventListener(MouseEvent.CLICK,onClick);
s_F7.addEventListener(MouseEvent.CLICK,onClick);
s_F8.addEventListener(MouseEvent.CLICK,onClick);
```

Dentro de la funcion, dependiendo de cual sea la posision vacia va preguntar si el cuadro que se clickeo esta alrededor del vacio, y si es asi, entonces se intercambiara la posision y ahora la posision vacia pasara a ser la del cuadro.

En este ejemplo se muestra que si la posision de la esquina superior izquierda es la posision vacia, va preguntar si el cuadro que le sigue a la derecha o el que esta debajo es el que se clickeo. Si es asi entonces la posision vacia pasara a tener un cuadro de la imagen y la posision donde estaba la imagen quedara vacia, posteriormente se hace el movimiento con un Tween desde la posision donde estaba el cuadro hasta la posision vacia.

Los movimientos van incrementando cada vez que se mueve un cuadro de lugar.

```
function onClick(evt:MouseEvent):void {
    if(pos[0][2] == 0){
        if(this[evt.target.name].x == pos[1][0] && this[evt.target.name].y == pos[1][1]){
            pos[0][2] = 1;
            pos[1][2] = 0;
            Tw = new Tween(this[evt.target.name],"x",Regular.easeIn,pos[1][0],pos[0][0],1,true);
            ban = true;
            mov++;
        }else{
            if(this[evt.target.name].x == pos[3][0] && this[evt.target.name].y == pos[3][1]){
                pos[0][2] = 1;
                pos[3][2] = 0;
                Tw = new Tween(this[evt.target.name],"y",Regular.easeIn,pos[3][1],pos[0][1],1,true);
                ban = true;
                mov++;
            }
        }
    }
```

Para saber si el jugador ha ganado el juego se verifica que cada cuadro de la imagen este en la posision donde debe ir.

Esto esta constantemente siendo verificado atravez de una funcion de evento de Timer.

```
if(s_F1.x == pos[0][0] && s_F1.y == pos[0][1] && s_F2.x == pos[1][0] && s_F2.y == pos[1][1]
    && s_F3.x == pos[2][0] && s_F3.y == pos[2][1] && s_F4.x == pos[3][0]
&& s_F4.y == pos[3][1] && s_F5.x == pos[4][0] && s_F5.y == pos[4][1]
&& s_F6.x == pos[5][0] && s_F6.y == pos[5][1] && s_F7.x == pos[6][0]
&& s_F7.y == pos[6][1] && s_F8.x == pos[7][0] && s_F8.y == pos[7][1])
{
//Muestra un mensaje en caso de haber completado el puzzle.
ganador_lab.text="El puzzle ha sido completado";
```

Para rendirse se uso una funcion de evento de raton donde se termina el turno del jugador que se rindio y le asigna 0 de puntuacion. Y se pone un mensaje de preparacion para el siguiente jugador si es que hay. Esto tambien sucede cuando se gana.

```
function formar(mouse:MouseEvent):void{

    //Muestra un mensaje en caso no poder armar el puzzle.
    ganador_lab.text="Te has rendido";
    //Contador donde indica cual sera el proximo jugador.
    Jp++;
    //Contador para usar el arreglo donde esta almacenador el nombre de los jugadores
    p++;
    //Reiniciando la cuenta regresiva.
    round=4;
    //Reiniciando los minutos, segundos y movimientos de las variables.
    seg=0;
    min=0;
    mov=0;
    times.push("0:00");
    score.push("0");
    //Mostrando las ventanas para la cuenta regresiva.
    ready_mc.visible=true;
    ronda_lab.visible=true;
    cuenta_lab.visible=true;
    terminado_mc.visible=true;
    timer.stop(); //Pausando el tiempo.
    partida.start(); //Iniciando la cuenta regresiva.

        //Condicion donde nos lleva al siguiente fotograma cuando todos las partidad sean terminadas.
        if(p > 1){
            timer.stop();
            partida.stop();
            gotoAndStop(8);
        }
}
```

Despues de que gane o se rinda el jugador 1, pasa a jugar el jugador 2, una vez que hayan ganado los dos jugadores o se hayan rendido, pasaran a un frame donde mostraran sus nombres, movimientos y cuanto tiempo se tardaron.

Frame 8

En esta parte del juego se guardan los resultados de los jugadores en un archivo de texto.

Primero se carga el archivo si es que existe.

```
function cargarlista (e:Event){
    //Se captura lo del txt que se cargo
    datosTXT = ""+e.target.data;
    //Se manda llamar a las funciones de RecuperarTxt y guardar
    RecuperarTxt();
    guardar();
}
//Se crea el cargador del txt
var cargador:URLLoader= new URLLoader(new URLRequest("\hola.txt"));
//Se le añade el evento para cargarlo
cargador.addEventListener(Event.COMPLETE,cargarlista);
```

Si no existe entonces a la variable donde se suponia que iba guardar los datos del archivo de texto solo se inicializa con los titulos de las columnas del archivo de texto.

Se añade un evento para en caso de un error, como al cargar el archivo.

```
cargador.addEventListener(IOErrorEvent.IO_ERROR, loadIOError);
//Funcion en caso de que no exista el archivo txt
function loadIOError(event:IOErrorEvent):void{
    //Se le asigna a la variable del txt solo el encabezado
    datosTXT = "Nombre      Score       Tiempo      "+"\n";
    //Se manda llamar a las funciones de RecuperarTxt y guardar
    RecuperarTxt();
    guardar();
}
```

Primero se extrae lo que tenga el archivo de texto si es que tiene algo lo guarda en los mismos arreglos donde se estaban guardando los datos de los jugadores recientes.

Va comparando carácter por carácter y mientras sea diferente de un espacio o un salto de línea, va ir concatenando lo que tenga en una variable, luego se asignara dependiendo del dato que sea a un arreglo especifico.

El primer dato es un nombre el segundo la cantidad de movimientos y el tercer el tiempo, con esto sabemos dónde ir guardando cada dato.

```
function RecuperarTxt():void{
    //En un ciclo que va desde el caracter 31 hasta el ultimo ya que los primeros datos son los titulos de los datos (Nombre,Score,Tiempo)
    for(i = 31; i < datosTXT.length; i++){
        //Va checando caracter por caracter, si el caracter es diferente de un expacio y si tambien es diferente de un salto de linea
        if(datosTXT.substring(i,i+1) != " " && datosTXT.substring(i,i+1) != "\n"){
            //bz se vuelve verdadera
            bz = true;
        }else{
            //Si es igual a un espacio o salto de linea
            //Si la variable aux no tiene nada guardado
            if(aux != ""){
                //Si tiene algo entonces se alamcena en los arreglos
                //Si es la primera vez entonces se guarda en los nombres
                if(cont == 0){
                    players.push(aux);
                }
                //Si es la segunda vez se guarda en los score
                if(cont == 1){
                    score.push(aux);
                }
                //Si es la tercera vez entonces se guarda en tiempo
                if(cont == 2){
                    times.push(aux);
                    //Se reincia para volver a almacenar el siguiente dato que es un nombre en el arreglo que corresponde
                    cont = -1;
                }
                //Se aumenta el contador para saber donde se deben guardar los datos
                cont++;
            }
            //bz es igual a falso
            bz = false;
            //La variable aux se le asignan comillas
            aux = "";
        }
        //Si bx es verdadera entonces se concatena el caracter en la variable aux
        if(bz == true){
            aux += datosTXT.substring(i,i+1);
        }
    }
    //Se reincian las variables
    cont = 0;
    aux = "";
}
```

Después se ordenan los datos para saber quién es el jugador que menos movimientos ha hecho sin contar a los que se han rendido.

```
for (j = 0; j < cant; j++){
    for (i = j+1; i < cant; i++){
        //Si el segundo elemento es mayor que el primero entonces se cambian pero
        if (Number(score[j]) > Number(score[i]) && (Number(score[i]) != 0)){
            //Del mismo modo tambien se cambian los elementos de los otros arreglc
            auxd = score[j];
            score[j] = score[i];
            score[i] = auxd;
            auxs = players[j];
            players[j] = players[i];
            players[i] = auxs;
            auxa = times[j];
            times[j] = times[i];
            times[i] = auxa;
        }
    }
}
```

Después se concatena en la variable donde estará la información que se guardara en el archivo de texto y se acomoda dándole espacios entre los campos para tener una forma de tabla.

Por último se guarda en el archivo de texto y se concatenan los arreglos es textos dinámicos para mostrarlo en el juego que se está ejecutando.

```
    datosTXT = "Nombre      Score       Tiempo      "+"\n";
    //Se captura la cantidad de elementos del arreglo
    cant = players.length;
    //En este ciclo se añaden espacios dinamicos a los elementos de los arreglos en variables auxiliares
    for(i = 0; i < cant; i++){
        nom = players[i];
        //Se calculan cuantos espacios falta para tener 12
        long = 12-nom.length;
        //Se agregan los espacios que faltan
        for(j = 0; j < long; j++){
            nom += " ";
        }
        mox = score[i];
        //Se calculan cuantos espacios falta para tener 12
        long = 12-mox.length;
        //Se agregan los espacios que faltan
        for(j = 0; j < long; j++){
            mox += " ";
        }
        ti = times[i];
        //Se calculan cuantos espacios falta para tener 12
        long = 12-ti.length;
        //Se agregan los espacios que faltan
        for(j = 0; j < long; j++){
            ti += " ";
        }
        //Se concatenan los datos con espacios dinamicos a la variables datosTXT y luego se da un salto de linea
        datosTXT += nom+mox+ti+"\n";
    }
    //Una vez acomodados los datos en la variable datosTXT se guarda en un archivo txt
    saveF.save(datosTXT,"hola.txt");
    //En este ciclo se concatenan en textos dinamicos los datos de los jugadores recien guardados en el txt
    for(i = 0; i < players.length; i++){
        sal_Nombre.text += players[i]+"\n";
        sal_Mov.text += score[i]+"\n";
        sal_Tiempo.text += times[i]+"\n";
    }
}
```

Conclusión

Gracias a este proyecto hemos aprendido a tener un mejor trabajo en equipo, hemos desarrollado nuestra lógica y hemos compartido y respetado nuestras ideas, para así elegir la mejor manera de hacer las cosas.

Este proyecto nos ayudó a agilizar nuestro pensamiento lógico gracias a la solución de resolver el puzzle y el saber en cómo funciona su mecánica, desarrollar un algoritmo que permita a las piezas desplazarse un destino