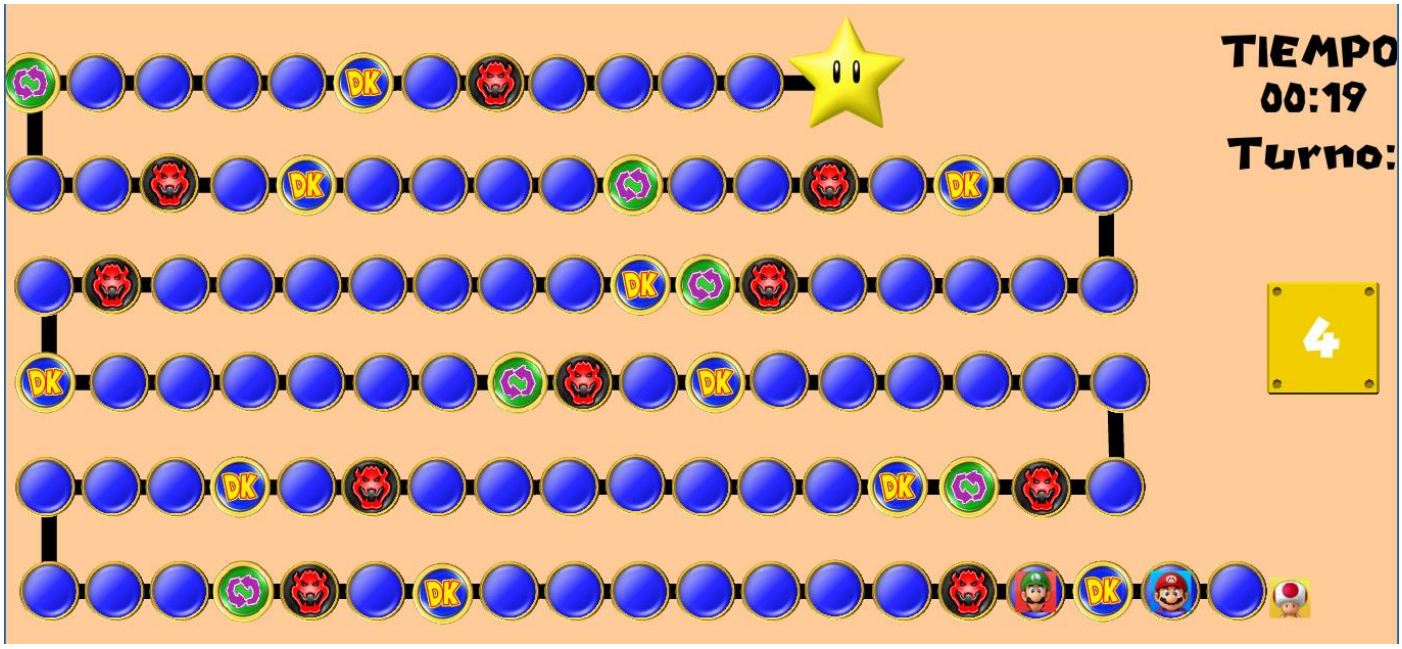
MANUAL TECNICO SERPIENTES Y ESCALERAS



UNIVERSIDAD POLITECNICA DE CIUDAD VICTORIA

MSI. MARIO HUMBERTO RODRIGUEZ CHAVEZ

INGENIERIA EN TECNOLOGIAS DE LA INFORMACIÓN

HERRAMIENTAS MULTIMEDIA - ITI-07116

ALUMNOS:

HERNANDEZ MARTINEZ EDWIN 1730005

ZUÑIGA LEDEZMA JUAN PABLO 1730097

INDICE

PORTADA…… 1
INTRODUCCION…… 3
MARCO TEORICO…… 4
EXPLICACION DE CODIGO DE FRAME A FRAME…… 5
FRAME PORTADA…… 5
FRAME INSTRUCCIONES…… 6
FRAME INTRODUCIR NOMBRES…… 7
FRAME CARGANDO…… 10
FRAME TABLERO SERPIENTES Y ESCALERAS…… 11
FRAME GANADOR…… 16
CONCLUSION…… 17

INTRODUCCION

Los juegos de azar son juegos en los cuales las posibilidades de ganar o perder no dependen exclusivamente de la habilidad del jugador, sino que interviene también el azar. La mayoría de ellos son también juegos de apuestas, cuyos premios están determinados por la probabilidad estadística de acertar la combinación elegida; mientras menores sean las probabilidades de obtener la combinación correcta mayor es el premio.

El azar es una casualidad presente, teóricamente, en diversos fenómenos que se caracterizan por causas complejas, no lineales y sobre todo que no parecen ser predictibles en todos sus detalles. Dependiendo del ámbito al que se aplique, se pueden distinguir cuatro tipos de azar:

- **Azar en matemáticas**. En matemáticas, pueden existir series numéricas con la propiedad de no poder ser obtenidas mediante un algoritmo más corto que la serie misma. Es lo que se conoce como aleatoriedad. La rama de las matemáticas que estudia este tipo de objetos es la teoría de la probabilidad. Cuando esta teoría se aplica a fenómenos reales se prefiere hablar de estadística.

- **Azar en la física**. El azar puede darse en sistemas físicos indeterministas y deterministas. En los sistemas no se puede determinar de antemano cuál será el suceso siguiente, como sucede en la desintegración de un núcleo atómico. Esta dinámica, azarosa, es intrínseca a los procesos que estudia la mecánica cuántica (subatómicos). Dentro de los procesos deterministas, también se da el azar en la dinámica de sistemas complejos impredecibles, también conocidos como sistemas caóticos.

MARCO TEORICO

Serpientes y escaleras es un antiguo juego de tablero indio, considerado actualmente como un clásico a nivel mundial. Se juega entre dos o más personas en un tablero numerado y dividido en casilleros, que posee además un número determinado de serpientes y escaleras que conectan, cada una, dos casilleros numerados. El movimiento se determina en la actualidad por un disco giratorio cuyos valores están comprendidos entre 1 y 6, o por medio de un dado.

El objetivo del juego es lograr que la ficha del jugador llegue desde el inicio (casillero inferior izquierdo) hasta el final (casillero superior izquierdo), ayudado por las escaleras y evitando las serpientes. La versión histórica nace de algunas lecciones de moral, donde el progreso de un jugador en el tablero representa una vida influida por virtudes (representadas por las escaleras) y por vicios (serpientes).

EXPLICACION DE CODIGO DE FRAME A FRAME

FRAME PORTADA.

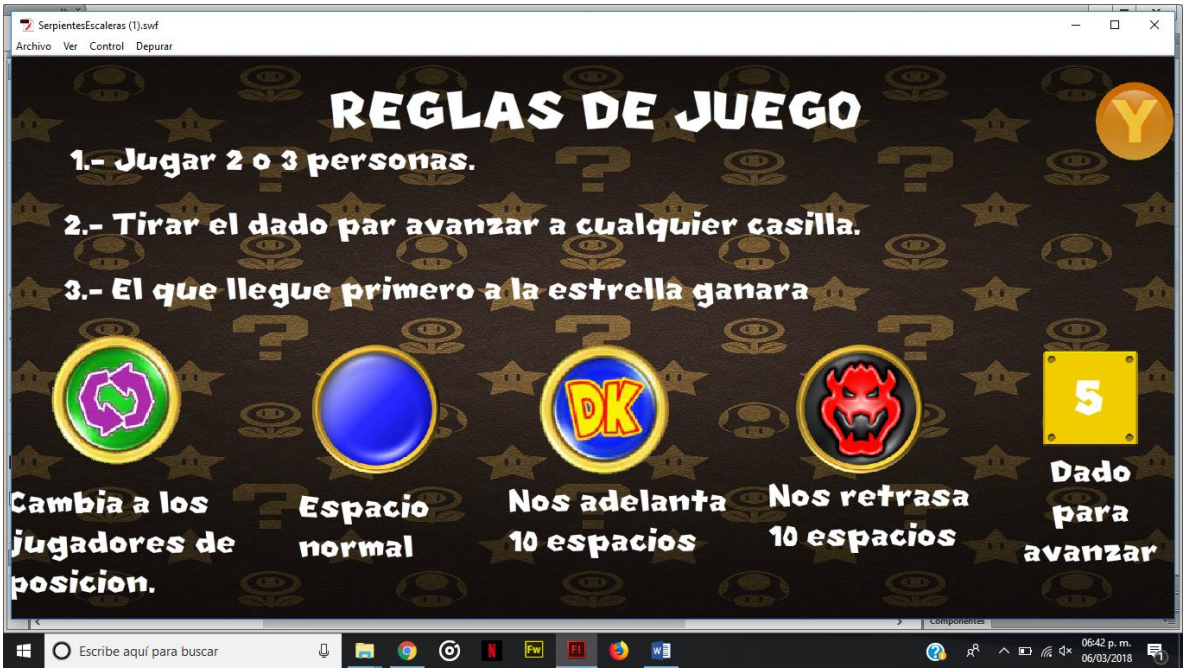
En pantalla vemos datos personales de nuestro equipo de trabajo, el cual trato de hacer la mejor realización de este proyecto. Tratando de implementar el juego Mario Party a el juego de serpientes y escaleras. Tanto los datos como el botón “Y” están animados, así como lo muestra la imagen. En la siguiente imagen (**imagen1**) mostramos el código de dicho diseño.

```
import fl.transitions.easing.*;
import fl.transitions.Tween;


//Nos detenemos en el primer footgrama
stop();
//Tweens de animación que nos permiten mover nuestros datos y boton para dar un mejor diseño en pantalla
var uno:Tween = new Tween(edwin, "x", Strong.easeIn,-1000,360,5,true);
var dos:Tween = new Tween(pablo, "x", None.easeIn,20000,348,5,true);
var tres:Tween = new Tween(iti, "x", Strong.easeIn,2000,636,5,true);
var cuatro:Tween = new Tween(profem, "x", None.easeIn,-1000,348,5,true);
var cinco:Tween = new Tween(uni2, "x", Strong.easeInOut,2000,430,5,true);
var seis:Tween = new Tween(empezar_btn, "x", None.easeIn,-1000,686,5,true);

//Funcion para pasar al siguiente fotograma mediante un enter
function presionar(event:MouseEvent):void{
    gotoAndStop(5);
}

//Le asignamos la funcion al boton
empezar_btn.addEventListener(MouseEvent.CLICK, presionar);
```

Imagen1

Este código nos ayuda a la detención de nuestro Frame, así como también a la edición de nutra portada, nos ayuda a acomodar nuestros datos personales de una manera mas formal, y a que nuestro botón “Y” nos lleve al siguiente Frame.

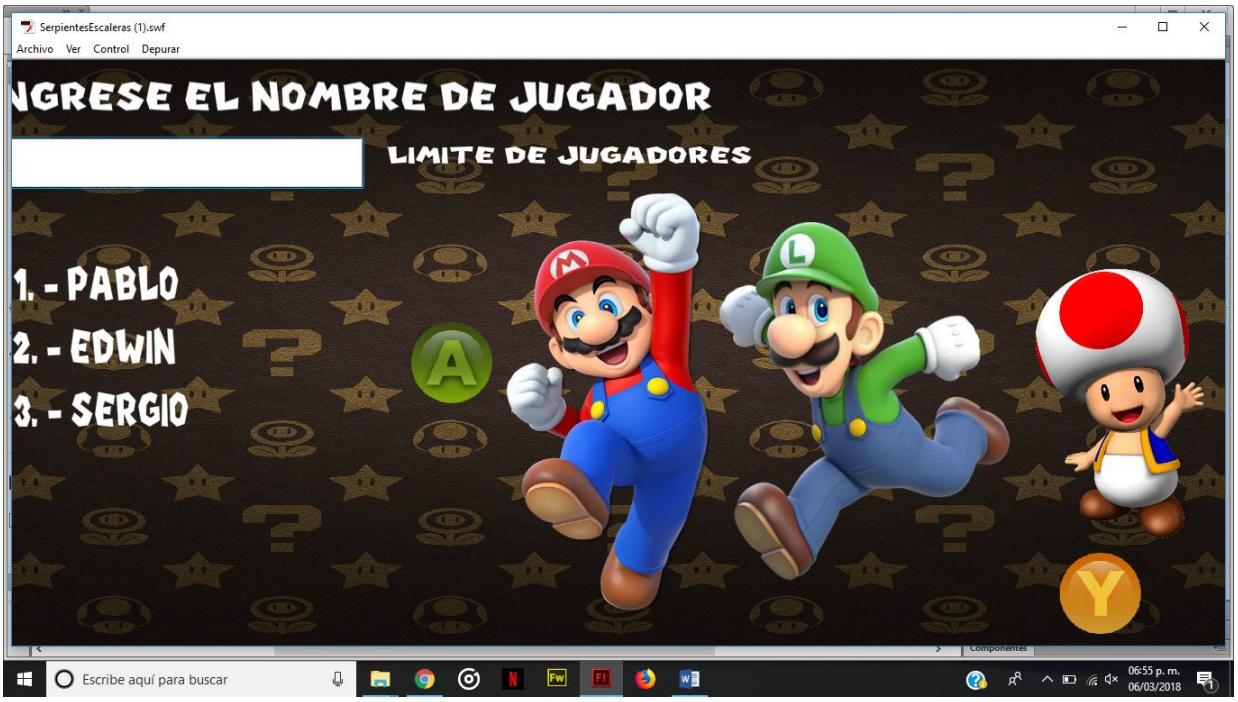
FRAME INSTRUCCIONES.



En este frame tenemos la explicación breve de lo que hace cada uno de los factores que nos ayudan en nuestro juego, así como las reglas. En este caso explicamos cómo los espacios pueden beneficiarnos o ser lo contrario a ello.

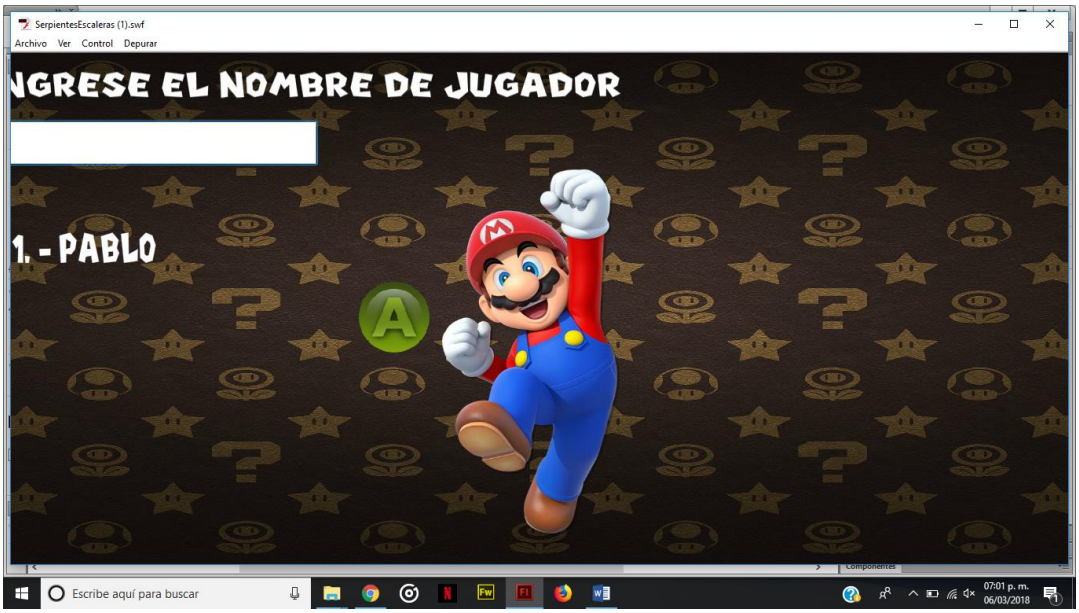
También colocamos el dado para la identificación de él al momento de entrar a nuestro juego, y tenemos un botón “Y” en la parte superior derecha para conducirnos al segundo frame; en cual utilizamos una función para encaminarnos al siguiente frame como lo tenemos en el anterio frame(**imagen1**).

FRAME INTRODUCIR DE NOMBRES.



En este frame almacenamos os nombres de los jugadores, y dependiendo de los nombres de los jugadores es como tomamos el número de los jugadores que van a empezar la partida. En este caso explicábamos anteriormente que mínimo para iniciar una partida debemos tener **2 jugadores** y el límite son **3 jugadores.**

Cada que introducimos un jugador nos parecerá un avatar el cual nos va a representar como jugadores, como se muestra en la **imagen2**



[**imagen2]**

De este modo dentro de las entrañas de nuestro frame encontramos una serie de pasos que nos almacenan los datos de nuestros jugadores y nos apoyan en le ejecución del juego.

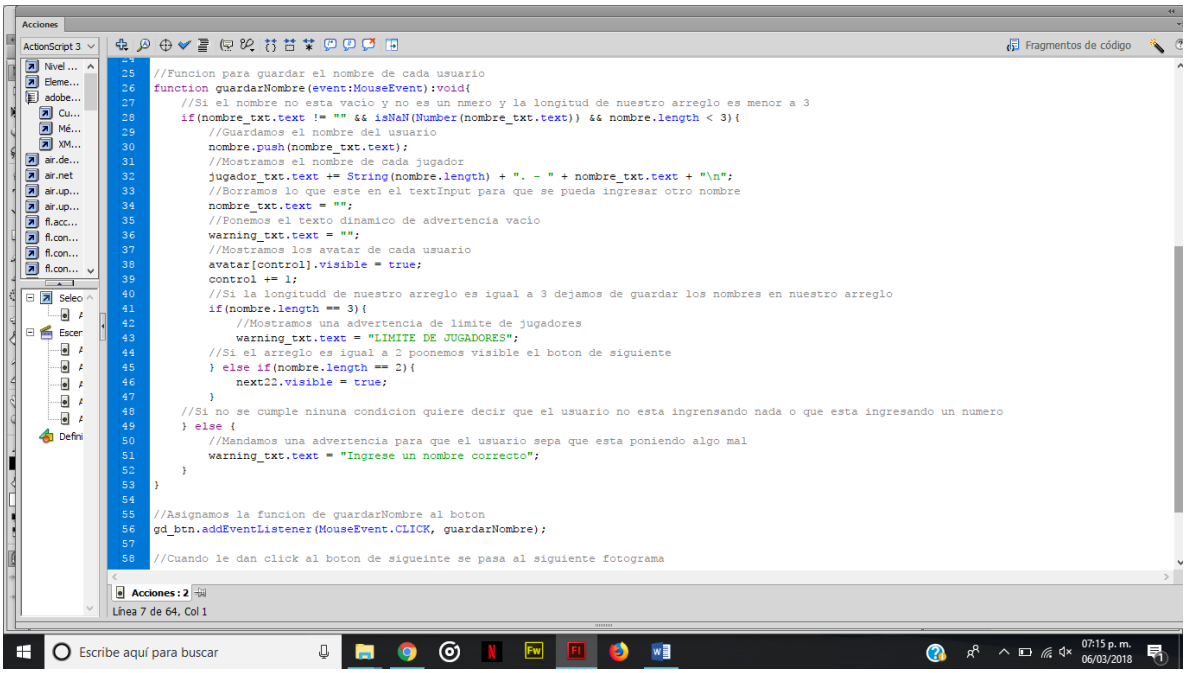
```
//Imprtamos ls librerias que se van a usar
import flash.events.MouseEvent;
import flash.ui.Mouse;

//Nos detenemos en el fotogrma
stop();

//Arrerglo para guardar las imagenes de los avatar
var avatar:Array = new Array(mario1, luigi1, toad1);

//Al entrar al arreglo los ponemos invisibles
for(var i:int = 0; i < 3; i++){
    avatar[i].visible = false;
}

//Variable para controlar las posiciones del arreglo
var control:int = 0;

//Arraglo para guardar los nombres de los jugadores
var nombre:Array = new Array();

//Ponemos el boton de siguiente en invisible
next22.visible = false;

//Funcion para guardar el nombre de cada usuario
function guardarNombre(event:MouseEvent):void{
    //Si el nombre no esta vacio y no es un nmero y la longitud de nuestro arreglo es menor a 3
    if(nombre_txt.text != "" && isNaN(Number(nombre_txt.text)) && nombre.length < 3){
        //Guardamos el nombre del usuario
        nombre.push(nombre_txt.text);
        //Mostramos el nombre de cada jugador
        jugador_txt.text += String(nombre.length) + ". - " + nombre_txt.text + "\n";
        //Borramos lo que este en el textInput para que se pueda ingresar otro nombre
        nombre_txt.text = "";
        //Ponemos el texto dinamico de advertencia vacio
```

Acciones : 2

Línea 7 de 64, Col 1

[imagen3]

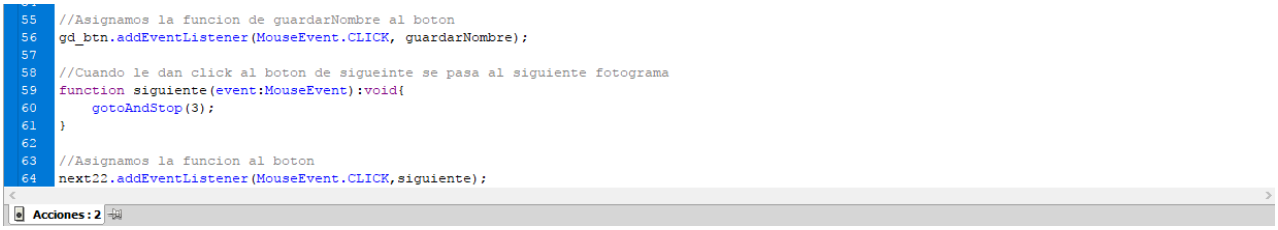
En la **imagen3** podemos observar que declaramos un arreglo. Dicho arreglo nos ayuda a almacenar los valores que son introducidos, estos valores solo pueden letras. Tenemos un contador que nos apoya en el conteo de las entradas de nuestro arreglo de nombres. Y un arreglo avatar que es el que nos ayuda que aparezcan dependiendo de lo que vayamos almacenando en nuestro frame.



```
//Funcion para guardar el nombre de cada usuario
function guardarNombre(event:MouseEvent):void{
    //Si el nombre no esta vacio y no es un nmero y la longitud de nuestro arreglo es menor a 3
    if(nombre_txt.text != "" && isNaN(Number(nombre_txt.text)) && nombre.length < 3){
        //Guardamos el nombre del usuario
        nombre.push(nombre_txt.text);
        //Mostramos el nombre de cada jugador
        jugador_txt.text += String(nombre.length) + ". - " + nombre_txt.text + "\n";
        //Borramos lo que este en el textInput para que se pueda ingresar otro nombre
        nombre_txt.text = "";
        //Ponemos el texto dinamico de advertencia vacio
        warning_txt.text = "";
        //Mostramos los avatar de cada usuario
        avatar[control].visible = true;
        control += 1;
        //Si la longitudd de nuestro arreglo es igual a 3 dejamos de guardar los nombres en nuestro arreglo
        if(nombre.length == 3){
            //Mostramos una advertencia de limite de jugadores
            warning_txt.text = "LIMITE DE JUGADORES";
        //Si el arreglo es igual a 2 poonemos visible el boton de siguiente
        } else if(nombre.length == 2){
            next22.visible = true;
        }
    //Si no se cumple ninuna condicion quiere decir que el usuario no esta ingrensando nada o que esta ingresando un numero
    } else {
        //Mandamos una advertencia para que el usuario sepa que esta poniendo algo mal
        warning_txt.text = "Ingrese un nombre correcto";
    }
}

//Asignamos la funcion de guardarNombre al boton
gd_btn.addEventListener(MouseEvent.CLICK, guardarNombre);

//Cuando le dan click al boton de sigueinte se pasa al siguiente fotograma
```

[imagen4]

En la **imagen4** tenemos una función que nos ayuda a almacenar los valores a nuestro arreglo al momento de presionar el botón “A” (Este botón lo podemos ver en la **imagen2**). Pero para alamacenar estos valores, primero verifica por medio de condiciones que no sea numérico, que contenga un valor positivo, es decir que no sea nulo y que no sobre pase los limites de 3 jugadores.

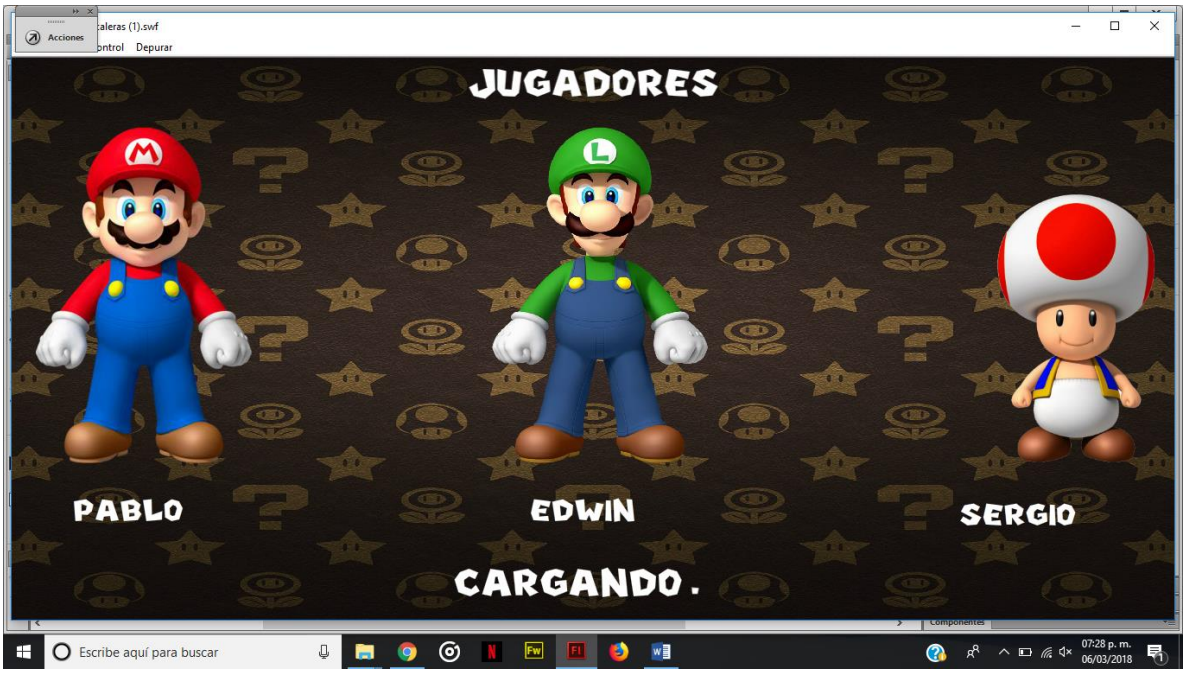


```
//Asignamos la funcion de guardarNombre al boton
gd_btn.addEventListener(MouseEvent.CLICK, guardarNombre);

//Cuando le dan click al boton de sigueinte se pasa al siguiente fotograma
function siguiente(event:MouseEvent):void{
    gotoAndStop(3);
}

//Asignamos la funcion al boton
next22.addEventListener(MouseEvent.CLICK,siguiente);
```

[imagen5]

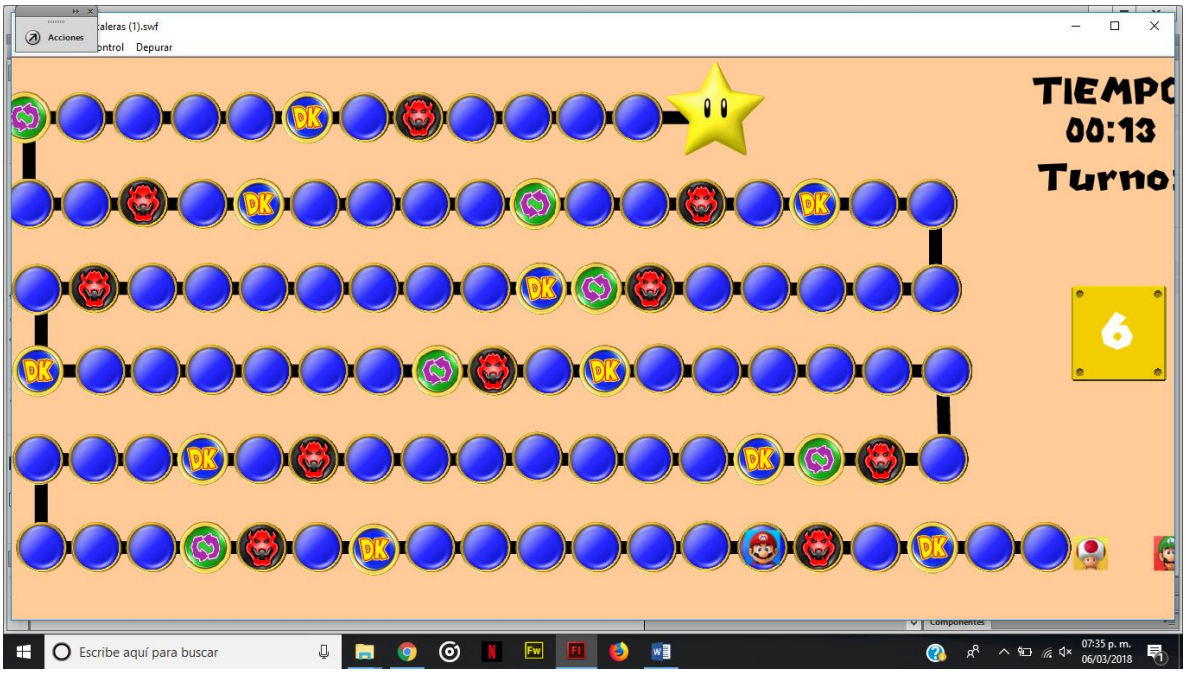
En la **imagen5** encontramos una función que nos ayuda a conducirnos al siguiente frame al momento de presionar el botón “Y” (este botón lo encontramos en la primer imagen de esta sección).

FRAME CARGANDO...



En este frame solo usamos un poco de estética, solo al diseñar los puntos de cargando y colocando los nombres de cada jugador y poniéndolos debajo de su avatar para que así puedan saber que avatar les ha tocado al empezar el juego.

FRAME TABLERO SERPIENTES Y ESCALERAS



En esta imagen mostramos detalladamente como esta nuestro tablero de serpientes y escaleras. Donde ya explicábamos en anteriores secciones lo que significaba cada uno de nuestro espacio. Ahora explicaremos detalladamente lo que hay detrás de éste tablero.

```
//Nos detenemos en el fotograma
stop();

//Arreglo donde guardamos todas las casillas
var casillas:Array = new Array(c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, c13, c14, c15, c16, c17, c18, c19,
                               c20, c21, c22, c23, c24, c25, c26, c27, c28, c29, c30, c31, c32, c33, c34, c35, c36,
                               c37, c38, c39, c40, c41, c42, c43, c44, c45, c46, c47, c48, c49, c50, c51, c52, c53,
                               c54, c55, c56, c57, c58, c59, c60, c61, c62, c63, c64, c65, c66, c67, c68, c69, c70,
                               c71, c72, c73, c74, c75, c76, c77, c78, c79, c80, c81, c82, c83, c84, c85, c86, c87,
                               c88, c89, c90, c91, c92, c93, c94, c95, c96, c97, c98, c99, c100);


//Bandera para saber si entro a una condicion de casilla especial
var bandDK:Boolean;
var bandBow:Boolean;
var bandRotar:Boolean;

var auxPos:int;

//Guardamos a nuestros jugadores en un arreglo
var avatarJugador:Array = new Array(jugador11, jugador22, jugador33);

//Si soloson 2 jugadores ponemos el tercer avatar invisible
if(nombre.length == 2){
    avatarJugador[2].visible = false;
}

//Variable para saber cual es la siguiente posicion del usuario
var posSiguiente1:int = -1;
var posSiguiente2:int = -1;
var posSiguiente3:int = -1;

//Variable para saber el turno de cada jugador
var turno:int = 1;
```

Dentro de nuestro código encontramos un arreglo que almacena todas las posiciones de nuestro tablero, de la 1 a 100. También tenemos banderas de apoyo las cuales no servirán en las demás líneas de código, declaramos un arreglo para los jugadores que van a disputar el encuentro. Colocamos una variable que nos indica la posición más próxima del usuario. Y una condición para saber cuántos jugadores van a jugar.

```
//Varialbes para el timer
var tmp:int = 0;
var cont1:int = 0;
var reloj:int = 0;
var timer:Timer = new Timer(1000,cont1++);
timer.start();


//Funcion para contar los segundos y minutos
function tiempo(tiempoevent:TimerEvent):void{
    //Incrementamos el contador de segundos
    tmp++;
    //Si el contador de segundos es menor a 10 le agregamos un 0 antes del
    if(tmp < 10){
        timer_mc.text = "0" + reloj + ":0" + tmp;
        //Si el contador es igual a 60 en lugar de poner el 60 inicializamos el contador en 0 y aumentamos
        //en 1 la variable de los minutos
    } else if(tmp == 60){
        reloj += 1;
        tmp = 00;
        timer_mc.text = "0" + reloj + ":" + tmp + "0";
        //Si el contador de los segundos esta de 10 a 59 solo ponemos los minutos y después los segundos
        //sin ningún 0
    } else {
        timer_mc.text = "0" + reloj + ":" + tmp;
    }
}
//Asignamos la funcion para contar el tiempo que llevan jugnado al timer
timer.addEventListener(TimerEvent.TIMER, tiempo);

//Variables para el timer del dado
var tmp2:int = 00;
var cont2:int;
var timer2:Timer = new Timer(10,cont2++);
```

En esta función hacemos la realización del timer, el timer nos ayuda a saber cuanto tiempo llevamos jugando con nuestros amigos.

```
//Funcion para el random del dado
function dadoRandom(tiempoEvent:TimerEvent):void{
    //Incrementamos la variable de tmp
    tmp2++;
    //Sacamos el random del dado, lo multiplicamos por 7 para que nos de un resultado de 6
    randomDado = Math.random() * 7;
    //Lo mostramos en el texto dinamico que tenemos
    dado_txt.text = String(randomDado);
    //En el caso en que nos haya arrojado un 0 ponemos un ciclo que mientras nos siga dando un 0 siga sacando el randomm
    while(randomDado == 0){
        randomDado = Math.random() * 7;
        dado_txt.text = String(randomDado);
    }
}

//Funcion para el presionado del dado
function dadoRan(event:MouseEvent):void{
    //Ponemos una condicion con la cual evaluamos una bandera, si tiene un valor falso entonces realiza lo siguiente
    if(click == false){
        //Empieza el timer del dado
        timer2.start();
        //Asignamos la funcion al timer
        timer2.addEventListener(TimerEvent.TIMER, dadoRandom);
        //Ponemos click en true para que el usuario detenga el boton
        click = true;
    //Si la bandera es verdadera entonces se realiza lo siguiete
    } else if(click == true){
        //Detenemos el timer
        timer2.stop();
        //Mandamos llamar la funcion de mover objetos la cual contiene los tweens para mover a los jugadores
        moverObjetos();
        //Ponemos la bandera la cual controla los click en falso
        click = false;
    }
```

En estas funciones las utilizamos para crear nuestro dado, en la primera corremos de 1 al 6 un random que nos llevará a un número aleatorio, y de esta manera en las siguientes líneas haremos que dependiendo de ese número avance nuestra ficha. En la sgunda función presionamos el botón “A” para iniciar el random del dado, y el mismo botón para pararlo y desaparecer, para que otros usuarios no puedan presionarlo. Esto nos ayuda a que nuestro programa no se trabe de tal maner tener un turno de dado para cada jugador.

```
//Funcion para mover los objetos
function moverObjetos():void{
    //Empezamos con una condicion la cual nos indica que si la longitud de nuestro arreglo la cual contiene los nombre es igual a 2 y turno vale 3
    //Ponemos turno en 1 ya que esto nos indica que solo hay dos jugadores y no entrara en turno 2
    if(nombre.length == 2 && turno == 3){
        //Ponemos el valor de turno en 1
        turno = 1;
    //Si la condicion anterior no se cumple avalua la siguiente condicion en la cual ponemos que si solo hay 3 nombres y sigue el turno 4
    //Pasamos turno a 1
    } else if(nombre.length ==  3 && turno == 4){
        //Ponemos el valor de turno en 1
        turno = 1;
    }
    //Si turno vale 1 quiere decir que sigue el jugador 1
    if(turno == 1){
        tiempoN.start();
        tiempoN.addEventListener(TimerEvent.TIMER, moverEnPos1);
    //Si no se cumple la condicion de arriba quiere decir que es el turno del siguiente jugador
    } else if(turno == 2){
        tiempoN2.start();
        tiempoN2.addEventListener(TimerEvent.TIMER, moverEnPos2);
    } else if(turno == 3){
        tiempoN3.start();
        tiempoN3.addEventListener(TimerEvent.TIMER, moverEnPos3);
    }
    turno += 1;
}
```

En el siguiente código elaboramos una función que nos ayuda a el movimiento de nuestros avatares, condicionando cuantos son los jugadores disponibles en esta partida.

También mostramos las variables de tiempo y las colocamos a cada jugador dependiendo de cuantos jugadores estén, las cuales serán utilizadas para mover el jugador indicado y para otras funciones.

```
//Variables para mover al jugador 1 cada que caiga en una casilla de DK
//Variable para incrementarla en el timer
var contDK1:int;
//Timer que sucede cada 250 milisegundos
var tiempoDK1:Timer = new Timer(250, contDK1++);
//Funcion para mover al usuario dependiendo 10 casillas si cae en una casilla de DK
function tiempoDonkeyK1(event:TimerEvent):void{
    //Si nuestro auxiliar auxPos es menor a 10 entonces se realiza lo siguiente
    if(auxPos < 10){
        //Ponemos el dado invisible para que el siguiente usuario no pueda presionarlo
        dado_btn.visible = false;
        //Aumentamos auxPos en 1
        auxPos += 1;
        //Aumentamos posSiguiente1 en 1 para indicar la siguiente posicion que va a tomar el jugador 1
        posiciones[0] += 1;
        //Le ponemos los tweens en X y en Y a la casilla que se encuentre en la posicion posSiguiente1 en el arreglo
        var jugador11x:Tween = new Tween(jugador11, "x", None.easeIn, jugador11.x, casillas[posiciones[0]].x + 10, 0.25, true);
        var jugador11y:Tween = new Tween(jugador11, "y", None.easeIn, jugador11.y, casillas[posiciones[0]].y + 10, 0.25, true);
    //En el momento en el que se deje de cumplir la condicion reliza lo siguiente
    } else {
        //Ponemos a auxPos en 0
        auxPos = 0;
        //Detenemos el timer
        tiempoDK1.stop();
        //Ponemos el dado invisible para que el siguiente usuario pueda presionarlo
        dado_btn.visible = true;
    }
}

//Variables para mover el jugador 2 cuando caiga en una casilla de DK
//Variable para incrementar en el timer
```

En esta imagen tenemos //Tiempo de Donkey// esta función nos evalúa que nuestro avatar haya caído en cualquiera de las posiciones correpondientes a Donkey Kong el cual comentábamos en las instrucciones que nos dejaba en el espacio DK mas cercano.

```
//Funcion la cual mandamos llamar para saber si el usuario numero 1 cayo en una casilla de DK
function donkeyK1():void{
    //Si cayo en alguna casilla de DK activamos una bandera
    if(posiciones[0] == 2 || posiciones[0] == 12 || posiciones[0] == 22 || posiciones[0] == 32 || posiciones[0] == 42
        || posiciones[0] == 52 || posiciones[0] == 62 || posiciones[0] == 72 || posiciones[0] == 82){
        bandDK = true;
    //En caso de que no se cumpla niguna de las posibilidades de arriba entonces la bandera se desactiva
    } else {
        bandDK = false;
    }
    //Si la bandera esta activa mandamos llamar al timer de DK1 donde estan todos los tweens para mover al avatar
    if(bandDK == true){
        tiempoDK1.start();
        tiempoDK1.addEventListener(TimerEvent.TIMER, tiempoDonkeyK1);
    }
}
```

En esta función evaluamos lo de arriba que es el caer en las posiciones referentes a la de DK, por medio de una condición dejamos predeterminadas que espacios son los que nos caerán.

```
//Funcion la cual mandamos llamar para saber si el usuario numero 1 cayo en una casilla de Bowser
function bowser1():void{
    //Si cayo en alguna casilla de Bowser activamos una bandera
    if(posiciones[0] == 4 || posiciones[0] == 14 || posiciones[0] == 24 || posiciones[0] == 34 || posiciones[0] == 44
        || posiciones[0] == 54 || posiciones[0] == 64 || posiciones[0] == 74 || posiciones[0] == 84 || posiciones[0] == 94){
        bandBow = true;
        //Y le asignamos a la variable kk lo que tengamos en el arreglo
        kk = posiciones[0];
    //En caso de que no se cumpla niguna de las posibilidades de arriba entonces la bandera se desactiva
    } else {
        bandBow = false;
    }
    //Si la bandera esta activa mandamos llamar al timer de B1 donde estan todos los tweens para mover al avatar
    if(bandBow == true){
        tiempoB1.start()
        tiempoB1.addEventListener(TimerEvent.TIMER, tiempoBowser1);
    }
}
```

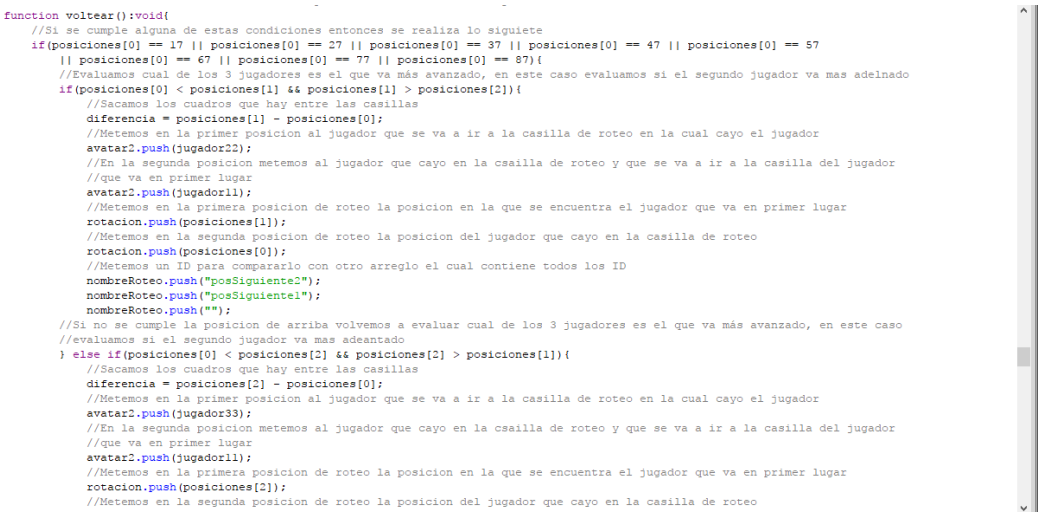
Esta función es una condición de cuando el usuario caeiga en cualquier espacio de que tenga la cara de Bowser se regresará 10 casillas.

```
function tiempoBowser3(event:TimerEvent):void{
    //Empezamos evaluando el ultimo bowser el cual te manda a la primera casilla del tablero, ponemos en la condición que si
    //auxPos es menor a 94 y la variable kk es igual a 94 entonces empiece con estas acciones
    if(auxPos < 94 && kk == 94){
        //Ponemos el dado invisible para que el siguiente usuario no pueda presionarlo
        dado_btn.visible = false;
        //Aumentamos la variable auxPos para que cuando sea igual a randomDado se deje de cumplir la condcion
        auxPos += 1;
        //Decrementamos el arreglo de posiciones en la posicion 2 en 1
        posiciones[2] -= 1;
        //Realizamos los tweens con el nuevo valor del arreglo
        var jugador33x:Tween = new Tween(jugador33, "x", None.easeIn, jugador33.x, casillas[posiciones[2]].x + 10, 0.25, true);
        var jugador33y:Tween = new Tween(jugador33, "y", None.easeIn, jugador33.y, casillas[posiciones[2]].y + 10, 0.25, true);
    //Si no se cumple la condicion de arriba se realiza lo siguiente
    } else if(auxPos < 10 && posSiguiente3 < 92){
        //Ponemos el dado invisible para que el siguiente usuario no pueda presionarlo
        dado_btn.visible = false;
        //Aumentamos auxPos en 1 para que llegue a ser igual a randomDado y asi se deje de cumplir la condicion
        auxPos += 1;
        //Decrementamos el arreglo de posiciones en la posicion 2 en 1
        posiciones[2]-= 1;
        //Realizamos los tweens con el nuevo valor del arreglo
        var jugador33x:Tween = new Tween(jugador33, "x", None.easeIn, jugador33.x, casillas[posiciones[2]].x + 10, 0.25, true);
        var jugador33y:Tween = new Tween(jugador33, "y", None.easeIn, jugador33.y, casillas[posiciones[2]].y + 10, 0.25, true);
        //Si el valor del arreglo en la posicion 1 es igual a 0 quiere decir que caimos en la primera casilla de bowser por lo cual
        //restablecemos los valores
        if(posiciones[2] == 0){
            auxPos = 0;
            tiempoB3.stop();
            //Ponemos el dado invisible para que el siguiente usuario no pueda presionarlo
            dado_btn.visible = true;
        }
    //Cuando se deje de cumplir una de las dos condiciones de arriba restablecemos los valores de todas nuestras variables y detenemos
    //el timer
```

Funcion //Tiempo de Bowser// nos evalua lo anteriormente comentado, si cae que en cualquier condición realizará el decremento. Si cae en la ultima posición de Bowser cerca del final, nos llevará al primer Bowser cerca del inicio.

```
//Variables del timer para mover al usuario 1 por todas las casillas
var contposN:int;
//Timer el cual sucede cada 250 milisegundos
var tiempoN:Timer = new Timer(250, contposN++);
//Funcion que se va a realizar cada 250 milisegundos
function moverEnPos1(event:TimerEvent):void{
    //Condicion en la cual evaluamos que auxPos sea menor a randomDado
    if(auxPos < randomDado){
        //Aumentamos el arreglo en la posicion 0 en 1
        posiciones[0] += 1;
        //Aumentamos auxPos en 1 para que cuando sea igual a randomDado se deje de cumplir
        auxPos += 1;
        //Metemos los tweens para mover el avatar en el nuevo valor de posiciones en la posicion 0
        var jugador11x:Tween = new Tween(jugador11, "x", None.easeIn, jugador11.x, casillas[posiciones[0]].x + 10, 0.25, true);
        var jugador11y:Tween = new Tween(jugador11, "y", None.easeIn, jugador11.y, casillas[posiciones[0]].y + 10, 0.25, true);
    //Cuando se deje de cumplir la condicion de arriba mandamos llamar a las funciones donde tenemos las condiciones para
    //saber si el usuario cayo en alguna casilla especial
    } else {
        //Restablecemos el valor de auxPos
        auxPos = 0;
        donkeyK1();
        bowser1();
        voltear();
        //Detenemos el timer
        tiempoN.stop();
        //Ponemos el dado invisible para que el siguiente usuario pueda presionarlo
        dado_btn.visible = true;
    }
}
```

Esta función nos apoya en el movimiento de cada uno de los usuarios, haciendo que nos guarde cada futuro espacio que pueda caer. Esta función es la base de todo el juego, ya que también aplica el movimiento dependiendo del tiempo y el espacio.



```
function voltear():void{
    //Si se cumple alguna de estas condiciones entonces se realiza lo siguiete
    if(posiciones[0] == 17 || posiciones[0] == 27 || posiciones[0] == 37 || posiciones[0] == 47 || posiciones[0] == 57
        || posiciones[0] == 67 || posiciones[0] == 77 || posiciones[0] == 87){
        //Evaluamos cual de los 3 jugadores es el que va más avanzado, en este caso evaluamos si el segundo jugador va mas adelnado
        if(posiciones[0] < posiciones[1] && posiciones[1] > posiciones[2]){
            //Sacamos los cuadros que hay entre las casillas
            diferencia = posiciones[1] - posiciones[0];
            //Metemos en la primer posicion al jugador que se va a ir a la casilla de roteo en la cual cayo el jugador
            avatar2.push(jugador22);
            //En la segunda posicion metemos al jugador que cayo en la csailla de roteo y que se va a ir a la casilla del jugador
            //que va en primer lugar
            avatar2.push(jugador11);
            //Metemos en la primera posicion de roteo la posicion en la que se encuentra el jugador que va en primer lugar
            rotacion.push(posiciones[1]);
            //Metemos en la segunda posicion de roteo la posicion del jugador que cayo en la casilla de roteo
            rotacion.push(posiciones[0]);
            //Metemos un ID para compararlo con otro arreglo el cual contiene todos los ID
            nombreRoteo.push("posSiguiente2");
            nombreRoteo.push("posSiguiente1");
            nombreRoteo.push("");
        //Si no se cumple la posicion de arriba volvemos a evaluar cual de los 3 jugadores es el que va más avanzado, en este caso
        //evaluamos si el segundo jugador va mas adeantado
        } else if(posiciones[0] < posiciones[2] && posiciones[2] > posiciones[1]){
            //Sacamos los cuadros que hay entre las casillas
            diferencia = posiciones[2] - posiciones[0];
            //Metemos en la primer posicion al jugador que se va a ir a la casilla de roteo en la cual cayo el jugador
            avatar2.push(jugador33);
            //En la segunda posicion metemos al jugador que cayo en la csailla de roteo y que se va a ir a la casilla del jugador
            //que va en primer lugar
            avatar2.push(jugador11);
            //Metemos en la primera posicion de roteo la posicion en la que se encuentra el jugador que va en primer lugar
            rotacion.push(posiciones[2]);
            //Metemos en la segunda posicion de roteo la posicion del jugador que cayo en la casilla de roteo
```

Función voltear es un espacio diferente a los demás, este espacio al momento de caer en las casillas correspondientes nos llevan a poder cambiar el lugar con el que va más delante de nosotros. En caso de que esto no se cumpliera, solo se quedaría en la misma posición.

```
    } else {
        //Si posiciones[0] es igual a 99 quiere decir que ganamos
        if(posiciones[0] == 99){
            //Detenemos los timer
            tiempoN.stop();
            timer.stop();
            //Nos pasamos al siguiente frame
            gotoAndStop(6);
            //Muestra el avatar del ganador y el nombre del ganador
            ganador1.visible = true;
            ganador3.visible = false;
            ganador2.visible = false;
            ganador200.text = "" + nombre[0];
        }
        //Restablecemos el valor de auxPos
        regresoRebote = 0;
        auxPos = 0;
        donkeyK1();
        bowser1();
        voltear();
        //Detenemos el timer
        tiempoN.stop();
        //Ponemos el dado invisible para que el siguiente usuario pueda presionarlo
        dado_btn.visible = true;
    }
}

//Funcion para revisar si hay un rebote, si el valor de posicion[i] es igual a 99 lo guardamos para despues compararlo
function rebote():void{
    if(posiciones[0] == 99){
        regresoRebote = posiciones[0];
    }
}

//Funcion para revisar si hay un rebote, si el valor de posicion[i] es igual a 99 lo guardamos para despues compararlo
```

El primer if de este código menciona que cuando el primer usuario caiga en 99 quiere decir que ha ganado y que ha caído número exacto, y lo mandamos a otro frame para indicar quien es el ganador.

Cada una de estas funciones aplica para los tres usuarios, en caso de que se jugaran menos de tres (dos jugadores), sería de la misma manera. En estos caso siempre tendríamos que tener un rival con quien jugar, ya que serpientes y escaleras es un juego de azar cuyo objetivo es jugar con más de dos jugadores.

FRAME GANADOR

Mostramos el avatar ganador y el nombre del jugador ganador, y lo exportamos a un PDF.

CONCLUSION

El proyecto que realizamos ha contribuido de manera muy importante para identificar y resaltar los puntos que hay que cubrir y considerar para llevar a cabo una implementación exitosa de los sistemas de información.

Nos deja muchas cosas importantes que reflexionar y muchas otras las ha reforzado como puntos angulares para llevar a cabo una buena implementación. Dentro de los puntos que consideramos tienen más importancia dentro de un proyecto de esta naturaleza son el detectar cuáles son las necesidades reales de las personas que trabajan día a día con los sistemas, que los procesos operativos de una empresa se apeguen a la realidad del trabajo diario y no sean un obstáculo burocrático, que se involucre a los usuarios en el proceso de implementación de los sistemas de manera que se sepa que es lo que ellos esperan y qué es lo que no esperan de él, definir de manera clara y lo más tangible posible los beneficios económicos, laborales, y de cualquier otra índole que se piensan alcanzar con los sistemas nuevos, de manera que las personas dentro de la empresa sepan cómo se van a ver beneficiados particularmente.