Manual técnico

Tetris

Herramientas Multimedia

16/04/2018

Autores:

Damarys Mendoza Vázquez (1730317)

Kate Aracely Rodríguez Estrada (170335)

MANUAL TÉCNICO

Tabla de contenido

Introducción

En esta actividad se logró diseñar e implementar el juego de Tetris con el programa de flash en el lenguaje de actionscript 3.0 con el objetivo de poder jugar, en una manera de cliente a servidor y en la cual se evalué el resultado entre ambos dando como resultado aquel con mayor puntaje.

Funcionalidad

- El enfoque que se le da al juego es atractivo en el entorno visual, y también se logra ser multijugador, con la opción de que los usuarios participen en un mismo momento, de tal manera que es un juego competitivo que su funcionalidad es entretener a los usuario y logren completar juntar más puntos

Desarrollo (Código)

```
package  {
    //Librerias a importar que se necesitaran en la clase
    import flash.display.Sprite;
    import flash.utils.Timer;
    import flash.events.TimerEvent;
    import flash.events.KeyboardEvent;
    import fl.transitions.Tween;
    import fl.transitions.easing.*;

    public class EstaSi extends Sprite{
        public function EstaSi() {
            //Tweens para la portada
        var tetris:Tween = new Tween (portada_mc.tetris_mc,"x",Bounce.easeOut,-423.85,191.9,3,true);
        var tetris1:Tween = new Tween (portada_mc.tetris_mc,"y",Bounce.easeOut,-182.85,21.1,3,true);
        var tetris2:Tween = new Tween (portada_mc.cargando,"y",None.easeIn,586.7,258.85,2,true);
        var cuadro:Tween = new Tween (portada_mc.figura,"y",None.easeIn,-164,617.2,3,true);
        var cuadro2:Tween = new Tween (portada_mc.figura2,"y",None.easeIn,-140.75,617.2,3,true);
        var cuadro4:Tween = new Tween (portada_mc.figura3,"y",None.easeIn,-136.8,610.15,3,true);
        var cuadro5:Tween = new Tween (portada_mc.figura4,"y",None.easeIn,-136.8,618.15,3,true);
        var katet:Tween = new Tween (portada_mc.kate,"x",Bounce.easeOut,500,142.15,3,true);
        var damaM:Tween = new Tween (portada_mc.dama,"x",Bounce.easeOut,-300.85,196.8,3,true);
        //declaracion de variables
        var escenario: uint = 29.8;//la medida de expansion de nuestro escenario
        var eArray:Array;//Array del escenario
        var eSprite:Sprite;//Sprite de nuestro escenario
        var Figuras:Array= new Array;//array que contendra las opciones de figura y los diferentes cas
        var colores: Array= new Array;
        var fig:Sprite;//sprite que contendra nuestra figura realizada
```

Se realiza la importación de las librerías que se ocuparan en el lapso del código. Empezamos con los tweens de nuestra portada y la declaración de las variables que utilizaremos en nuestro escenario, de igual manera los arrays que nos servirán para las figuras.

Se continúa con la declaración de las variables y tenemos dos timers, el primero es el que se muestra en el juego y el segundo es el tiempo que tardara en ingresar al juego

```
var actual: uint;// es la figura actual
var siguienteF:uint;//variable para la siguiente figura
var RotacionA:uint; //variable que nos auxiliara para realizar la rotacion
var filas:int;// para las filas
var columnas:int;//para las columnas
var Tiempo:Timer = new Timer (500);//cinco milisegundos que tardara en caer la figura
var Juego:Boolean=false;//variable para determinar si el juego a terminado
var pun_txt:String;
var puntos:int=0;//variable que se le asignara el puntuaje

//tiempo del juego
        var tmpp:int=0;
        var cont01:int;
var timerr:Timer=new Timer(1000,cont01++);

    //tiempo para entrar a la portada
    var tmp:int=0;
    var cont1:int;
    var timer:Timer=new Timer(1000,cont1++);
    timer.start();
    timer.addEventListener(TimerEvent.TIMER,tiempo);

    function tiempo(tiempoevent: TimerEvent) {
        var nombre:String;
```

Se implementa un trace para verificar el tiempo de carga, y se pone la condición para nuestra caja de texto. Y se da la impresión en el fotograma del tiempo de juego.

```
trace("Espera mientras carga"+"  " +tmp);
tmp++;
if (tmp > 8) { //si pasa un lapso de tiempo
    portada_mc.visible=true; //el movieclip de portada se mantendra en true
    tmp++; //se incrementa
    if (tmp > 4 && nombre_txt.text.length>0 ) {
        nombre=String(nombre_txt.text);
        timer.stop(); //se detiene el tiempo
        portada_mc.visible=false; //se quita el movieclip
        nombrejugador_txt.text = nombre;
        nombre_txt.visible=false;
        namestatico_txt.visible = false;
    }

var minutos:int; //minutos del
timerr.start();
timerr.addEventListener(TimerEvent.TIMER,tiemp);
function tiemp(e:TimerEvent):void{
    tmpp++; // tmp son segundos y se van incrementando
    if(tmpp > 59){ //cuando tiempo sea mayor a 59 osea 60 segundos
        minutos++;//se incrementan los minutos en 1
        tmpp = 0; //se reinician los segundos
    }
    tiempo_txt.text = minutos + " min." + " " + tmpp + " seg."; // se imprimen minutos con

            }//realizamos la llamada de nuestras funciones a utilizar
```

Se comienza con el llamado de algunas funciones que se explicaran un poco más adelante, la función de GenerarEs nos sirve para poder generar nuestra cuadricula o escenario donde caerán nuestras figuras, primero se le asigna los colores que portara, tenemos la declaración tanto del Sprite y Array del mismo, y con dos for recorremos nuestras filas y columnas.

```
                }//realizamos la llamada de nuestras funciones a utilizar
                JuegoTerminado();
                GenerarEs();
                MovFigurass();
                siguienteF = Math.floor(Math.random()*7);
                GenerarF();
                stage.addEventListener(KeyboardEvent.KEY_DOWN, Teclas);

        }
        //funcion para generar el escenario
    function GenerarEs():void{
        var colors:Array=new Array ("0x1a5276", "0x58d68d");//los colores que se le asignaran al escenario
        eArray=new Array;//Array del escenario
        var eSprite:Sprite = new Sprite;//sprite del escenario
        eSprite = new Sprite;
        addChild(eSprite);//se agrega el sprite que contiene el escenario
        for (var i:uint = 0; i<20; i++){//con un for verificamos nuestras  filas
            eArray[i] = new Array;
            for (var j: uint=0; j<10; j++){//con un for verificamos nuestras  columnas
                eArray[i][j]=0;
                eSprite.graphics.beginFill(colors[(j + i)%2]); //se agregan las filas y las columnas y se div
                                                               //entre dos para que haga el brinco entre fil
                eSprite.graphics.drawRect(escenario *j, escenario *i, escenario, escenario);
            }

        }
    }
```

Declaramos nuestra función que nos permitirá la rotación o cambio de estado de una figura, como bien se sabe en tetris se tiene siete figuras originales y base a continuación se visualiza las matrices con los estados que pueden tener.

```
    function MovFigurass():void{
        Figuras[0]= [
        [//matriz para la forma "L"
         //aqui solo son necesaria dos formas pero para ser posible que se pueden cambiar al mov de las tecla
         //es necesario que sean cuatro por la razon de que otras figuras si tienen cuatro cambios posibles
            [0,0,0,0],
            [1,1,1,1],
            [0,0,0,0],
            [0,0,0,0],
            ],
        [
            [0,1,0,0],
            [0,1,0,0],
            [0,1,0,0],
            [0,1,0,0],
        ],
        [
            [0,0,0,0],
            [1,1,1,1],
            [0,0,0,0],
            [0,0,0,0],
        ],
        [
            [0,1,0,0],
            [0,1,0,0],
            [0,1,0,0],
            [0,1,0,0],
            ]
```

En este primer ejemplo de la figura “L” solo tiene dos estados, sin embargo son necesarias cuatro matrices porque es el maximo de cambios en todas las figuras y mas adelante se evaluara en la tecla correspondiente.

Se le asigna un color hexadecimal que tomara la posición número cero del array de colores.

```
colores[0]=0x00FFFF;
Figuras[1] = [
[
        [0,0,0,0],
        [1,1,1,0],
        [0,1,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,0,0],
        [1,1,0,0],
        [0,1,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,0,0],
        [1,1,1,0],
        [0,0,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,0,0],
        [0,1,1,0],
        [0,1,0,0],
        [0,0,0,0],
    ]
];
colores[1]=0xAA00FF;
```

Tercera figura original

```
colores[1]=0xAA00FF;
Figuras[2] = [
[

        [0,0,0,0],
        [1,1,1,0],
        [1,0,0,0],
        [0,0,0,0],
    ],
     [
        [1,1,0,0],
        [0,1,0,0],
        [0,1,0,0],
        [0,0,0,0],
    ],
    [
        [0,0,1,0],
        [1,1,1,0],
        [0,0,0,0],
        [0,0,0,0],
    ],
      [
        [0,1,0,0],
        [0,1,0,0],
        [0,1,1,0],
        [0,0,0,0],
    ]
];
```

Cuarta figura original

```
,,
colores[2] = 0xFFA500;
Figuras[3]= [
    [

        [1,0,0,0],
        [1,1,1,0],
        [0,0,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,1,0],
        [0,1,0,0],
        [0,1,0,0],
        [0,0,0,0],
    ],
    [
        [0,0,0,0],
        [1,1,1,0],
        [0,0,1,0],
        [0,0,0,0],
    ],
    [
        [0,1,0,0],
        [0,1,0,0],
        [1,1,0,0],
        [0,0,0,0],
    ]
];
```

Quinta figura original

```
colores[3] =0x0000FF;
Figuras[4]=[
[

        [0,0,0,0],
        [1,1,0,0],
        [0,1,1,0],
        [0,0,0,0],
    ],
    [
        [0,0,1,0],
        [0,1,1,0],
        [0,1,0,0],
        [0,0,0,0],
    ],
    [
        [0,0,0,0],
        [1,1,0,0],
        [0,1,1,0],
        [0,0,0,0],
    ],
    [
        [0,0,1,0],
        [0,1,1,0],
        [0,1,0,0],
        [0,0,0,0],
    ]
];
```

Sexta figura original

```
colores[4] = 0xFF0000;
Figuras[5]=[
[
        [0,0,0,0],
        [0,1,1,0],
        [1,1,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,0,0],
        [0,1,1,0],
        [0,0,1,0],
        [0,0,0,0],
    ],
    [
        [0,0,0,0],
        [0,1,1,0],
        [1,1,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,0,0],
        [0,1,1,0],
        [0,0,1,0],
        [0,0,0,0],
    ]
];
colores[5]=0x00FF00;
```

Séptima figura original

```
colores[5]=0x00FF00;
Figuras[6]= [
[
        [0,1,1,0],
        [0,1,1,0],
        [0,0,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,1,0],
        [0,1,1,0],
        [0,0,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,1,0],
        [0,1,1,0],
        [0,0,0,0],
        [0,0,0,0],
    ],
    [
        [0,1,1,0],
        [0,1,1,0],
        [0,0,0,0],
        [0,0,0,0],

    ]
];
```

La función GenerarF nos permitirá crear nuestras figuras, pero esta función está ligada a otras dos primero se incluye una condición con la variable de Juego de ser así se realiza un random por siete (el número de figuras originales) y haremos el llamado de la función que muestra la siguiente figura en el fotograma

```
    //Generar la figura
  function GenerarF():void{
     if(!Juego){//si nuestra variable tipo boleana es verdadera entonces se hace lo siguiente
         actual = siguienteF;// la figura que ahora se va generar es la que se mostraba en el
         //recuadro de la siguiente figura
         siguienteF= Math.floor(Math.random()*7);//con un random se elige la figura
         Siguiente();//de igual manera mandamos llamar nuestra funcion que muestra la siguiente figura
         RotacionA = 0;//se posiciona en medio
         filas = 0;
         if( Figuras[actual][0][0].indexOf(1)==-1){
             filas=-1;
         }
         columnas = 3;//en donde cae nuestra figura en el escenario
         Dibujar();//manda llamar la funcion de dibujar a la figura
         if (Posicionar(filas, columnas, RotacionA)){//si la figura esta bien posicionada, entonces...
             Tiempo.addEventListener(TimerEvent.TIMER, Tcaida);//se tiene un Listener del tiempo, con la fu
             Tiempo.start();
         }else{
             Juego = true;
         }
     }
 }
```

Con esta función ya tenemos en un contenedor la fig que se muestra en el escenario y con la función lugar, limitamos sus movimientos o su caída dentro del perímetro de nuestro escenario, predispuesto.

```
 function Dibujar():void{//Funcion que dibujara nuestra figura
     var fi:uint= actual;//la variable fi toma el valor de la figura actual
     fig= new Sprite;
     addChild (fig);
     for (var i:int = 0; i< Figuras[fi][RotacionA].length; i++){//con el for tomamos en cuentas el array
                                                 //las figuras y la variable que acaba de tomar el valor de 1
     for(var j: int = 0; j< Figuras[fi][RotacionA][i].length; j++) {
         if (Figuras[fi][RotacionA][i][j]==1){
             fig.graphics.beginFill(colores[fi]);//se le asigna su debido color a la fig
             fig.graphics.drawRect(escenario*j, escenario*i, escenario, escenario);

             }
         }
     }
     Lugar();//llamado de la funcion de lugar
} //con esta funcion tendremos en cuenta el limite de nuestra figura para poder caer
function Lugar():void{
     fig.x= columnas*29.8;// la posicion en x y y sera igual a la multiplicacion de las filas y columnas
     fig.y= filas*29.8;//respectivamente por la medida de nuestro escenario
```

En esta función hacemos un switch que tendrá las opciones de teclas que el usuario pueda oprimir y lo que estas realizaran. Por ejemplo la tecla número treinta y siete, nos permite deslizarnos hacia la izquierda, la treinta y ocho el cambio de estado en la figura, como se muestra si nuestra variable cambio es igual a cuatro el máximo de matrices por figura, se reinicia de tal manera que el usuario pueda seguir cambiando su estado

```
}//funcion que permitira el movimiento de nuestra figura
 function Teclas (e:KeyboardEvent):void{
    if(!Juego){//si la variable del juego esta en verdadero
        switch (e.keyCode){
            case 37://en caso que le oprima a la tecla  37 significa
            // que la figura tendra un movimiento asi la izquiera en el escenario
            if(Posicionar(filas, columnas-1, RotacionA)){
                columnas--;//por lo cual se realiza un decremento en la variable de las columnas
                Lugar(); //y es necesario el llamado de la funcion de lugar para evaluar que no este limit
            }
            break;
            case 38://en caso que le oprima a la tecla 38 significa que el usuario
            //decidio cambiar el estado de la figura original
            var cambio:uint=RotacionA + 1 %Figuras[actual].length;
            if(cambio==4){//si nuestra variable cambio es igual a 4 que son las matrices que tenemos por c
                cambio=0;// se reinicia de tal manera que sea posible seguir cambiando entre los cuatro es
            }
            if (Posicionar(filas, columnas, cambio)){
                RotacionA=cambio;
                removeChild(fig);
                Dibujar();//hacemos llamado de nuestro funcion dibujar
                Lugar();// y tambien de Lugar
            }
            break;
```

En caso de la tecla treinta y nueva nos permite deslizarnos en sentido de la derecha. Y la cuarenta un aumento en la caída de la figura.

```
            case 39://en caso de que oprima la tecla 39  significa que la figura se movera hacia la derec
            if (Posicionar(filas, columnas + 1, RotacionA)){
                columnas++; //incrementamos la variable de columnas para que se realice el movimiento
                Lugar();//y mandamos llamar la funcion de lugar para la limitacion de la cuadricula

            }
            break;
        case 40://con la tecla 40 es para deslizar con mayor velocidad
            if( Posicionar(filas+1, columnas, RotacionA)){
                filas++;//es por eso que se aumenta el valor de filas
                Lugar();
            } else{

                Caer();
                GenerarF();
            }
            break;

        }
    }
}
```

```
    function Posicionar(fil:int, col:int, lad:uint): Boolean{//declaracion de variables que se
       var fi:uint = actual;//utilizaran dentro de esta funcion
       for(var i:int= 0; i<Figuras[fi][lad].length; i++){
           for (var j:int=0; j < Figuras[fi][lad][i].length; j++){
           if (Figuras[fi][lad][i][j]==1){
               if (col+j<0){
                   return false;
               }
               if (col+j>9){
                   return false;
               }
               if (fil + i >19){
                   return false;
               }
               if (eArray[fil+i][col+j]==1){
                   return false;
               }
           }
       }
    }
    return true;
}
```

Se detiene el tiempo que se tiene predispuesto para la caída de las figuras

```
//funcion para la caida de nuestra figuras
  function Caer():void{
    var fi:uint= actual;//tomara el valor de la figura que se encuentra actual
    var aterriz:Sprite;//se declara el Sprite
    for (var i:int =0; i<Figuras [fi][RotacionA].length; i++){//con dos for evaluaremos
        for (var j: int=0; j< Figuras[fi][RotacionA][i].length; j++){//el array de las figuras la figura ac
            if (Figuras[fi][RotacionA][i][j]==1){
                aterriz=new Sprite;
                addChild(aterriz);
                aterriz.graphics.beginFill(colores[actual]);//se asigna su valor correspondiente
                aterriz.graphics.drawRect(escenario * (columnas+j), escenario * (filas+i), escenario, escen
                aterriz.name="r"+ (filas+ i)+ "c" + (columnas+j);
                eArray[filas + i][columnas+j]=1;
            }


        }
    }
    removeChild (fig);
    Tiempo.removeEventListener (TimerEvent.TIMER, Tcaida);
    Tiempo.stop();//se detiene el tiempo de caida
    Colision();
}
```

Colisión.- Evaluamos si se tiene que realizar la eliminación de una columna en caso de que todos sus espacios se encuentren completos y es entonces donde se sumara a la variable de puntos cincuenta, y se imprimirá en su respectiva caja de texto.

```
//verifica que no este
  function Colision():void{
    for(var i: int=0; i<20; i++){//con este for se evalua
        if (eArray[i].indexOf(0)==-1){//si se elimino una fila
            puntos=puntos+50;// la variable puntos se le sumara 50 por cada una
            trace(puntos);
            puntos_txt.text=String(puntos);//se imprimira en la caja de texto respectivamente
            for(var j:int=0; j<10; j++){
                eArray[i][j]=0;
                removeChild(getChildByName("r"+ i +"c" + j));
            }
            for (j=i; j>=0; j--){//se decrementa la variable j
                for (var u:int =0; u<10; u++){
                    if(eArray[j][u]==1){
                        eArray[j][u]=0;
                        eArray[j+1][u]= 1;
                        getChildByName("r" + j + "c" + u).y += escenario;
                        getChildByName("r" + j + "c" + u).name= "r" + (j + 1)+ "c" + u;

                    }

                }
            }
        }
    }
}
```

En esta imagen tenemos dos funciones una de ellas es para mostrar en el escenario la siguiente figura que caerá y su color respectivo.

```
 function Tcaida(e:TimerEvent):void{
    if(Posicionar (filas + 1, columnas, RotacionA)){
        filas++;
        Lugar();

    } else{
        Caer();
        GenerarF();
    }
}
 function Siguiente():void{
    if(getChildByName("next") != null){
        //pun_txt=("juego terminado");
        //trace ("paro");
        removeChild(getChildByName("next"));
    }
    var SIGF: Sprite= new Sprite;
    SIGF.x=345; //posicion que tomara nuestro sprite
    SIGF.y=258.6;
    SIGF.name="next";
    addChild (SIGF);
    for(var i: int=0; i< Figuras[siguienteF][0].length; i++){
        for (var j:int=0; j< Figuras[siguienteF][0][i].length; j++){
            if (Figuras[siguienteF][0][i][j]==1){
                SIGF.graphics.beginFill(colores[siguienteF]);
                SIGF.graphics.drawRect(escenario * j, escenario* i, escenario, escenario);
                    }
                }
```

Conclusión

En la elaboración de este manual técnico se logró desarrollar ampliamente la explicación sobre cómo se encuentran cada una de las opciones en nuestro proyecto del juego Tetris, la visualización, el diseño y la temática que se le proporciono todo esto ligado al cumpliendo de las especificaciones solicitadas por el docente, en tiempo y forma.