JUEGO TETRIS

MANUAL TÉCNICO

ITI 2-2

HERRAMIENTAS MULTIMEDIA

MSI MARIO HUMBERTO RODRIGUEZ CHAVEZ

JESUS ALFREDO ANDRIO RODRIGUEZ
BERENICE LIZETH DE LEON AGUILERA

ABRIL 2018

ÍNDICE

INTRODUCCION:

En este manual se describirán los pasos que se realizaron para concluir con la actividad del proyecto del juego llamado tetris.

Primero se describirán los pasos de el archivo donde están todos los fotogramas y las acciones de estos mismos, después de explicara lo que se codifico en la clase denominada “tetris”, y por ultimo se explicara la codificación de la segunda clase llamada “objeto”.

Acciones fotograma 1

```
stop();
import flash.events.MouseEvent;
import fl.transitions.Tween;//Liberias para
import fl.transitions.easing.*;
import fl.transitions.TweenEvent;
import flash.events.Event;
bp.addEventListener(MouseEvent.CLICK, fbp);
```

Se muestran las galerías necesarias para realizar las transiciones, un evento del ratón y un botón.

```
var letraT: Tween= new Tween(T,"y",Elastic.easeInOut,-100,25,5,true);
var letrae: Tween= new Tween(e,"y",Elastic.easeOut,-100,25,5,true);
var letrat: Tween= new Tween(t,"y",Elastic.easeInOut,-100,25,5,true);
var letrar: Tween= new Tween(r,"y",Elastic.easeOut,-100,25,5,true);
var letrai: Tween= new Tween(i,"y",Elastic.easeInOut,-100,25,5,true);
var letras: Tween= new Tween(s,"y",Elastic.easeOut,-100,25,5,true);
```

Estas son las variables de cada letra del título de la pantalla principal, que aparecen con sus respectivos tweens al momento de ejecutar la película.

Acciones fotograma 2

```
import flash.events.MouseEvent;

br.addEventListener(MouseEvent.CLICK,fbr);
function fbr(event:MouseEvent){
    gotoAndStop(3);
}
bcomo.addEventListener(MouseEvent.CLICK, fcomo);
function fcomo(event:MouseEvent):void{
    gotoAndStop(6);
}
```

Aquí se declara la función de los dos botones para ir al fotograma correspondiente.

Acciones fotograma 3

```
import flash.events.MouseEvent;

bj.addEventListener(MouseEvent.CLICK, fbj);

var vnom:Array= new Array();
var contaj:Number;
bgua.addEventListener(MouseEvent.CLICK, guardar);
function guardar(event:MouseEvent):void{
    vnom.push(intro.text);
    contaj++
    trace(vnom);
    intro.text="";
}
```

Aquí se declara la función de otros dos botones, uno que nos lleva al fotograma siguiente que es donde se encuentra el juego y el otro que cumple la función de almacenar o guardar el nombre del jugador.

Acciones fotograma 4

```
import flash.events.Event;
import flash.events.MouseEvent;
import flash.utils.Timer;
import flash.events.KeyboardEvent;
import flash.events.Event;
import flash.events.TimerEvent;
import fl.transitions.*;
import fl.transitions.easing.*;
import flash.text.engine.EastAsianJustifier;
import flash.events.MouseEvent;
import flash.net.URLLoader;
import flash.net.URLRequest;
import flash.display.Loader;
import flash.net.*;

// contaj:Number;
nombrej_txt.text = vnom[0];
```

Aquí se colocan las librerías necesarias, este es el fotograma donde se encuentra el juego. También aquí se manda llamar al texto de la variable donde se guarda el dato del nombre del jugador.

```
var tmp:Array= new Array();
var tmp1:int=0;
var cont1:int;
var timer1:Timer= new Timer(1000,cont1++);

timer1.start();
timer1.addEventListener(TimerEvent.TIMER,tiempo1)

          function tiempo1(tiempoevent:TimerEvent):void
          {
              tmp1++;
              tiempo1_txt.text= tmp1+" s.";

          }

function salir(e:MouseEvent){
    tmp.push(String(tmp1));
```

Se declaran las variables para el tiempo y el contador del mismo, a la vez que se declara su función. También está declarado un botón para salir del juego.

Acciones fotograma 5

```
stop();
gana.text=String(vnom[0]);
tie.text=String(tmp[0]);
import flash.net.LocalConnection;
import flash.events.MouseEvent;

var enviar:LocalConnection= new LocalConnection();
var enviar1:LocalConnection= new LocalConnection();
com.addEventListener(MouseEvent.CLICK, fcom);
function fcom(event:MouseEvent):void{
    var textol:String= gana.text;
    var texto2:String;
    texto2=String(tie.text);
    enviar.send("Conexion","mensaje",textol);
    enviar.send("Conexion","mensaje1",texto2);
}
```

Aquí se declaran las variables para tomar los datos de las cajas de texto de los acumuladores para mostrar al jugador y el tiempo que obtuvo en el transcurso del juego.

Acciones fotograma 6

```
import flash.events.MouseEvent;

bre.addEventListener(MouseEvent.CLICK, fbre);
function fbre(e:MouseEvent):void{
    gotoAndStop(2);
}
```

En este fotograma solo se declara la función de un botón para regresar a otro fotograma.

Acciones fotograma resultado

```
import flash.net.LocalConnection;

var enviar:LocalConnection= new LocalConnection();
enviar.connect("Conexion");
enviar.client=this;
//enviar1.connect("Conexion1");
//enviar1.client=this;
function mensaje(str:String):void{
    hola.text= str;
}
function mensaje1(str1:String):void{

    hola2.text= str1;
}
```

Las funciones de mensaje mandan los datos a las casillas de resultados y los muestran después en los cuadros de texto llamadas hola y hola2.

Clase tetris

```
package  {

    import flash.display.MovieClip;
    import flash.events.MouseEvent;
    import flash.utils.Timer;
    import flash.events.KeyboardEvent;
    import flash.events.Event;
    import flash.events.TimerEvent;
    import fl.transitions.*;
    import fl.transitions.easing.*;
    import flash.text.engine.EastAsianJustifier;
    import flash.net.Socket;
    import flash.net.LocalConnection;
    import flash.net.SecureSocket;
    import flash.net.NetConnection;
    import flash.net.XMLSocket;
    import flash.ui.Keyboard;
    import flash.net.URLLoader;
    import flash.net.URLRequest;
```

Se importan todas las librerías necesarias a la clase para que funcione correctamente.

```
public class tetris_class extends MovieClip {

    var bla:Socket= new Socket();
    var ser:XMLSocket= new XMLSocket();
    public function tetris_class() {

    }
    public function fbp(event:MouseEvent){
        gotoAndStop(2);
    }

    public function fbj(event:MouseEvent):void{
        gotoAndStop(4);
    }
    public function ftet(event:MouseEvent):void{
    var urll:URLRequest= new URLRequest("test.html");
        navigateToURL(urll, "safe");
    }
```

Aquí se utilizan los servidores sokets para cumplir la petición que se pide y responderla. También están declaradas las funciones de dos botones y se manda llamar el documento en html.

Clase objetos

```
package {
    import flash.display.Sprite;
    import flash.utils.Timer;
    import flash.events.TimerEvent;
    import flash.events.KeyboardEvent;

    public class Objetos extends Sprite{
        private const TS:uint = 24;
        private var fieldArray:Array;
        private var fieldSprite:Sprite;
        private var tetrominoes:Array = new Array();
        private var colors:Array = new Array();
        private var tetromino:Sprite;
        private var currentTetromino:uint;
        private var nextTetromino:uint;
        private var currentRotation:uint;
        private var tRow:int;
        private var tCol:int;
        private var timeCount:Timer = new Timer(500);
        private var gameOver:Boolean = false;
        private var contador:Number;
```

En esta clase se importan las librerías correspondientes que se utilizaran y las variables de manera privada.

```
public function Objetos(){
    generateField();
    initTetrominoes();
    nextTetromino = Math.floor(Math.random() * 8);
    generateTetromino();
    stage.addEventListener(KeyboardEvent.KEY_DOWN, onKDown);
}
private function generateField():void{
    var colors:Array = new Array("0x4444444","0x5555555");
    fieldArray = new Array;
    var fieldSprite:Sprite = new Sprite;
    addChild(fieldSprite);
    fieldSprite.graphics.lineStyle(0, 0x000000);
    for (var i:uint = 0; i < 20; i++){
        fieldArray[i] = new Array;
        for (var j:uint = 0; j < 10; j++){
            fieldArray[i][j] = 0;
            fieldSprite.graphics.beginFill(colors[(j% 2 + i % 2) % 2]);
            fieldSprite.graphics.drawRect(TS * j, TS * i, TS, TS);
            fieldSprite.graphics.endFill();
        }
```

Aquí se declara un math.floor y math.random para las figuras denominadas tetrinomios, para que estas salgan de manera aleatoria en el escenario. También se establecen sus límites, el color, el conteo de las figuras y los gráficos.

```
private function initTetrominoes():void{
    tetrominoes[0] = [
    // I
    [
    [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]
    ]
    ];
    colors[0] = 0x00FFFF;
```

En esta función por medio de un sistema de matrices se establecen las dimensiones que pueden tomar la figura o las casillas que puede utilizar, en este caso se declaran para la figura I.

```
//J
tetrominoes[1] = [
[
[0,0,0,0],
[0,1,1,1],
[0,1,0,0],
[0,0,0,0]
],
[
[0,0,0,0],
[1,1,0,0],
[0,1,0,0],
[0,1,0,0]
],
[
[0,1,0,0],
[0,1,1,1],
[0,0,0,0],
[0,0,0,0]
],
[
[0,0,0,0],
[0,1,1,0],
[0,1,0,0],
[0,1,0,0]
]
];
colors[1] = 0xAA00FF;
```

En esta función por medio de un sistema de matrices se establecen las dimensiones que pueden tomar la figura o las casillas que puede utilizar, en este caso se declaran para la figura J.

```
//L
tetrominoes[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]
]
];
colors[2] = 0xFFA500;
```

En esta función por medio de un sistema de matrices se establecen las dimensiones que pueden tomar la figura o las casillas que puede utilizar, en este caso se declaran para la figura L.

```
// T
tetrominoes[3] = [
[
[0,1,0,0],
[1,1,1,0],
[0,0,0,0],
[0,0,0,0]
],
[
[0,1,0,0],
[1,1,0,0],
[0,1,0,0],
[0,0,0,0]
],
[
[0,0,1,0],
[0,0,1,1],
[0,0,1,0],
[0,0,0,0]
],
[
[0,0,0,0],
[1,1,1,0],
[0,1,0,0],
[0,0,0,0]
]
];
colors[3] = 0x0000FF;
```

En esta función por medio de un sistema de matrices se establecen las dimensiones que pueden tomar la figura o las casillas que puede utilizar, en este caso se declaran para la figura T.

```
// Z
tetrominoes[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]
]
];
colors[4] = 0xFF0000;
```

En esta función por medio de un sistema de matrices se establecen las dimensiones que pueden tomar la figura o las casillas que puede utilizar, en este caso se declaran para la figura Z.

```
// S
tetrominoes[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]
]
];
colors[5] = 0x00FF00;
```

En esta función por medio de un sistema de matrices se establecen las dimensiones que pueden tomar la figura o las casillas que puede utilizar, en este caso se declaran para la figura S.

```
// o
tetrominoes[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]
]
];
colors[6] = 0xFFFF00;
```

En esta función por medio de un sistema de matrices se establecen las dimensiones que pueden tomar la figura o las casillas que puede utilizar, en este caso se declaran para la figura O.

```
//.
tetrominoes[7] = [
[
[0,0,0,0],
[0,0,1,0],
[0,0,0,0],
[0,0,0,0]
],
[
[0,0,0,0],
[0,0,1,0],
[0,0,0,0],
[0,0,0,0]
],
[
[0,0,0,0],
[0,0,1,0],
[0,0,0,0],
[0,0,0,0]
],
[
[0,0,0,0],
[0,0,1,0],
[0,0,0,0],
[0,0,0,0]
]
];
colors[7] = 0xFFAA00;
```

En esta función por medio de un sistema de matrices se establecen las dimensiones que pueden tomar la figura o las casillas que puede utilizar, en este caso se declaran para la figura de un cuadro.

```
private function generateTetromino():void{
    if (!gameOver){
        currentTetromino = nextTetromino;
        nextTetromino = Math.floor(Math.random() * 8);
        drawNext();
        currentRotation = 0;
        tRow = 0;
        if (tetrominoes[currentTetromino][0][0].indexOf(1) == -1){
            tRow = -1;
        }
        tCol = 3;
        drawTetromino();
        if (canFit(tRow,tCol,currentRotation)){
            timeCount.addEventListener(TimerEvent.TIMER,onTime);
            timeCount.start();
        }else{
            gameOver = true;
```

Aquí se declara la función para cuando la figura llego al límite del escenario y es tiempo de que salga la siguiente figura de manera aleatoria o random que se muestra a un lado del escenario para salir una vez se cumpla la función.

```
private function drawTetromino():void{
    var ct:uint = currentTetromino;
    tetromino = new Sprite;
    addChild(tetromino);
    tetromino.graphics.lineStyle(0, 0x000000);
    for (var i:int=0; i < tetrominoes[ct][currentRotation].length; i++){
        for (var j:int=0; j < tetrominoes[ct][currentRotation][i].length; j++){
            if (tetrominoes[ct][currentRotation][i][j] == 1){
                tetromino.graphics.beginFill(colors[ct]);
                tetromino.graphics.drawRect(TS * j, TS * i, TS, TS);
                tetromino.graphics.endFill();
```

Aquí se establecen los gráficos de la rotación de la figura

```
    placeTetromino();
}
private function placeTetromino():void{
    tetromino.x = tCol * TS;
    tetromino.y = tRow * TS;
}
private function onKDown(event:KeyboardEvent):void{
    if (! gameOver){
        switch (event.keyCode){
            case 37 :
                if (canFit(tRow,tCol - 1,currentRotation)){
                    tCol--;
                    placeTetromino();
                }
```

Aquí se establece el movimiento en la tecla izquierda para que esta mueva hacia ese lado las figuras random que saldrán en el escenario.

```
    break;
case 38 :
    var rot:uint = currentRotation + 1 % tetrominoes[currentTetromino].length;
    if (rot == 4){
        rot = 0;
    }
    if (canFit(tRow,tCol,rot)){
        currentRotation = rot;
        removeChild(tetromino);
        drawTetromino();
        placeTetromino();
    }
    break;
case 39 :
    if (canFit(tRow,tCol + 1,currentRotation)){
        tCol++;
        placeTetromino();
    }
```

Aquí se establece el movimiento en la tecla derecha para mover las figuras que salen en el escenario hacia ese lado.

```
        break;
    case 40 :
        if (canFit(tRow + 1,tCol, currentRotation)){
            tRow++;
            placeTetromino();
        }else{
            landTetromino();
            generateTetromino();
        }
        break;
```

Aquí se establece el funcionamiento de la tecla flecha hacia arriba que es la que cumple la función de rotar las figuras mientas caen en el escenario.

```
private function canFit(row:int,col:int,side:uint):Boolean{
    var ct:uint = currentTetromino;
    for (var i:int = 0; i < tetrominoes[ct][side].length; i++){
        for (var j:int = 0; j < tetrominoes[ct][side][i].length; j++){
            if (tetrominoes[ct][side][i][j] == 1){
                //Out of left Boundary
                if (col + j < 0){
                    return false;
                }
                //Out of rigth Boundary
                if (col + j > 9){
                    return false;
                }
                //Out of botton Boundary
                if (row + i > 19){
                    return false;
                }
                //over another Boundary
                if (fieldArray[row + i][col + j] == 1){
                    return false;
```

Esta función establece a las figuras cuando están cayendo en el escenario.

```
        return true;
    }
    private function landTetromino():void{
        var ct:uint = currentTetromino;
        var landed:Sprite;
        for (var i:int = 0; i < tetrominoes[ct][currentRotation].length; i++){
            for (var j:int = 0; j < tetrominoes[ct][currentRotation][i].length; j++){
                if (tetrominoes[ct][currentRotation][i][j] == 1){
                    landed = new Sprite  ;
                    addChild(landed);
                    landed.graphics.lineStyle(0, 0x000000);
                    landed.graphics.beginFill(colors[currentRotation]);
                    landed.graphics.drawRect(TS * (tCol + j), TS * (tRow + i), TS, TS);
                    landed.graphics.endFill();
                    landed.name = "r" + (tRow + i) + "c" + (tCol + j);
                    fieldArray[tRow + i][tCol + j] = 1;
```

Esta función checa cuando una pieza esta sobre otra pieza haciendo que quede sobre esta.

```
    removeChild(tetromino);
    timeCount.removeEventListener(TimerEvent.TIMER, onTime);
    timeCount.stop();
    checkForLines();
}
private function checkForLines():void{
    for (var i:int = 0; i<20; i ++){
        if (fieldArray[i].indexOf(0) == -1){
            for (var j :int =0; j<10; j++){
                fieldArray[i][j] = 0;
                removeChild(getChildByName("r" + i + "c" + j));
                contador++;
            }
            for (j = i; j>= 0; j--){
                for (var k:int = 0; k<10; k++){
                    if (fieldArray[j][k] == 1){
                        fieldArray[j][k] = 0;
                        fieldArray[j + 1][k] = 1;
                        getChildByName("r" + j + "c" + k ).y +=TS;
                        getChildByName("r" + j + "c" + k ).name = "r"+ (j+1)+"c"+k;
```

Esta función checa que las líneas estén completas y cuenta puntos.

```
 private function onTime(e: TimerEvent):void{
     if (canFit(tRow + 1, tCol, currentRotation)){
         tRow++;
         placeTetromino();
     }else{
         landTetromino();
         generateTetromino();
     }
 }
 private function drawNext():void{
     if (getChildByName("next") != null){
         removeChild(getChildByName("next"));
         }
     var next_t:Sprite = new Sprite  ;
     next_t.x = 300;
     next_t.name = "next";
     addChild(next_t);
     next_t.graphics.lineStyle(0, 0x000000);
     for (var i : int = 0; i<tetrominoes[nextTetromino][0].length; i++){
         for (var j:int = 0; j<tetrominoes[nextTetromino][0][i].length; j++){
             if (tetrominoes[nextTetromino][0][i][j] == 1){
                 next_t.graphics.beginFill(colors[nextTetromino]);
                 next_t.graphics.drawRect(TS * j,TS * i,TS,TS);
                 next_t.graphics.endFill();
```

Aquí se declara la función de la pieza siguiente.

CONCLUSIONES

En esta actividad aprendimos a identificar nuevas funciones y a utilizar sokets lo cual tuvo un pequeño grado más de dificultad ya que tuvimos que investigar por diferentes medios la manera de poder hacer que el juego se pudiera jugar en dos computadoras, a la vez que teníamos que hacer pruebas y establecer límites y matrices o dimensiones en cada figura.