Proyecto Final: Tetris
Herramientas Multimedia
Manual Técnico
HM-ITI-07116

Alumno:
Jonathan Garcia Gonzalez (1730380)
Jovan Garcia Gonzalez (1730381)

Profesor:
MSI Mario Humberto Rodríguez Chávez

Introducción

Tetris es un videojuego de puzzle originalmente diseñado y programado por Alekséi Pázhitnov en la Unión Soviética. Fue lanzado el 6 de junio de 1984, mientras trabajaba para el Centro de Computación Dorodnitsyn de la Academia de Ciencias de la Unión Soviética en Moscú, RSFS de Rusia. Su nombre deriva del prefijo numérico griego *tetra-* (todas las piezas del juego, conocidas como Tetrominós que contienen cuatro segmentos) y del tenis, el deporte favorito de Pázhitnov.

En el *Tetris* se juega con los tetrominós, el caso especial de cuatro elementos de poliominós. Los poliominós se han utilizado en los rompecabezas populares por lo menos desde 1907, y el nombre fue dado por el matemático Solomon W. Golomb en 1953. Sin embargo, incluso la enumeración de los pentominós data de la antigüedad.

El juego (o una de sus muchas variantes) está disponible para casi cada consola de videojuegos y sistemas operativos de PC, así como en dispositivos tales como las calculadoras gráficas, teléfonos móviles, reproductores de multimedia portátiles, PDAs, reproductores de música en red e incluso como huevo de pascua en productos no mediáticos como los osciloscopios.

En este manual se verá la explicación de cómo es que realizamos este juego en base a todo nuestro conocimiento adquirido en la materia de herramientas multimedia usando nuestra lógica y modo de trabajo.

Índice

Desarrollo (scripts).

Fotograma 1. (Portada).

En el archivo .fla, el fotograma 1 consiste en la portada con los datos del proyecto y el título de este, la portada se encuentra animada con tweens que están definidos en la clase Principal.


```
//funcion portada
function portada(): void {
    var move1: Tween = new Tween(upv, "x", Strong.easeInOut, upv.x - 1000, upv.x, 1, true);
    var move2: Tween = new Tween(hm, "x", Strong.easeInOut, hm.x + 1000, hm.x, 1, true);
    var move3: Tween = new Tween(name1, "x", Strong.easeInOut, name1.x - 1000, name1.x, 1, true)
    var move4: Tween = new Tween(name2, "x", Strong.easeInOut, name2.x - 1000, name2.x, 1, true)

    var move11: Tween = new Tween(name3, "x", Strong.easeInOut, name3.x - 1000, name3.x, 1, true
    var alp: Tween = new Tween(alumnos, "alpha", Strong.easeInOut, 0, 1, 1, true);
    var alp2: Tween = new Tween(profesor, "alpha", Strong.easeInOut, 0, 1, 1, true);
    var alp3: Tween = new Tween(iti, "alpha", Strong.easeInOut, 0, 1, 1, true);

    var move5: Tween = new Tween(t, "x", Strong.easeInOut, t.x - 10, t.x, 1.5, true);
    var ani1: Tween = new Tween(t, "alpha", Strong.easeInOut, 0, 1, 1.5, true);
    var move6: Tween = new Tween(e, "y", Strong.easeInOut, e.y - 10, e.y, 1.5, true);
    var ani2: Tween = new Tween(e, "alpha", Strong.easeInOut, 0, 1, 1.5, true);
    var move7: Tween = new Tween(t2, "y", Strong.easeInOut, t2.y - 10, t2.y, 1.5, true);
    var ani3: Tween = new Tween(t2, "alpha", Strong.easeInOut, 0, 1, 1.5, true);
    var move8: Tween = new Tween(r, "y", Strong.easeInOut, r.y - 10, r.y, 1.5, true);
    var ani4: Tween = new Tween(r, "alpha", Strong.easeInOut, 0, 1, 1.5, true);
    var move9: Tween = new Tween(i, "y", Strong.easeInOut, i.y - 10, i.y, 1.5, true);
    var ani5: Tween = new Tween(i, "alpha", Strong.easeInOut, 0, 1, 1.5, true);
    var move10: Tween = new Tween(s, "x", Strong.easeInOut, s.x + 10, s.x, 1.5, true);
    var ani6: Tween = new Tween(s, "alpha", Strong.easeInOut, 0, 1, 1.5, true);

    var move17: Tween = new Tween(ofig, "y", Strong.easeInOut, ofig.y - 10, ofig.y, 3, true);
    var ani7: Tween = new Tween(ofig, "alpha", Strong.easeInOut, 0, 1, 2.5, true);
```

Fotograma 2. (Selección de dificultad).

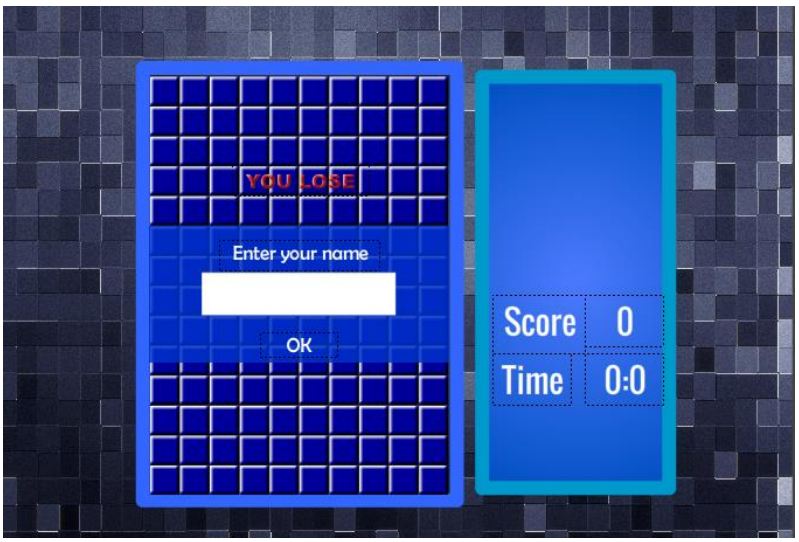
Para realizar esto lo que hicimos en la interfaz poner un número y el nombre que representaba este Dificultad 1: Muy facil a 10 Inferno, esta dificultad se cambia dando clic a las flechas derecha: para incrementarla e izquierda para disminuirla.

```
144
145         function dplus(e: MouseEvent): void {
146             dificultadnum++;
147             fdificultad();
148             var mas: Tween = new Tween(dificultad, "alpha", Strong.easeInOut, 0, 1, 1.5, true);
149         }
150
151         function dminus(e: MouseEvent): void {
152             dificultadnum--;
153             fdificultad();
154             var menos: Tween = new Tween(dificultad, "alpha", Strong.easeInOut, 0, 1, 1.5, true);
155         }
156
157         function fdificultad(): void {
158
159             if (dificultadnum == 0) {
160                 dificultadnum = 1;
161             }
162             if (dificultadnum == 1) {
163                 dificultad.text = "Muy facil";
164             }
165             if (dificultadnum == 2) {
166                 dificultad.text = "Facil";
167             }
168             if (dificultadnum == 3) {
169                 dificultad.text = "Moderado";
170             }
171             if (dificultadnum == 4) {
172                 dificultad.text = "Normal";
173             }
174             if (dificultadnum == 5) {
175                 dificultad.text = "Avanzado";
176             }
```

La dificultad funciona con una variable dificultadnum la cual aumenta y disminuye de 1 a 10, más adelante en un timer se usa esta variable para representar otro valor

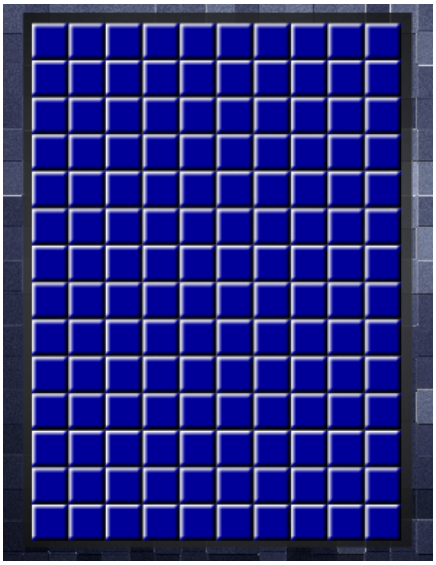
Fotograma 3. (Juego).

En el fotograma 3 se encuentra la parte del juego, donde se encuentran las figuras y los elementos que usaremos para crear las figuras.



Matriz de cuadros.

El método que se usó para mostrar las figuras en el momento que colisionan fue que en el escenario se creó una matriz de cuadros que se convirtieron en símbolos MovieClip para que al comenzar el juego se pongan invisibles y al momento de colisionar se pongan visibles y dibujen la figura que callo.



```
    cuadros5[figc].transform.colorTransform = Rojo;
}
var mapa: Array = new Array(m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14
for (var i: int = 0; i < 5; i++) {
    figuras[i].x = placefig.x;
```

Cada instancia de los cuadros se colocó en un array llamado mapa.

Caída de figuras.

Para realizar las caídas formamos las 5 posibles figuras con cuadros por medio de código y ordenándolos en sus coordenadas correspondientes y después en un timer para que estas caigan le aumentamos 40 en 40 en el eje de las y cada tiempo asignado según por la dificultad que se asigna por medio del número seleccionado anteriormente.

Selección de velocidad de caída con un timer:

```
341         var cont1: int;
342         //se determina la velocidad del bloque en base a la dificultad
343         if (dificultadnum == 1) {
344             vel = 1000;
345         }
346         if (dificultadnum == 2) {
347             vel = 900;
348         }
349         if (dificultadnum == 3) {
350             vel = 850;
351         }
352         if (dificultadnum == 4) {
353             vel = 750;
354         }
355         if (dificultadnum == 5) {
356             vel = 700;
374
375         //timer para que caiga el bloque
376         var timer: Timer = new Timer(vel, cont1++);
377         //variables para los ciclos
```

Caída de las figuras: (Esta parte de código se hace con todas las figuras, este es un ejemplo de la figura 2)

```
                posY4 = c24.y;
            }
            if (bloque == 1) {
                c21.y += 40;
                c22.y += 40;
                c23.y += 40;
                c24.y += 40;
                posY1 = c21.y;
                posY2 = c22.y;
                posY3 = c23.y;
                posY4 = c24.y;
            }
            if (bloque == 2) {
```

Colisiones.

Para las colisiones de las figuras se utilizaron lo que es la matriz de cuadros y la figuras que están cayendo, en primer lugar se verifica para cada figura si ha llegado al límite de la zona. si es así se verifica cuál figura es la que colisionó y a continuación recorre la matriz para dibujar la figura y asignarle el mismo color que la figura.

```
function tiempo(tiempoevent: TimerEvent): void {
            if (posY1 == 620 || posY2 == 620 || posY3 == 620 || posY4 == 620) {
                for (k = 0; k < 4; k++) {
                    for (j = 0; j < 130; j++) {
                        if (cuadros1[k].y == mapa[j].y && cuadros1[k].x == mapa[j].x) {
                            mapa[j].visible = true;
                            mapa[j].transform.colorTransform = Morado;
                            colores[j]=1
                        }
                        if (cuadros2[k].y == mapa[j].y && cuadros2[k].x == mapa[j].x) {
                            mapa[j].visible = true;
                            mapa[j].transform.colorTransform = Naranja;
                            colores[j]=2
                        }
                        if (cuadros3[k].y == mapa[j].y && cuadros3[k].x == mapa[j].x) {
                            mapa[j].visible = true;
                            mapa[j].transform.colorTransform = Amarillo;
                            colores[j]=3
                        }
                        if (cuadros4[k].y == mapa[j].y && cuadros4[k].x == mapa[j].x) {
                            mapa[j].visible = true;
                            mapa[j].transform.colorTransform = Celeste;
                            colores[j]=4
                        }
                        if (cuadros5[k].y == mapa[j].y && cuadros5[k].x == mapa[j].x) {
                            mapa[j].visible = true;
```

Movimiento de figuras.

Se usó una función press de evento de teclado para que al presionar una tecla haga una señal dándole un valor a una variable para realizar un procedimiento en otra función de Event

```
        //funcion presionado
        function press(event: KeyboardEvent): void {
            //determinar la tecla izquierda ascii no 37
            if (event.keyCode == 37) {
                izq = true;
                delay++;
            } else {
                if (event.keyCode == 38) {
                    arr = true;
                    delay++;
                } else {
                    if (event.keyCode == 39) {
                        der = true;
                        delay++;
                    } else {
                        if (event.keyCode == 40) {
                            abaj = true;
                        }
                    }
                }
            }
        }
```

También se hizo una función soltar para dejar de realizar esta acciones

```
1085            //funcion para soltar las teclas
1086            function soltar(event: KeyboardEvent): void {
1087                if (event.keyCode == 37) {
1088                    izq = false;
1089                    delay = 0;
1090                } else {
1091                    if (event.keyCode == 38) {
1092                        arr = false;
1093                        delay = 0;
1094                    } else {
1095                        if (event.keyCode == 39) {
1096                            der = false;
1097                            delay = 0;
1098                        } else {
1099                            if (event.keyCode == 40) {
1100                                abaj = false;
1101                            }
1102                        }
1103                    }
1104                }
1105            }
```

En la función movimiento se realiza lo siguiente para todas las figuras, para que se muevan de izquierda y derecha, de manera fluida también gracias a la variable delay

```
1238            if (delay >= 1) {
1239                delay = 0;
1240                if (bloque == 0) {
1241                    if (izq == true && c11.x > 60) {
1242                        c11.x -= 40;
1243                        c12.x -= 40;
1244                        c13.x -= 40;
1245                        c14.x -= 40;
1246                        for (k = 0; k < 4; k++) {
1247                            for (j = 0; j < 130; j++) {
1248                                if (cuadros1[k].x == mapa[j].x && cuadros1[k].y == mapa[j].y && mapa[j].visible == true) {
1249                                    c11.x += 40;
1250                                    c12.x += 40;
1251                                    c13.x += 40;
1252                                    c14.x += 40;
1253                                }
1254                            }
1255                        }
1256                    } else {
1257                        if (der == true && c13.x < 420) {
1258                            c11.x += 40;
1259                            c12.x += 40;
1260                            c13.x += 40;
1261                            c14.x += 40;
1262                            for (k = 0; k < 4; k++) {
1263                                for (j = 0; j < 130; j++) {
1264                                    if (cuadros1[k].x == mapa[j].x && cuadros1[k].y == mapa[j].y && mapa[j].visible == true)
1265                                        c11.x -= 40;
1266                                        c12.x -= 40;
1267                                        c13.x -= 40;
1268                                        c14.x -= 40;
1269                                    }
1270                                }
```

Para la rotación de estas se estaba modificando la posición para formar diferente posición de figura moviendo los cuadros de 40 en el eje de las x y de las y. esto da la ilusión de que la figura en verdad está rotando pero lo que se hace es que se cambian de posición algunos cuadros para darle otra forma.

Ejemplo rotación de figura 1:

```
1133                    delay = 0;
1134
1135                    if (bloque == 0) {
1136                        formal++;
1137                        if (formal == 4) {
1138                            formal = 0;
1139                        }
1140                        if (formal == 0) {
1141                            c14.y -= 80;
1142                            c13.x += 40;
1143                            c13.y += 40;
1144                        }
1145                        if (formal == 1) {
1146                            c11.y += 40;
1147                            c11.x += 40;
1148                        }
1149                        if (formal == 2) {
1150                            c11.y -= 40;
1151                            c11.x -= 40;
1152                            c14.y += 80;
1153                        }
1154                        if (formal == 3) {
1155                            c13.y -= 40;
1156                            c13.x -= 40;
1157                        }
1158                    }
```

para el movimiento de caída acelerada simplemente se creo otro timer con mayor velocidad o mejor dicho menor tiempo para que incremente el contador de este.

Eliminacion de lineas.

La eliminación de líneas se realiza mediante la función eliminarln que se manda a llamar cuando se hace una colisión, en esta dicha función se verifica fila por fila si se encuentra visible es decir si está completa, si el caso es verdadero entonces el contador de las filas llegará hasta 10 (número total por fila), una vez verifica todas las filas, se eliminan las que tengan 10 figuras visibles, después se verifica si hubo linea borrada en caso de que si se realiza el recorrido de líneas.

```
function eliminarln() {
    //se cuenta el numero de bloques que se encuentran visibles en cada fila
    for (n = 0; n < 10; n++) {
        if (mapa[n].visible == true) {
            fila1++;
        }
    }
    for (n = 10; n < 20; n++) {
        if (mapa[n].visible == true) {
            fila2++;
        }
    }
    for (n = 20; n < 30; n++) {
        if (mapa[n].visible == true) {
            fila3++;
        }
    }
    for (n = 30; n < 40; n++) {
        if (mapa[n].visible == true) {
            fila4++;
        }
    }
```

```
if (fila1 == 10) {
    for (n = 0; n < 10; n++) {
        mapa[n].visible = false;
    }
    filacompleta = true;
    //se aumenta el contador filasrec para saber cuantas filas se completaron
    filasrec++;
    filai = 10;
}

if (fila2 == 10) {
    for (n = 10; n < 20; n++) {
        mapa[n].visible = false;
    }
    filacompleta = true;
    filasrec++;
    filai = 20;
}
if (fila3 == 10) {
    for (n = 20; n < 30; n++) {
        mapa[n].visible = false;
    }
    filacompleta = true;
    filasrec++;
    filai = 30;
}
```

Fotograma 4 Resultados

Los resultados del juego se realizan en el fotograma final, el fotograma 4, al perder el juegos, o sea colisionando en la última fila de la zona de juego, se toma los datos del jugador para que posteriormente se vaya al fotograma 4 y se impriman en textos dinámicos los resultados del jugador, en caso de ser multijugador el puntaje se actualizará al terminar de jugar el segundo.

Tetris en Red.

Para realizar el juego en red utilizamos variables de tipo LocalConnection estas nos sirven para hacer una conexión local con otro archivo, aplicación o documento, así que para funcionar deben de estar en una red en donde los dos estén conectados simultáneamente y abrir el juego cliente y servidor al mismo tiempo.

Para hacer la comunicación entre el juego cliente y servidor mandamos información o datos de variables al otro archivo .swf y le mandamos hacer diferentes funciones.

Conexión de cliente a servidor.

```
84          //coneccion con el servidor para recibir los datos necesarios
85          connection.connect("servidor");
86          connection.client = this;
87          //clase principal: se detiene en el primer fotograma (portada)
88          stop();
```

Mandar datos por medio de parámetros, también se da el nombre del que lo envía “servidor” y la función que realizara el otro archivo flash. (“esperar”).

```
129
130     }
131     function jugar(e: MouseEvent): void {
132         connection.send("servidor", "esperar", difnum.text);
133         gotoAndStop(3);
134         fotograma3();
```

El cliente recibe los datos realizando la función indicada

```
73      //FUNCION esperar: Espera al servidor a que inicie el y designe la dificultad
74      public function esperar(dificultad: String): void {
75          trace(dificultad);
76          dificultadnum = Number(dificultad);
77          jugar();
78
79      }
```

Usamos esto para mandar la dificultad que el servidor elige, mandar y recibir los datos del jugador para actualizar el marcador al finalizar el juego.

Conclusión.

Como conclusión tenemos que usando métodos de lógica y diseñando nuestros propios algoritmos podemos crear diversas aplicaciones de entretenimiento, en el caso del tetris tuvimos que realizar varias ideas para poder llegar a un objetivo, varias de las ideas fallaron en cumplirlo pero nos ayudó a comprender el comportamiento y estructura de nuestro programa.

Usando diversos métodos que el flash ofrece podemos crear un contenido bastante aceptable y competente para el mercado.

Mediante el uso de ideas compartidas podemos agilizar el trabajo en un equipo en un Proyecto.

Usar diferentes medio para llegar a un fin puede resultar útil para la realización de un Proyecto. Cuando surgía un problema usando lógica y realizando algoritmos o diagramas de ideas podríamos llegar a una solución, lo que nos hizo ver la importancia del razonamiento lógico.