5 Texto

En esta parte del tutorial no se verá nada del mundo 3D, que más tiene en interés al autor, y es que hasta este momento XNA 3.1 no tiene una librería del texto en 3D, como si lo tiene DirectX u OpenGL, pero que de todas formas se tomará en cuenta en este escrito.

El tipo de fuente que se utiliza en XNA es prácticamente toda aquella que se tenga instalada en el Sistema Operativo o que se pueda agregar como una textura de una fuente. Las primeras se agregarán al proyecto como una descripción del Sprit Font, así que debe tener instalado la fuente. La segunda toma una imagen con los caracteres de la fuente, por lo que no necesita la fuente instalada en la computadora.

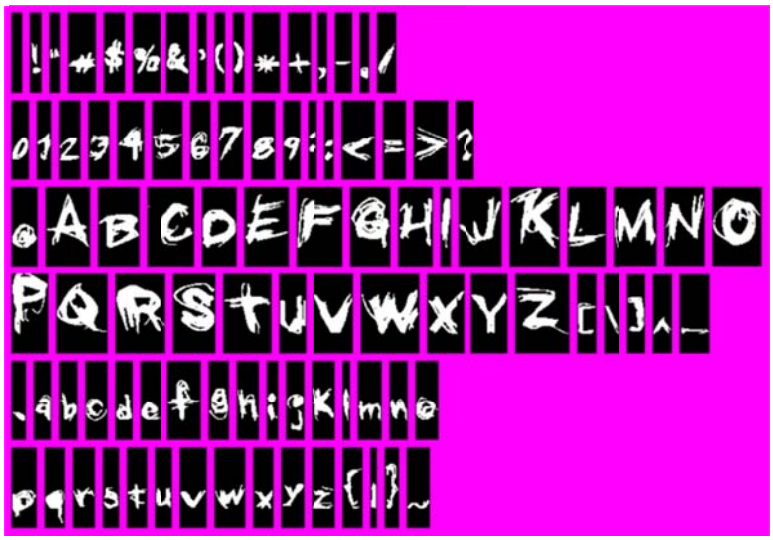

Ilustración 5-1 Sprite Font Texture

Para comenzar este ejemplo, necesitará descargar las imágenes de las fuentes que se agregaron en la misma carpeta en que descargo este tutorial. Pero si quieren crear las suyas pueden descargar la herramienta **Bitmap Font Maker de XNA CREATORS CLUB ONLINE**[14]. Estas texturas serán procesadas por el Content Pipeline de XNA como **Sprite Font Texture**, que es uno de los procesos estándar que proporciona XNA.

Lo que hace el proceso **Sprite Font Texture** es tomar una entrada, que en este caso es de tipo **Texture2DContent** que representa una textura regular de dos dimensiones; y la transforma en una fuente dando como salida a un **SpriteFontContent**. Esto lo hace al cambiar los pixeles de **Color.Magenta** a **Color.TransparentBlack**, y claro estas texturas tienen un canal alfa para hacer transparentes las regiones oscuras.

5.1 SpriteFont

En este ejemplo sólo se mostrará un conjunto de enunciados que muestran el uso de texto en XNA. Así que comenzaremos por crear un proyecto nuevo de XNA Game Studio 3.1, utilizando la plantilla **Windows Game (3.1)**.

Ahora hay que agregar el **SpritFont**, esto se hace en el explorador de soluciones del proyecto. Como ya se había visto anteriormente, coloque el cursor sobre el icono de **Content** y oprima el botón derecho del ratón para agregar un nuevo elemento al proyecto, como se muestra en la Ilustración 5-2.

[14] http://creators.xna.com/es-ES/utilities/bitmapfontmaker

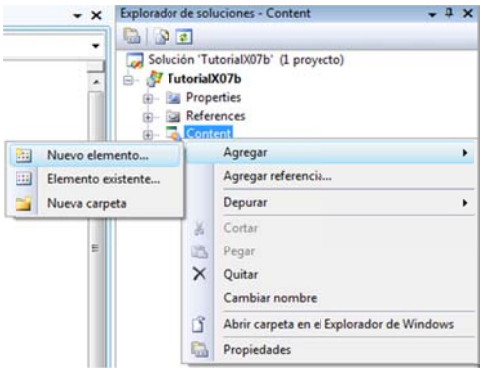

Ilustración 5-2 Nuevo elemento…

Inmediatamente se abre una ventana diálogo, **Agregar nuevo elemento – Content**, como se muestra en la Ilustración 5-3. Seleccione la plantilla **Sprite Font**, es la que muestra una letra A, cambie el nombre por el de la fuente que quiere colocar en la aplicación y dé clic en el botón **Agregar**; no es necesario que el nombre del **SpriteFont** sea el mismo que el de la fuente, es sólo por orden. Recuerde que para poder utilizar el **SpritFont** debe tener instalada la fuente en el Sistema Operativo.

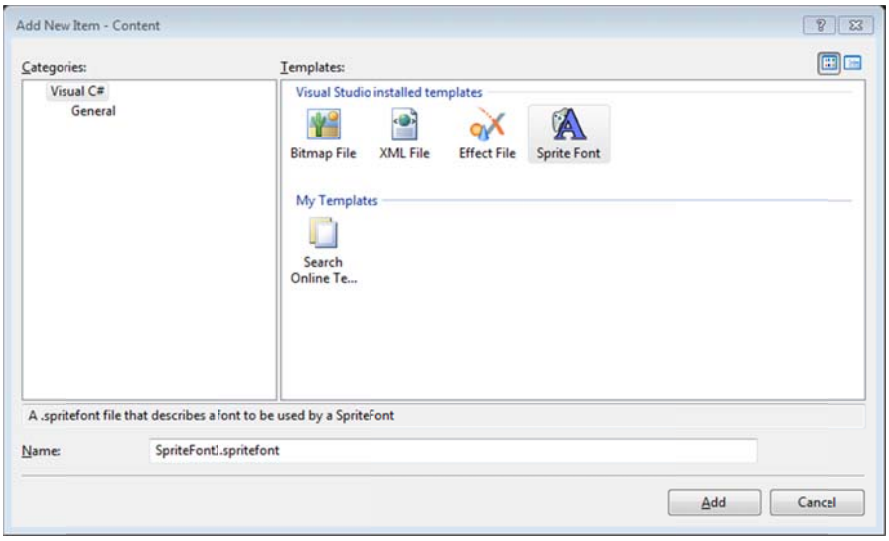

Ilustración 5-3 Agregar un Sprite Font

Visual Studio abrirá de inmediato el archivo ***.spritefont** que agregamos al proyecto anteriormente. Como podrá ver es un archivo XML que contiene la descripción de una fuente y que será leído por el Content Pipeline de XNA; no es necesario que aprenda sobre XML para poder agregar texto en XNA, sin embargo, sería una buena idea hacerlo porque usted mismo podría crear sus propia definición de documentos o esquema para instanciar los modelos gráficos dentro de un mundo 3D, dándoles su poción, orientación y nombre a cada uno de ellos. Además que .NET tiene su librería para manipular XML.

En fin, el XML contiene diez etiquetas que corresponden a las propiedades que comúnmente tienen las fuentes, las cuales se presenta en la Tabla 5-1[15].

Tabla 5-1

Etiqueta	Tipo de dato	Descripción

[15] Para más información revise la siguiente dirección: http://msdn.microsoft.com/en-us/library/bb447759.aspx

<FontName>	String	Es el nombre de la fuente que se tiene instalada en el Sistema Operativo, y que prácticamente toma cualquiera que esté, excepto las de tipo *.fon.
<Size>	Single	Es el tamaño en punto flotante de la fuente.
<Spacing>	Single	Es el número de píxeles a añadir entre cada carácter cuando la cadena se dibuja.
<UseKerning>	Boolean	Especifica si será usado el ajuste de espacio cuando se dibuje la fuente. Su valor por default es true.
<Style>	"Regular," "Bold," "Italic," o "Bold, Italic"	Es el estilo de la fuente a ser importada.
<DefaultCharacter>	Char	Es el carácter Unicode que substituirá al carácter que no se encuentra en la fuente importada. La especificación de esta etiqueta es opcional.
<CharacterRegions>	Uno o más etiquetas <CharacterRegion>	Uno o más rangos numéricos indicando que subconjunto de caracteres Unicode será importado.
<CharacterRegion>	Una etiqueta <Start> y una <End>	Es el inicio y el final de una región de caracteres Unicode.
<Start>	Char	Es el primer carácter Unicode a incluir en un <CharacterRegion>
<End>	Char	Es el último carácter Unicode a incluir en un <CharcaterRegion>

En la línea 4 del Código 5-1 el nombre de la fuente se ha cambiado por **Kids**, en la línea 6 se ha puesto el valor de 20 para que se pueda apreciar la cadena en la ilustración que enseguida aparecerá; el valor del espaciado será de 2, línea 8, pues por default es 0 y a veces por el tipo de fuente queda muy amontonado como le ocurre a **Kids**. También se ha des comentado la etiqueta **DefaultCharacter**, por si hubiera alguna letra de la cadena que no pueda ser encontrada en el intervalo de caracteres de código ASCII imprimible que se le ha asignado, líneas 18 y 19.

Código 5-1

```
1.      <?xml version="1.0" encoding="utf-8"?>
2.      <XnaContent xmlns:Graphics="Microsoft.Xna.Framework.Content.Pipeline.Graphics">
3.        <Asset Type="Graphics:FontDescription">
```

```
4.          <FontName>Kids</FontName>
5.
6.          <Size>20</Size>
7.
8.          <Spacing>2</Spacing>
9.
10.         <UseKerning>true</UseKerning>
11.
12.         <Style>Regular</Style>
13.
14.         <DefaultCharacter>*</DefaultCharacter>
15.
16.         <CharacterRegions>
17.           <CharacterRegion>
18.             <Start>&#32;</Start>
19.             <End>&#126;</End>
20.           </CharacterRegion>
21.         </CharacterRegions>
22.       </Asset>
23.     </XnaContent>
```

Dejando a un lado el XML, es momento de regresar a C#, y comenzaremos declarando una instancia de **SpritFont** en las variables de instancia de la clase **Game1** que generó Visual Studio en el archivo **Game1.cs**.

Código 5-2

```
1.      public class Game1 : Microsoft.Xna.Framework.Game
2.      {
3.          GraphicsDeviceManager graphics;
4.          SpriteBatch spriteBatch;
5.          SpriteFont kids;
6.
7.          public Game1()
8.          {
9.              graphics = new GraphicsDeviceManager(this);
10.             Content.RootDirectory = "Content";
11.             this.Window.Title = "TutorialX07";
12.             this.Window.AllowUserResizing = true;
13.             this.IsMouseVisible = true;
14.         }
15.
16.         protected override void Initialize()
17.         {
18.             base.Initialize();
19.         }
20.
21.         protected override void LoadContent()
22.         {
23.             spriteBatch = new SpriteBatch(GraphicsDevice);
24.
25.             kids = Content.Load<SpriteFont>("Kids");
26.         }
27.
28.         protected override void UnloadContent()
29.         {
30.         }
31.
32.         protected override void Update(GameTime gameTime)
33.         {
34.             if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
35.                this.Exit();
36.
37.             base.Update(gameTime);
38.         }
39.
40.         protected override void Draw(GameTime gameTime)
41.         {
42.             GraphicsDevice.Clear(Color.White);
43.
44.             spriteBatch.Begin();
45.             spriteBatch.DrawString(kids, "¡Hola mundo!", new Vector2(0.0F, 0.0F),
```

```
46.                 Color.Black);
47.             spriteBatch.End();
48.
49.             base.Draw(gameTime);
50.         }
51.     }
```

En el método **LoadContet**, líneas 21 – 26 Código 5-2, se carga el ***.spritfont** que se añadió en el **Content**. El tipo de dato que se usa es de tipo **SpritFont** en el método genérico **Content.Load**. Hay que recordar que sólo se debe tomar el **Asset Name** con el que **ContentManaged** lo identifica, no se necesita la extensión del archivo.

Como último paso, hay que mandar a dibujar el **SpriteFont** en el método **Draw**, líneas 40 - 50. Para comenzar hay que preparar el dispositivo gráfico para dibujar los sprites con **SpriteBatch.Begin**. Enseguida se dibuja la cadena con el método **SpritBatch.DrawString** que tiene seis sobrecargas, en este caso se seleccionó el primero que nos da el **IntelliSense**.

```
public void DrawString(SpriteFont spriteFont, string text, Vector2
position, Color color)
```

Cuyos parámetros se muestran en la Tabla 5-2.

Tabla 5-2

Parámetro	Descripción
spriteFont	El sprit de la fuente.
text	La cadena a dibujar.
position	La localidad, en coordenadas de la pantalla, donde será dibujado el texto.
color	Es el color para el texto.

Inicie el programa con la tecla **F5**, para poder ver algo similar a la Ilustración 5-4, si surge cualquier error de compilación encuentre el problema para resolverlo y trate de nuevo.

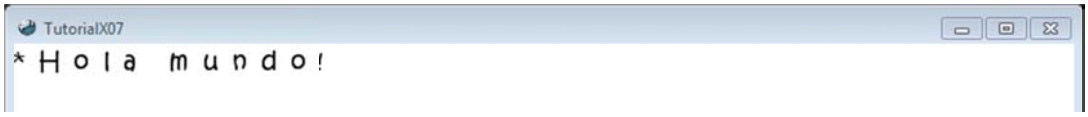

Ilustración 5-4 Carácter desconocido

Como se puede ver, cuando se agregan caracteres con acentos, diéresis o la letra “ñ”, no podrán mostrarse, a cambio se dibuja el signo que se le dio a la etiqueta **DefaultCharacter** del **Spritfont**; esto es porque dichos caracteres no se en encuentran dentro del intervalo que se le dio al XML[16]. Para resolver esto, hay que agregar otro intervalo de caracteres ASCII o aumentar el número en que termina el intervalo.

```
<CharacterRegions>
    <CharacterRegion>
        <Start>&#32;</Start>
        <End>&#255;</End>
    </CharacterRegion>
```

[16] Véase la siguiente dirección para revisar en que intervalo del código ASCII se encuentran ciertos caracteres http://msdn.microsoft.com/en-us/library/4z4t9ed1(VS.71).aspx

En este caso se aumentó el intervalo, para mostrar los signos de puntuación que se ocupan en el español. Vuelva a correr el programa y verá que el mensaje escrito será el que quería mostrar.

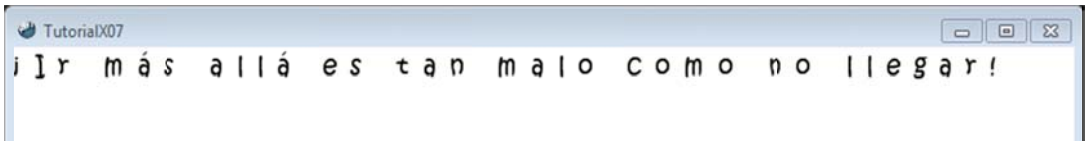

Ilustración 5-5 Aumenta el rango de caracteres reconocibles

Las demás sobrecargas del método **SpriteBatch.DrawString** para dibujar texto, se dejan como ejercicio para el lector.

5.2 Sprite Font Texture

En caso que el tipo de fuente no se encuentre instalado en el Sistema Opertativo, o se quiera enriquecer más el estilo de la fuente se podría exportar al programa una imagen con las letras necesarias que se ocupan en el proyecto, para ello el **Sprite Font Texture** es una buena solución.

Comience por agregar un nuevo elemento al **Content**; esta vez será un archivo de imagen como se muestra en la Ilustración 5-6. A estas alturas ya sabrá como agregar nuevos elementos en el **Content**, así que omitiremos esos pasos. La imagen debe tener ciertas propiedades para que el **Content Pipeline** pueda procesarla, y ésta son que tenga un color magenta y este habilitado el canal Alfa.

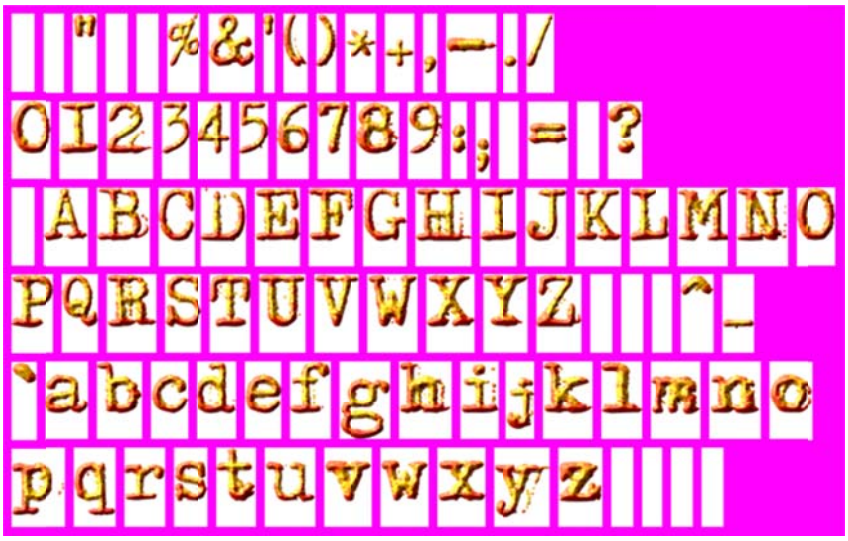

Ilustración 5-6 Adler

Sin embargo, hay que hacer unas modificaciones en el **Content Processor**, que es el que se encargara en procesar la imagen de forma adecuada para crear un **Sprite Font Texture**. Así que diriga el icono de la imagen que acaba de agregar en el **Content**, lo anterior se realiza en el explorador de soluciones, posteriormente haga clic para poder ver las propiedades del archivo. Por default, el valor que tiene la propiedad **Content Processor** de la lista **XNA Framework Content Pipeline**, es **Texture – XNA Framework**, véase Ilustración 5-7.

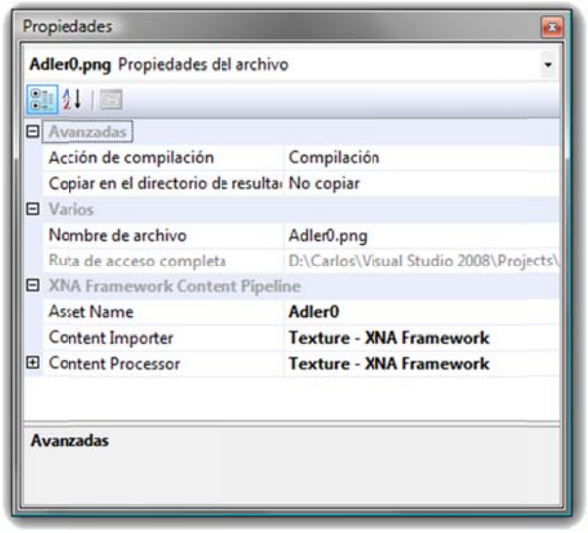

Ilustración 5-7 Propiedades de una imagen

Cambie la manera en que el **Content Pipeline** procesará la imagen por **Sprite Font Texture – XNA Framework**, como se ve en la Ilustración 5-8.

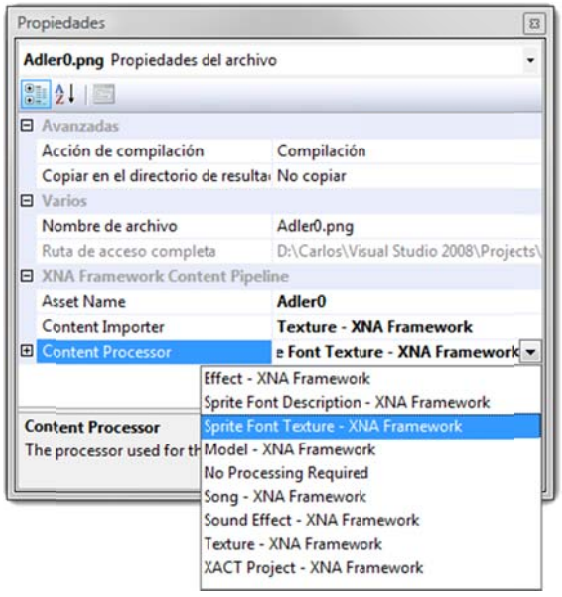

Ilustración 5-8 Cambiando el procesador de contenido

Cabe aclarar que la textura que ha tomado el **Content Pipeline** debe tener todos los caracteres necesarios para poder dibujar la cadena deseada, porque si no es así, también abra que dejar un carácter por default en caso que no exista.

Ahora hay que agregar el **Sprite Font Texture** al código, comience por declarar una variable de instancia en la clase **Game1**, línea5 Código 5-3.

En el método **LoadContent**, líneas 22 -29, al igual que en el ejemplo anterior, se carga el **Sprit Font Texture** con el nombre del **Asset**. Y es aquí donde se modifican las propiedades del **SpriteFont**, el más importante es el **SpriteFont.DefaultCharacter**, línea 26, pues no falta el símbolo que no se pueda representar y dé un error en tiempo de ejecución, hay que asegurarse también que el carácter que se tome por default esté en el **Sprite Font Texture**, porque si así no fuera se caería en el mismo problema y esto también lo es en el XML del ejemplo anterior.

Código 5-3

```
1.      public class Game1 : Microsoft.Xna.Framework.Game
2.      {
3.          GraphicsDeviceManager graphics;
4.          SpriteBatch spriteBatch;
5.          SpriteFont adler;
6.
7.          public Game1()
8.          {
9.              graphics = new GraphicsDeviceManager(this);
10.             Content.RootDirectory = "Content";
11.             this.Window.Title = "TutorialX07";
12.             this.Window.AllowUserResizing = true;
13.             this.IsMouseVisible = true;
14.
15.         }
16.
17.         protected override void Initialize()
18.         {
19.             base.Initialize();
20.         }
21.
22.         protected override void LoadContent()
23.         {
24.             spriteBatch = new SpriteBatch(GraphicsDevice);
25.             adler = Content.Load<SpriteFont>("Adler0");
26.             adler.DefaultCharacter = '*';
27.             adler.LineSpacing = 0;
28.             adler.Spacing = 0;
29.         }
30.
31.         protected override void UnloadContent()
32.         {
33.         }
34.
35.         protected override void Update(GameTime gameTime)
36.         {
37.             if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
38.                 this.Exit();
39.             base.Update(gameTime);
40.         }
41.
42.         protected override void Draw(GameTime gameTime)
43.         {
44.             GraphicsDevice.Clear(Color.Black);
45.
46.             spriteBatch.Begin();
47.             spriteBatch.DrawString(adler, DateTime.Now.TimeOfDay.ToString(),
48.             new Vector2(GraphicsDevice.Viewport.Width / 4.5F,
49.                 GraphicsDevice.Viewport.Height / 2),
50.                 Color.White);
51.             spriteBatch.End();
52.             base.Draw(gameTime);
53.         }
54.     }
```

En el método **Draw**, líneas 42 - 53, como en el ejemplo anterior de dibuja la cadena con el mismo método sobrecargado **SpriteBatch.Drawing**, línea 47. Sin embargo, esta vez se mostrará el reloj del sistema como la cadena; para ello se obtiene la propiedad **DateTime.Now.TimeOfDay** en forma de **String**. Y la posición de la cadena será tomando en cuenta el ancho y alto del Viewport. Para que no se vea alterado el color de la cadena, pues se pretende que el **Sprite Font Texture** tiene los colores y efectos que no nos puede dar el XML, se deja en **Color.White**.

Por fin, esto es todo con respecto a texto en XNA, o por lo menos lo más básico para colocar información valiosa en el Viewport, pues a veces no sabemos qué valores toma los vértices de nuestros modelos 3D o de colisión en tiempo real.

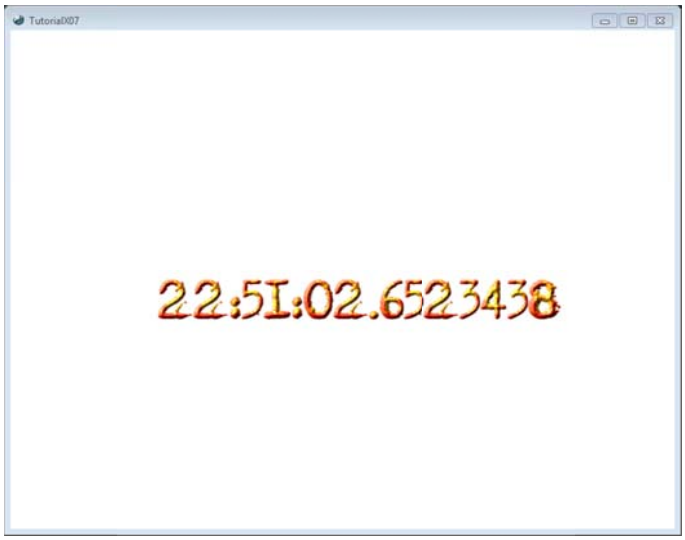

Ilustración 5-9 Reloj

Inicie el programa con la tecla **F5** y corrija cualquier error de compilación si así lo necesitará; verá un lindo reloj más o menos como la Ilustración 5-9.