Guía de iniciación al GLScene

Autor: David Martín de Vidales

ÍNCICE

1. Introducción y objetivos

En este manual se dará una visión general de GLScene, así como también los conocimientos necesarios para la instalación y uso básico de esta librería, describiendo los componentes esenciales. No se pretende hacer un manual exhaustivo de todos los componentes ni posibilidades que ofrece, sino que pretende ser una guía de inicio.

Se da por supuesto que el lector tiene conocimientos de programación en Delphi, no obstante se hará un pequeño recordatorio de este entorno de programación y los pasos serán muy detallados.

¿Que es GLScene?

GLScene es una librería 3D para Delphi basada en OpenGL que proporcionan un conjunto de componentes y objetos que permiten el renderizado de escenas 3D de un modo sencillo. Esta en desarrollo desde 1999 (creada inicialmente por Mike Lischke), es de código abierto y bajo la Licencia Publica Mozilla.

Actualmente esta soportada por las versiones 5, 6 y 7 de Delphi, además existen versiones para Borlad C++ Builder versiones 5 y 6. También tiene cierta compatibilidad con Kylix (la versión para Linux de Delphi), pero aun no están disponibles todas las características y esta en desarrollo.

GLScene está en continuo desarrollo, y casi diariamente salen nuevas versiones y mejoras de la librería, para poder obtener la ultima versión se ha de acudir al CVS (Concurrent Version System) del proyecto, que se encuentra en sourceforge:
http://cvs.sourceforge.net/viewcvs.py/glscene/

¿Y que es OpenGL?

OpenGL es una librería gráfica para el renderizado de gráficos 2D y 3D diseñada por Silicon Graphics, y aceptada como uno de los principales estándares por la mayoría de fabricantes de hardware.

2. Características

GLScene tiene docenas de componentes y objetos para ayudar y facilitar el diseño de aplicaciones con escenarios 3D. A continuación se proporciona un listado organizado de sus principales características:

· *Descripción de escena*

- Estructura jerárquica de objetos, con tantos objetos como permita la memoria, fácilmente extensible.
- Administrador interactivo de escena
- Sencillas funciones de rotación y translación para cada objeto.
- Objetos predefinidos (todos los estándar).
- Objetos de estructura para manejar objetos compuestos (objetos dummy cube y proxy)
- Objetos procedurales (mapas de alturas, sólidos de revolución, etc.)
- Soporte para objetos 2D y 1D (sprites, líneas, puntos..) totalmente integrados.
- Objetos HUD (Heads-up display)
- Objetos útiles (rejillas, cielos, flechas, etc.)
- Soporte de clases para acceso directo a OpenGL
- Objetos de cámara y luces que pueden ser usados en cualquier lugar de la jerarquía de objetos de la escena.
- El comportamiento de los objetos se puede unir a los objetos para animarlos.
- Efectos especiales (antes y después del renderizado.)
- Soporte de sistemas de partículas
- Importación de ficheros 3D Studio con cálculo automático y preciso del vector normal e importación de las coordenadas de texturas.
- Otros formatos importados, como: OBJ/OBJF, SMD, MD2, STL, TIN, PLY, etc.

· *Materiales*

- Objeto material fácil de usar y optimizado.
- Librería de materiales para compartir y reutilizar materiales.
- Soporte de las componentes ambiente, difusa, emisión, especular y brillo.
- Soporte de modos de fundido (transparencia, aditivo, etc.)
- Canal alfa (transparencia)
- Soporte de los formatos de textura de OpenGL, incluidos los comprimidos (DXT, S3TC etc.)
- Clases para soporte del sombreado Cg.
- Soporte de imagen polimorfica para las texturas (permite multitud de formatos así como también texturas procedurales)
- Más de 150 colores predefinidos, además de los colores estándar y la especificación directa RGBA
- Propiedades de movimiento y escalado te texturas fáciles de usar independientemente de las coordenadas de la textura.
- Soporte de Bitmaps de 32 bits

· *Renderizado*

- Uso automático del hardware OpenGL si esta disponible.
- Modelo de cámara con longitud focal y objetivo.
- Múltiples vistas de una o más escenas, fácil cambio de vista mediante de selección de

la cámara.
- Soporte para niebla y 'profundidad de la vista'
- Soporte para renderizado a fichero, a bitmap o a impresora en cualquier color y resolución de pixel.
- Soporte para pantalla completa y cambio dinámico de resolución
- Efectos de reflexiones y espejos.
- Soporte para transparencia de sistemas de partículas de alto rendimiento.
- Desecho automático de frustrum (jerárquico o por objeto)

· *Animación*
- Propagación de los eventos de progresión en el tiempo
- Animación de esqueleto (múltiples huesos por vértice)
- Interpolación de fotogramas
- Características fácilmente ampliables.
- Física dinámica: inercia, aceleración y fuerza.
- Cadencia automática de la escena en tiempo real.

· *Interfaz*
- Funciones fáciles de usar para determinar objetos seleccionados
- Funciones de ayuda para el movimiento de cámaras.
- Funciones de ayuda para la translación de objetos seleccionados.
- Funciones de ayuda para la conversión entre las coordenadas de las escena y del mundo, raycasting.

· *Sonido*
- Soporte built-in para fuentes y oyentes de sonido 3D
- Actualización automática de posición, velocidad y orientación de fuentes y oyentes.
- Componente de librería de muestras de sonido.
- Administrador de sonido para WaveOut, BASS y FMOD.

· *Utilidades*
- Funciones geométricas optimizadas y utilidades.
- Soporte de funciones y clases de spline cúbica
- Manipulación y optimizaciñon de malla.
- Componentes para hacer salva pantallas con todas las características en solo unos clicks.
- Determinación precisa de la velocidad del fotograma.
- Temporizador asíncrono (multihilo)
- Soporte para joystick
- Acceso asíncrono al teclado, soporte de mapa de teclas.

3. Como instalar GLScene

En esta sección nos centraremos en la instalación de GLScene en Delphi7, en versiones anteriores se realiza de forma similar.

Como ya se ha comentado, para obtener la última versión del GLSene se ha de acudir al CVS del proyecto. Existe también otra posibilidad, descargar la última versión completa de la sección de descargas de la página de sourceforge (versión 0.9.1 beta), opción totalmente desaconsejada, ya que se trata de una versión de 2003.

Aquí no explicare como se utilizan los repositorios, para más información sobre como descargar la versión CVS visitar la web de sourceforge, recomiendo utilizar la utilidad TotoiseCVS (o alguna similar).

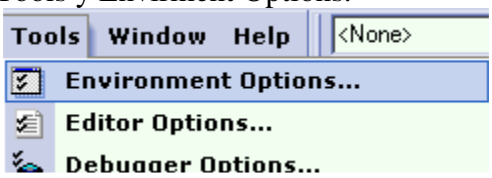
Una vez obtenida la última versión de la librería debemos tener con conjunto de carpetas:

- **BCB5**, **BCB6** y **BCBDemos** son para Borland Builder y no nos interesan.
- **Kylix**, ficheros para kylix, tampoco nos interesan.
- **Delphi4**, **Delphi5**, **Delphi6** y **Delphi7** donde se encuentran los ficheros para instalar la librería dependiendo de la versión de delphi (la versión 4 y no esta soportada por las nuevas versiones de la librería)
- **Demos**, con diferentes aplicaciones de demostración, muy útiles para aprender.
- **Help**, ficheros de ayuda de la librería (muy desfasados, ya que se actualiza constantemente)
- **Source**, ficheros fuente de la librería.
- **Utilities**, utilidades no esenciales

Aun que podemos tener la librería en cualquier directorio, aconsejo que esté dentro de del directorio donde tenemos instalado Delphi, en concreto las librerías, por ejemplo, si en nuestro sistema tenemos instalado Delphi en "c:\Archivos de Programa\Borland\delphi7" podemos poner los directorios de la librería en "c:\Archivos de Programa\Borland\delphi7\Lib\GLScene".

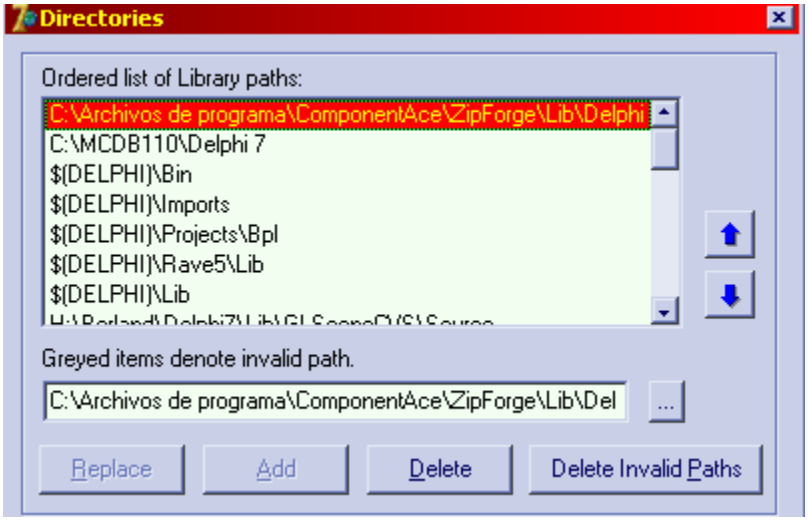
El primer paso para la instalación, una vez que tenemos los archivos de la librería en el directorio que queremos, es iniciar Delphi 7. Antes de instalar el paquete de componentes hemos de añadir las rutas en las cuales se encuentran los ficheros de la librería. Los pasos son los siguientes:

1- En el menú seleccione Tools y Envirment Options.



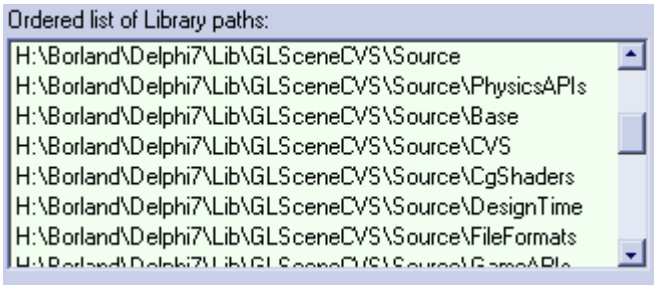
2- Aparecerá una ventana con varias pestañas, seleccione la pestaña Library.

3- Pinche sobre el botón al lado de Library path, y aparecerán la lista de directorios en los que delphi busca los ficheros de librerías.



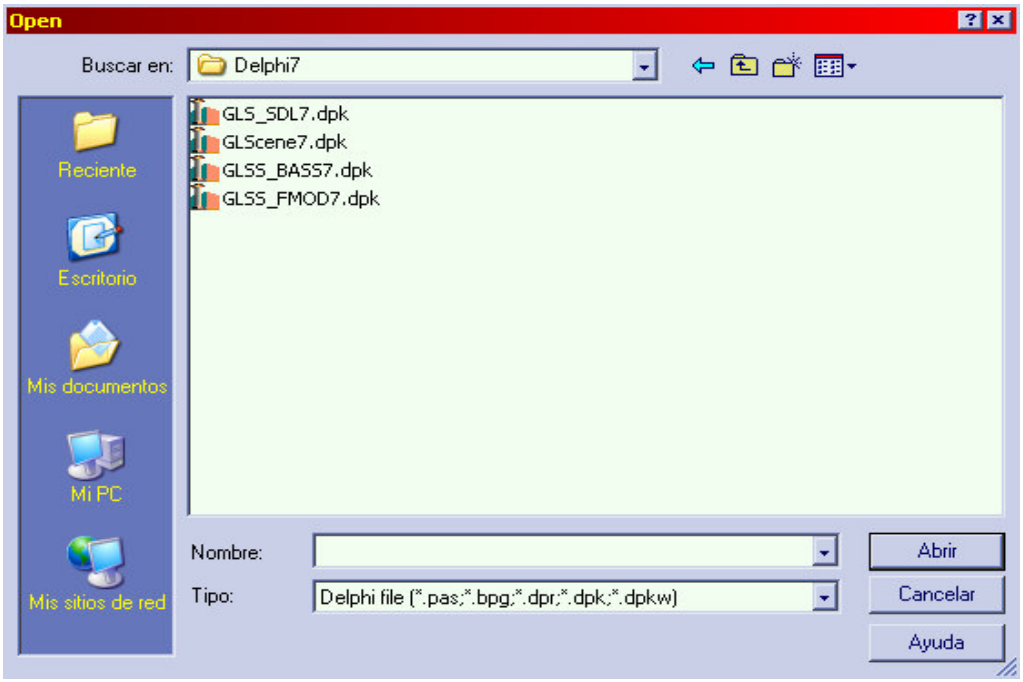
4- Pinche sobre el botón a la derecha de la ruta activa: 
5- Aparecerá una ventana con el árbol de directorios, seleccione el directorio donde se encuentran los ficheros fuentes ("...\GLSene\sources") y acepte.
6- Añada este directorio a la lista de paths presionando el botón Add: 

7- Realizar esto mismo para cada uno de los subdirectorios dentro del directorio sources, al final debe quedar una lista parecida a esta (depende del directorio en el que tengamos instalado Delphi):

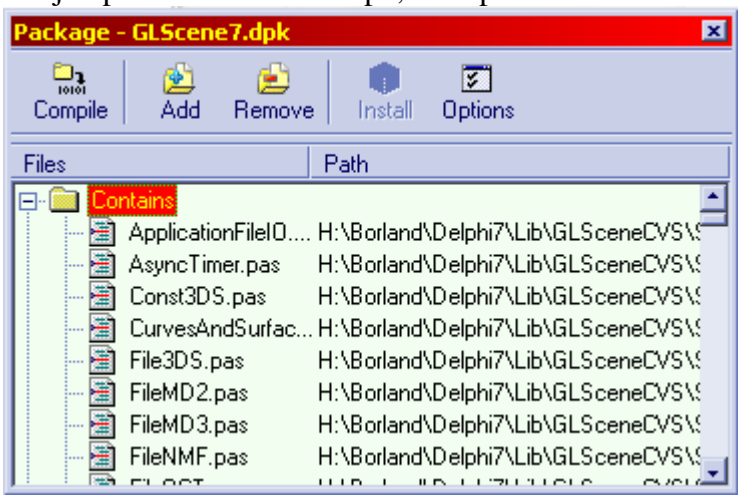


Una vez que ya tenemos las rutas de la librería en el listado de rutas procederemos a instalar los componentes:

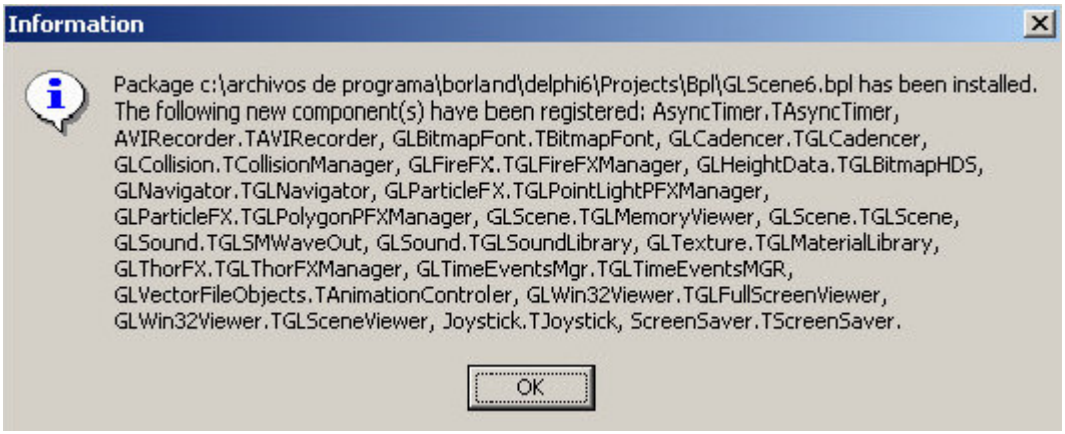
1- Cerremos cualquier fichero que tengamos abierto en Delphi (menú File, Close All)
2- Después abriremos cada uno de los ficheros de paquetes del directorio ..\GLScene\Delphi7 (menú File, Open)



3- Empecemos por ejemplo con GLScene7.dpk, nos aparecerá una ventana como esta:



4- Presionamos sobre el botón Compile, y luego en el botón Install. Una vez hecho esto te deberá aparecer un mensaje de confirmación, avisándote de los nuevos componentes instalados. Algo parecido a esto:



Si todo a ido bien ahora deberías tener en la barra de componentes cuatro nuevas pestañas, GLScene, GLScene Utils, GLScene Shaders y GLScene PFX, con todos los componentes que acabas de instalar. ¡Enhorabuena, ya puedes empezar a usar GLScene!

4. Entorno y componentes

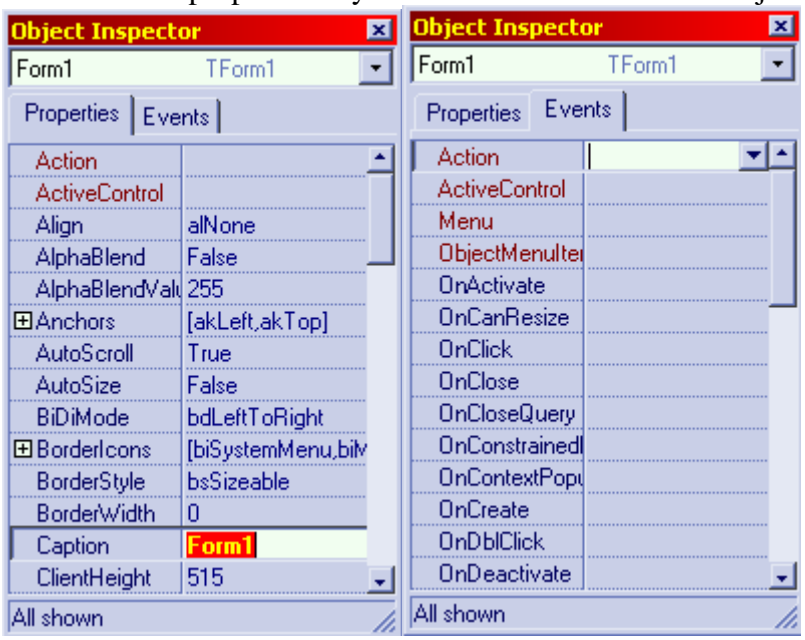
4.1 Entorno de desarrollo Delphi

Como se ha dicho anteriormente se da por hecho que el lector tiene conocimientos de programación en Delphi 7, no obstante se hará un pequeño recordatorio de la distribución del entorno de desarrollo y algunos elementos básicos que se han de conocer de Delphi.

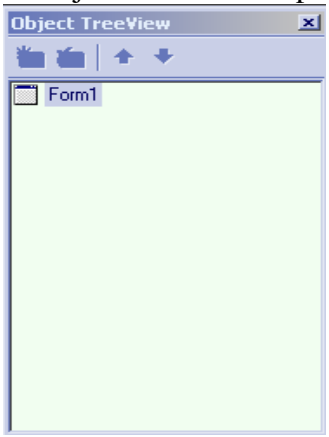
Delphi es un entorno de desarrollo que utiliza un lenguaje de programación de alto nivel orientado a objetos con sintaxis muy similar al pascal. Delphi posee un entorno parecido al de Visual Basic, completamente visual, pero apoyándose en el lenguaje Object Pascal (Pascal orientado a objetos), más potente que el basic.

Los elementos básicos del entorno de trabajo de Delphi son:

Object Inpector
En esta ventana tenemos las propiedades y eventos de cada uno de los objetos.

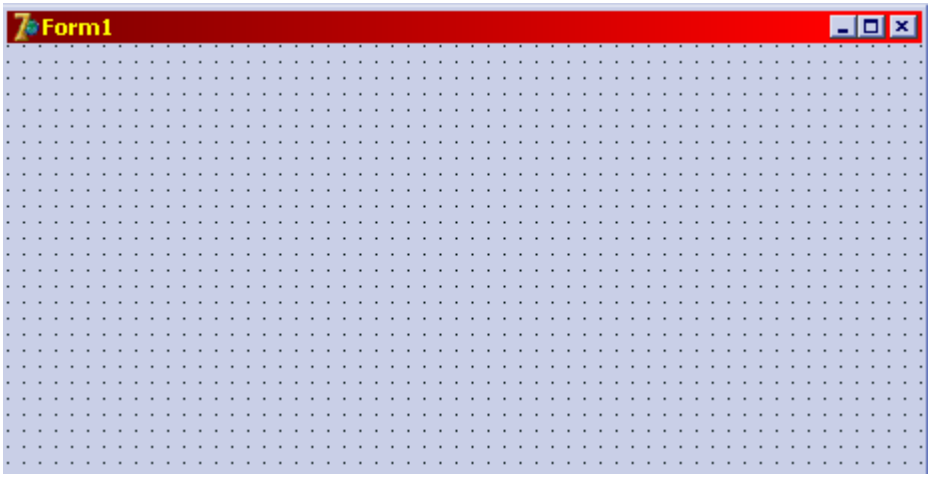


Object TreeView
Ventana donde tenemos el árbol de objetos de nuestra aplicación

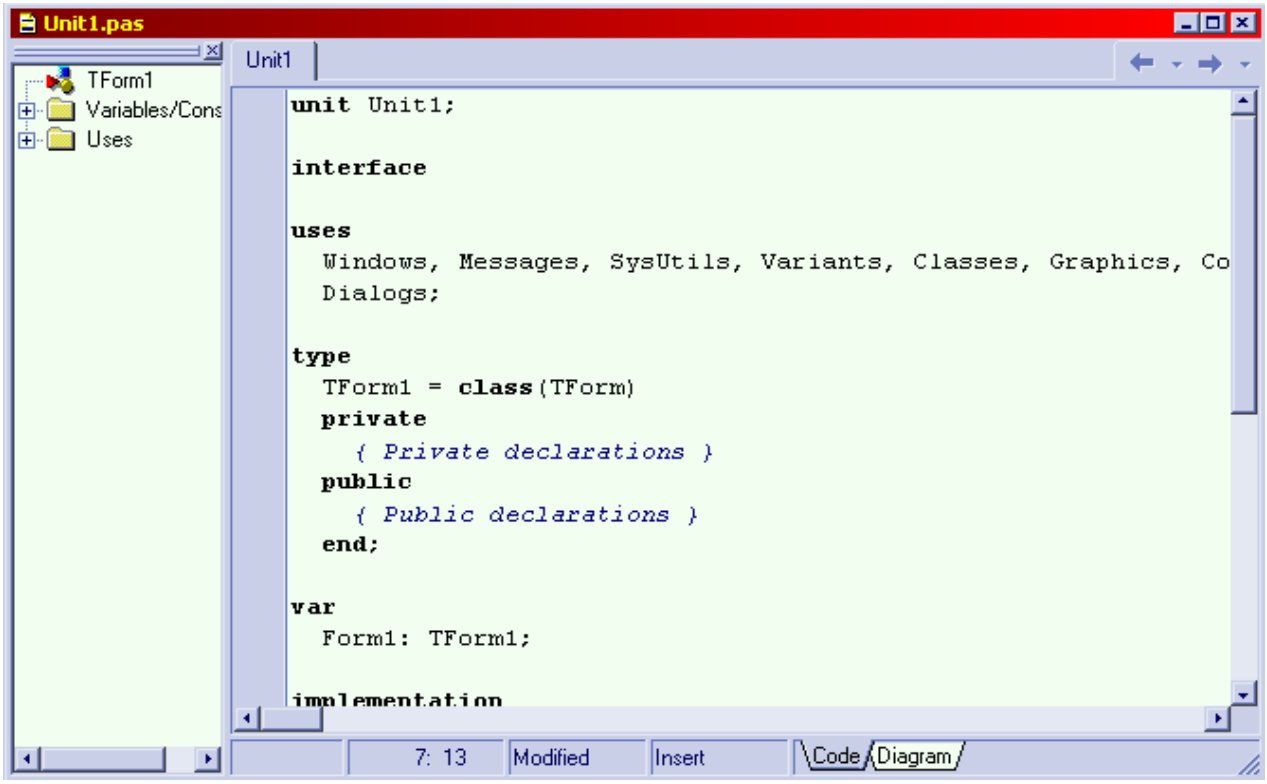


Barra de componentes
En esta barra, compuesta por multitud de pestañas, se encuentran los componentes.

Aplicación
Ventana de la aplicación que estamos realizando (también llamada formulario, de 'Form')



Ventana de Código
Donde introduciremos el código de nuestra aplicación.



4.2 Componentes principales de GLScene

Después de haber instalado GLScene en delphi aparecen en la barra de componentes cuatro nuevas pestañas, GLScene, GLScene Utils, GLScene Shaders y GLScene PFX. La primera contiene el conjunto de componentes mas importantes de GLScene, la segunda son componentes de apoyo como componentes para hacer salvapantallas, administrador de colisiones, componentes para el uso de Joystick, etc. GLScene Shaders contiene componentes para la gestión de las sombras.

A continuación se enumeran los componentes principales con una pequeña descripción, se podrán ver más detalladamente en los ejemplos prácticos, donde realmente se puede enseñar el uso de cada componente.

- **GLScene**: es el componente principal de la librería, dentro de el se almacenara la escena en si. El primer componente a añadir a nuestra aplicación.
- **GLSceneViewer**: este es el único componente visual de GLScene, en el se muestra la escena, desde el punto de vista de la cámara asociada.
- **GLMemoryView**: parecido al GLSceneViewer, pero a diferencia de este la escena es almacenada en memoria.
- **GLMaterialLibrary**: aquí se almacenan los materiales que usemos en nuestra escena, de forma que puedan administrarse fácilmente y poder ser reusados en diferentes objetos.
- **GLCadencer**: es el 'reloj' de nuestra escena, marca los tiempo en las animaciones, eventos y acciones que se produzcan.
- **GLPolygonFXManager**, **GLPointLightPFXManager**, **GLFireFXManager**, **GLThorFXManager**: estos componentes se usan para generar efectos definidos, totalmente configurables, como efectos de fuego, rayos, etc.
- **GLSoundLibrary**: librería con muestras de sonido.
- **GLSMWaveOut**: componente para la salida de sonido.
- **GLFullScreenViewer**: muy parecido a GLSceneViewer, pero a diferencia de este se usa para ver la escena a pantalla completa.
- **BitmapFont**: componente para crear y cargar fuentes para escribir en nuestra escena.
- **AnimationController**: Componente para el control de la animación de personajes.
- **AVIRecorder**: para grabar video en formato AVI de lo que suceda en nuestra

escena (tiene asociado un componente GLSceneViewer, que es la vista que se graba)

· **GLBitmapHDS**  , **GLCustomHDS**  , **GLHeightTileFieldHDS**  y **GLBumpMapHDS**  son componentes para crear escenarios a base de mapas de alturas.

· **CollisionManager**  : como su nombre indica es un gestor de colisiones.

· **Joystick** 

· **ScreenSaver**  : componente para crear salva pantallas.

· **GLNavigator**  : gracias a este componente es muy fácil moverse por la escena

· **GLUserInterface**  : como el nombre indica, es un componente para gestionar la interfaz con el usuario, tiene relación con el GLNavigator

Estos son los componentes más destacados, de los cuales solo utilizaremos 5 o 6 en este manual.

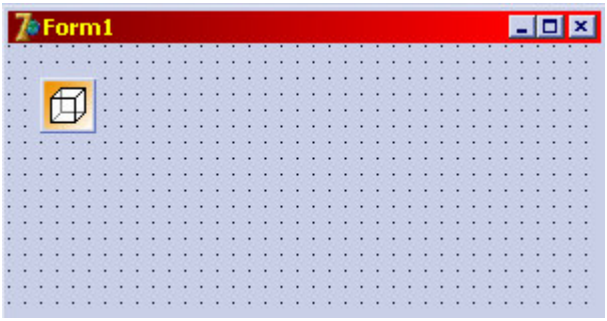
5. Primeros pasos

5.1 Ejemplo básico: El cubo

Una vez instalado GLScene y con una versión general de sus componentes principales empecemos con un ejemplo práctico de su uso. En este primer ejemplo vamos a crear una escena muy simple, con un único cubo en el centro, una cámara que apunte hacia él y una luz para iluminar la escena.

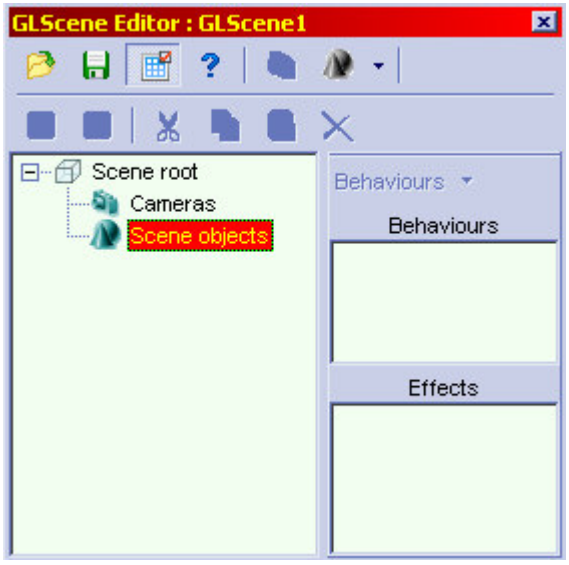
1- Creamos una nueva aplicación (menú File/New/Application)

2- Vamos a la barra de componentes, a la pestaña GLScene y pinchamos sobre el primer componente "GLScene"  y luego sobre nuestra aplicación. Debería quedarnos algo así:



3 - Ahora pinchamos sobre GLSceneViewer  y lo añadimos también sobre nuestra aplicación, dándole el tamaño que deseemos. Dentro de este componente visual será donde se muestre la escena.
Los objetos por defecto se llamaran GLScene1 y GLSceneViewer1, si queremos que GLSceneViewer1 ocupe toda la ventana de la aplicación lo seleccionamos, y en la ventana de propiedades (Object Inspector) ponemos en la propiedad 'Align' el valor 'Client'.

4- Una vez incluidos estos dos componentes en nuestra aplicación hay que montar nuestra escena. Pinchamos dos veces sobre el objeto GLScene1 que hemos creado en nuestra aplicación, se abrirá una ventana con la que editaremos nuestro escenario en tiempo de diseño.



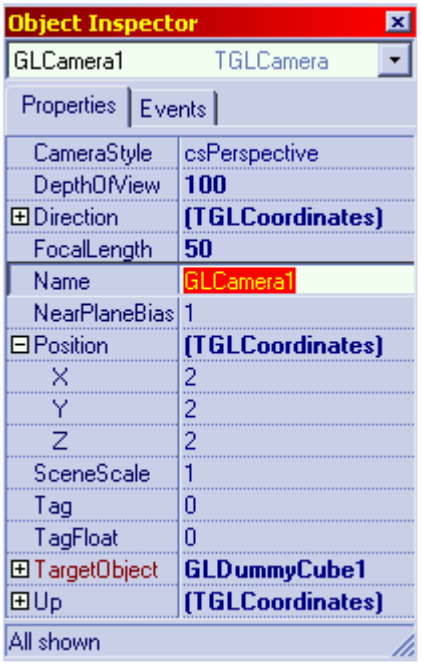
5- Seleccionamos 'Cameras' en el editor y pinchamos con el botón derecho, en el menú emergente pinchamos sobre 'Add camera'

6- Ahora repetimos el paso anterior pero seleccionando esta vez 'Scene Objects', y en el menú emergente 'Add Object' y 'DummyCube'. Esto añadirá un dummy cube a nuestra escena (es un cubo solo visible en tiempo de diseño)

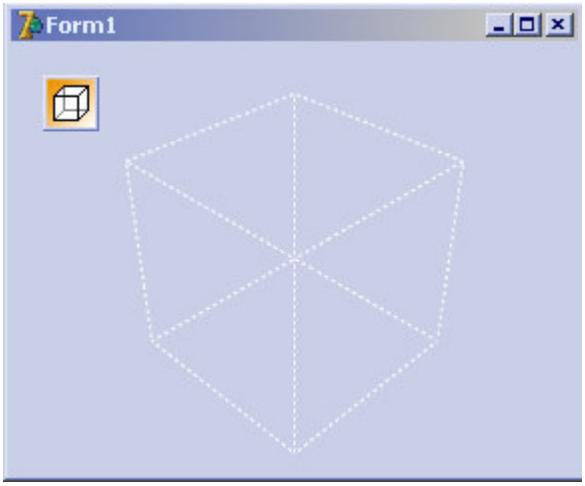
7- Para poder ver la escena en nuestro GLSceneViewer tenenemos que asignarle la cámara que acabamos de crear. Seleccionamos el objeto GLSceneViewer1 en nuestra aplicación (o desde la ventana 'Object TreeViewer'), y en la propiedad cámara (ventana 'Object Inspector', pestaña propiedades) le asignamos la cámara 'GLCamera1', que es la

que acabamos de crear en nuestra escena.

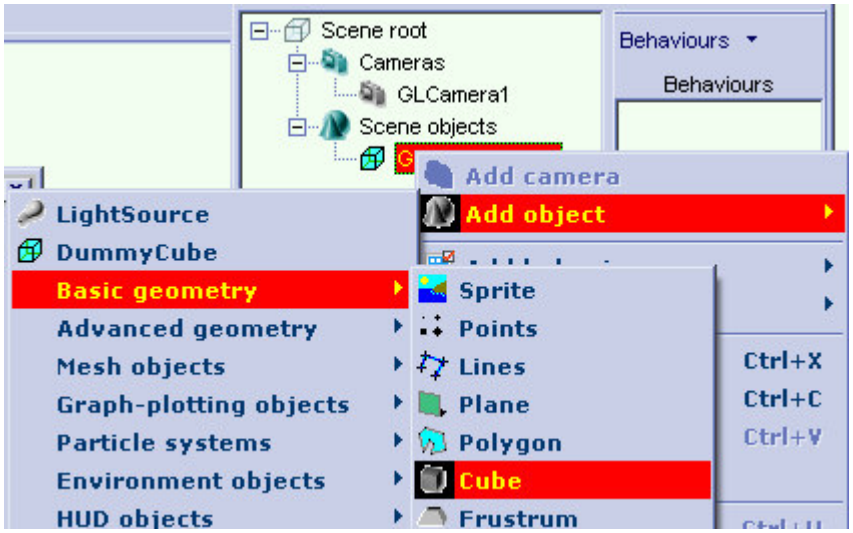
8- Ahora situamos la cámara, que inicialmente esta en la posición (0, 0, 0), en una posición adecuada. Seleccionamos 'GLCamera1', y en la propiedad 'position' le ponemos el valor en 'x', 'y' y 'z' de 2. Y por último en la propiedad 'Target Object' ponemos el valor de 'GLDummyCube1', para que la cámara apunte hacia este objeto.



Después de estos pasos ya podemos ver el dummy cube de nuestra escena, y la aplicación queda así:

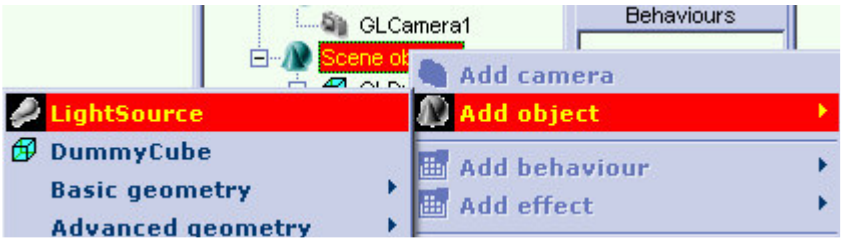


9- Añadamos ahora el cubo que vamos a ver en nuestra escena, vamos al editor de la escena, seleccionamos DummyCube1 y pinchamos con el botón derecho, en el menú seleccionamos 'Add Object', 'Basic Geometry' y Cube.

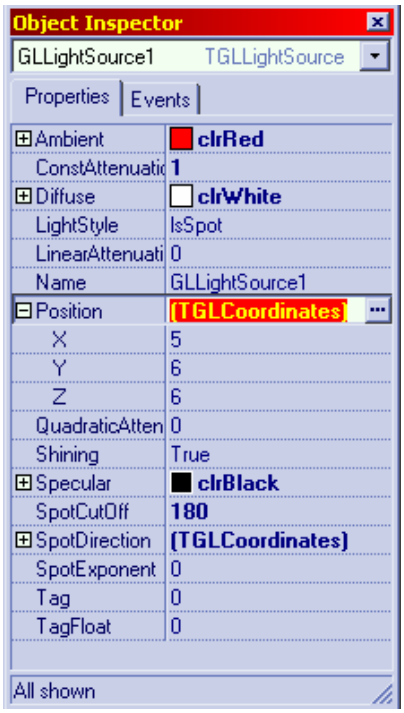


Con esto ahora tendremos un cubo "colgando" dentro de nuestro Dummy cube en el editor, y podremos verlo en la aplicación.

10- Para no ver el cubo negro debemos añadir una luz. Sobre 'Scene Objects' añadimos 'LightSource'



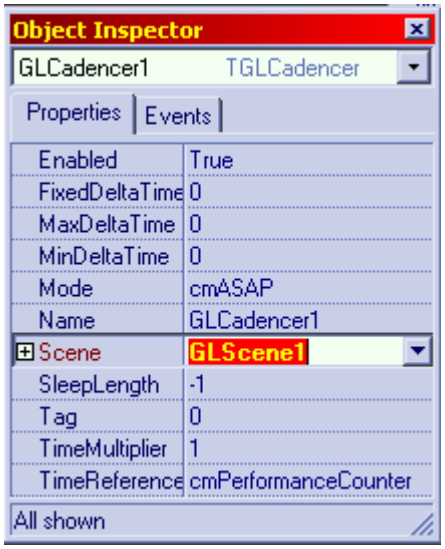
En las propiedades de la luz que acabamos de crear cambiamos la posición a (5, 6, 6) por ejemplo, y el valor Ambient a 'clrRed'.



5.2 Animemos nuestro cubo

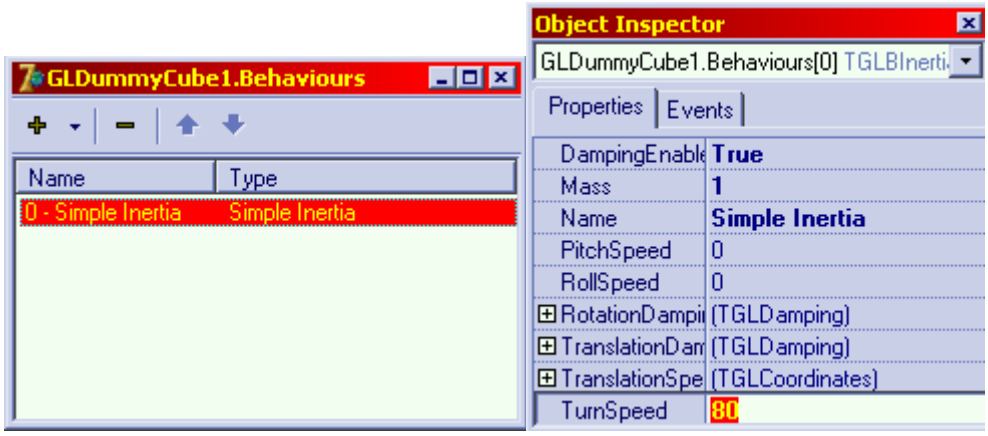
Ya tenemos el cubo con un color rojizo (debido a que la componente ambient de la luz es roja). Si le damos a Play  podemos ver nuestra aplicación en ejecución. Pero es muy estática, así que vamos a animarla un poco, para ello sigamos los siguientes pasos:

1- Añadamos a la aplicación el componente GLCadencer  , y en la propiedad Scene ponemos el valor GLScene1. Es necesario añadirle este componente a nuestra escena para poder realizar animaciones.



2- En las propiedades del DummyCube pinchamos al lado de 'Behaviours', y aparecerá una ventana como esta:

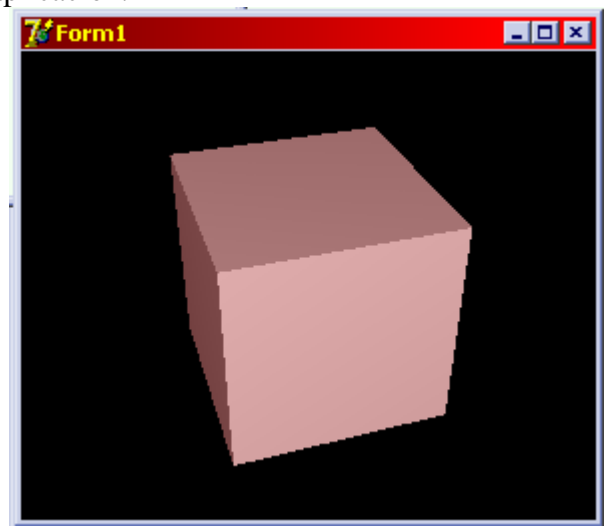
Pinchamos sobre el '+' y seleccionamos 'Simple Inertia', en las propiedades de 'Simpe Inertia' ponemos 'TurnSpeed' a 80



3- Ahora en la ventana de código añadimos en 'Uses' la librería 'GLBehaviours'

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, GLScene, GLObjects, GLMisc, GLCadencer, GLWin32Viewer,
  GLBehaviours;
```

4 - Ya está la animación acabada. Ahora ponemos el fondo negro, en las propiedades de GLSceneViewer1 en 'buffer', 'backgroundcolor' le damos el valor 'clrBlack' y ya solo queda ejecutar la aplicación.

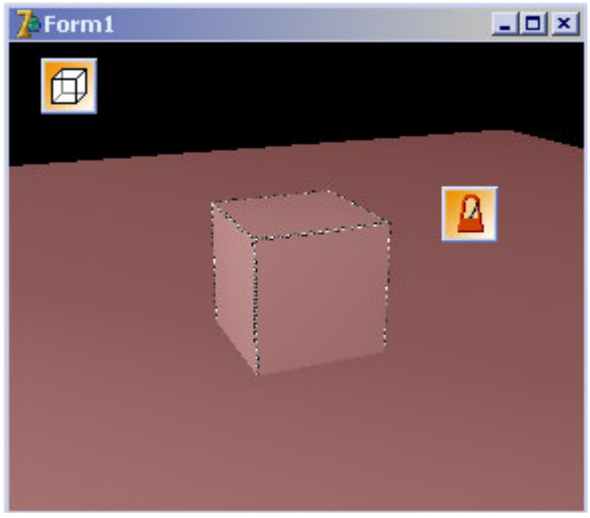


5.3 Texturas y sombras

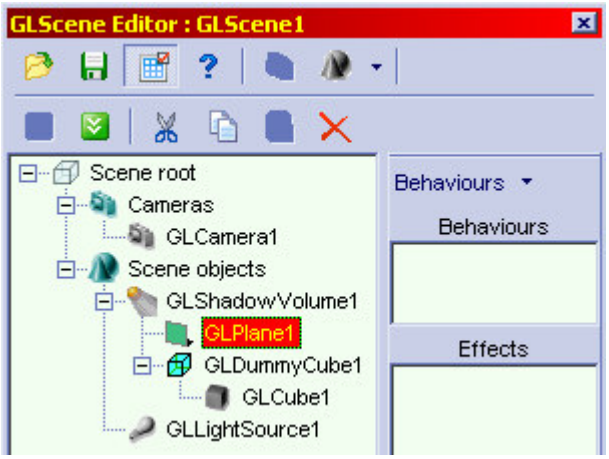
Vamos ahora a hacer una escena algo más realista, añadámosle un suelo sobre el que estará el cubo y que haga sombra sobre este. Después le daremos una textura tanto al suelo como al cubo, usando para ello la librería de materiales.

1- Añadir a 'Scene Object' un objeto 'plane' (dentro de la geometría básica, al igual que el cubo), este plano hará de suelo. Le daremos las siguientes propiedades:
   - Position: x = 0, y = -0,5 , z = 0
   - height = 15 (alto)
   - with = 15 (ancho)
   - pichangle = 90 (rotación en grados)

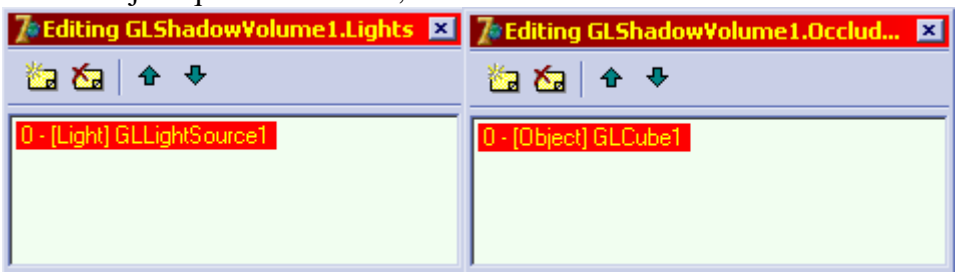
2- Alejemos un poco la cámara, por ejemplo a la posición (5, 2, 2), nos debería quedar algo así:



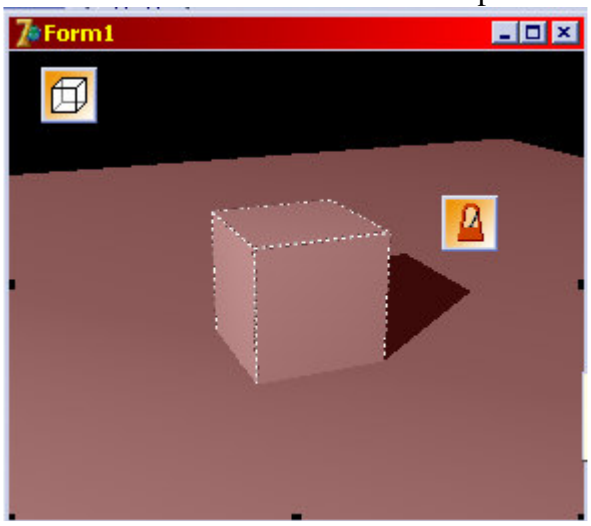
3- Para añadir la sombra añadiremos un objeto llamado 'ShadowVolume' que se encuentra en 'SpecialObjects' y arrastrando con el ratón pondremos el plano y el dummy cube colgando de este objeto.



4- En las propiedades del objeto GLShadowVolume1 que acabamos de crear modificamos svoScissorClip=False, svoWorldScissorClip=True, svoDesignVisible=True. En la propiedad 'Lights' añadimos la luz GLLightSource1, y en la propiedad 'Occluders' añadimos el objeto que hara sombra, en este caso GLCube1.

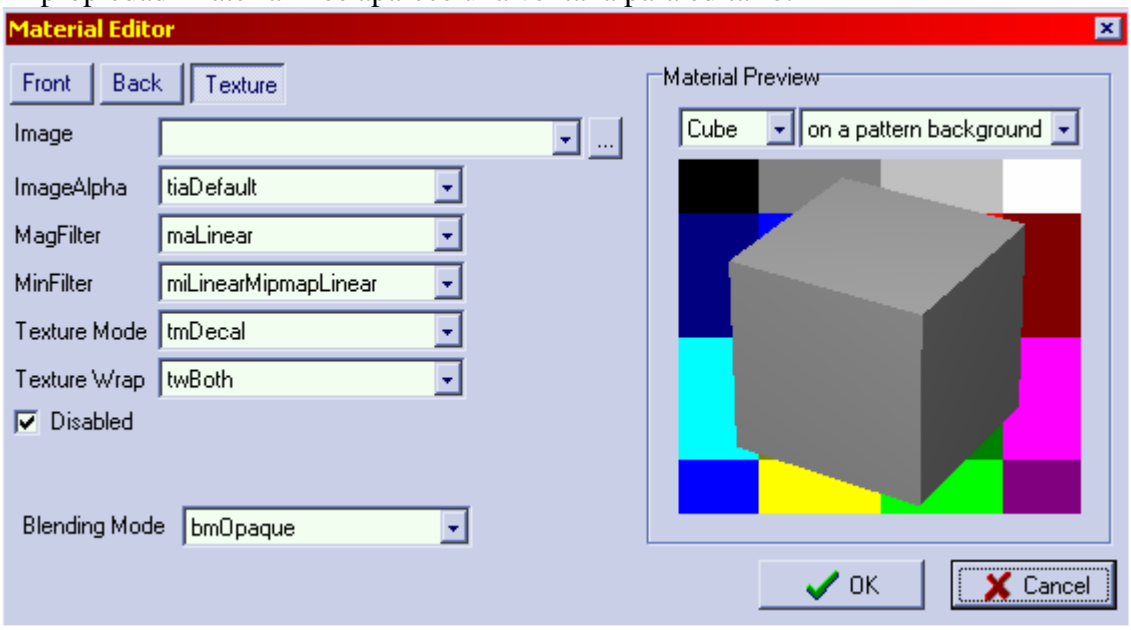


5- Ahora para poder ver la sombra necesitamos modificar una propiedad del objeto GLSceneViewer1 que define el contexto de renderizado, en concreto, dentro de buffer/ContextOptions la opción 'roStencilBuffer' hay que ponerla a true. Una vez hecho esto ya podremos ver la sombra del cubo sobre el plano.

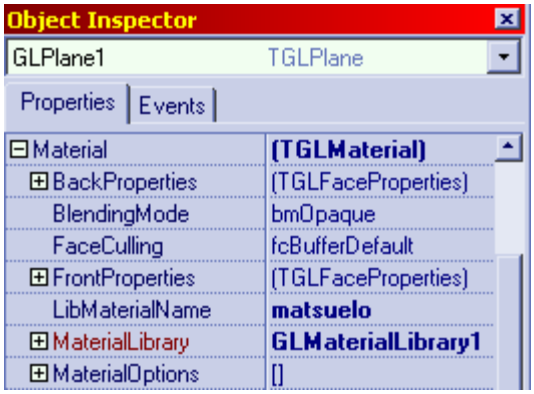


6- Una vez tenemos la sombra pongámosle textura tanto al cubo como al plano. Primero añada el componente GLMaterialLibrary  a la aplicación. Y en la propiedad 'material' del GLMaterialLibrary1 que acabamos de crear añadamos un material, que llamaremos "matsuelo".

En matsuelo cambiemos la propiedad 'Material', al pinchar en la parte derecha de la propiedad 'Material' nos aparece una ventana para editarlo.



Seleccionamos Texture y en image ponemos la imagen que queremos que sea la textura del suelo, en nuestro caso simula ser hierva, después en la opción Texture Mode elegimos la opción 'tmModulate'. Le damos a OK y ahora vamos a las propiedades del plano, desplegamos las propiedades de 'Material' y en 'MaterialLibrary' seleccionamos MaterialLibrary1, después en 'LibMaterialName' el nombre del material que acabamos de crear, es decir "matsuelo".



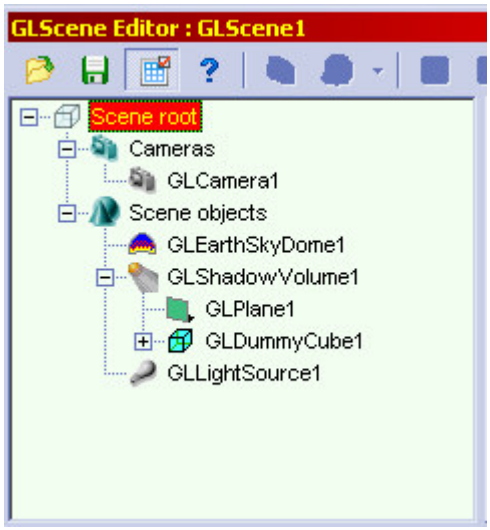
Repitamos esto mismo para el cubo, creamos otro material en la librería, en las opciones del cubo indicamos que vamos a usar la librería MateriaLibrary1, y seleccionamos el material que hemos creado en 'LibMaterialName'. MUY IMPORTANTE: si los ficheros de textura que usamos están en formato jpg tendremos que añadir en el código la librería 'jpeg' (en Uses), sino no cargara las texturas.

7- Para que la textura del suelo quede mas realista modifiquemos la propiedad xtiles y ytiles, poniendo ambas a 10, y para que la imagen sea más clara cambiar el componente ambient de la luz a 'clrOldGold'. Al final nos queda:

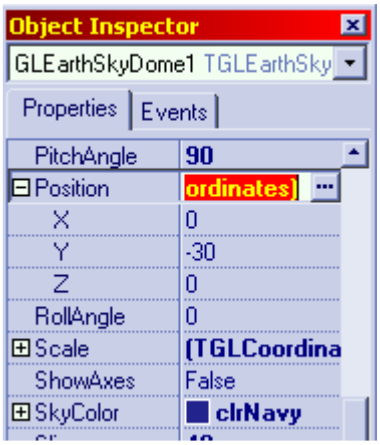
5.4 Objetos de entorno y efectos

Nuestro siguiente paso será quitar ese fondo negro para intentar hacer la escena un poquito más realista. Intentaremos simular un cielo, con un pequeño degradado en el horizonte. Los pasos a seguir son los siguientes:

1- En el editor de escenario, en 'Scene objects' añadir el objeto 'EarthSkyDome', que se encuentra en 'Enviromen objects'. Después hay que arrastrarlo con el ratón para colocarlo el primero en la lista de objetos (por motivos de prioridad en el renderizado).

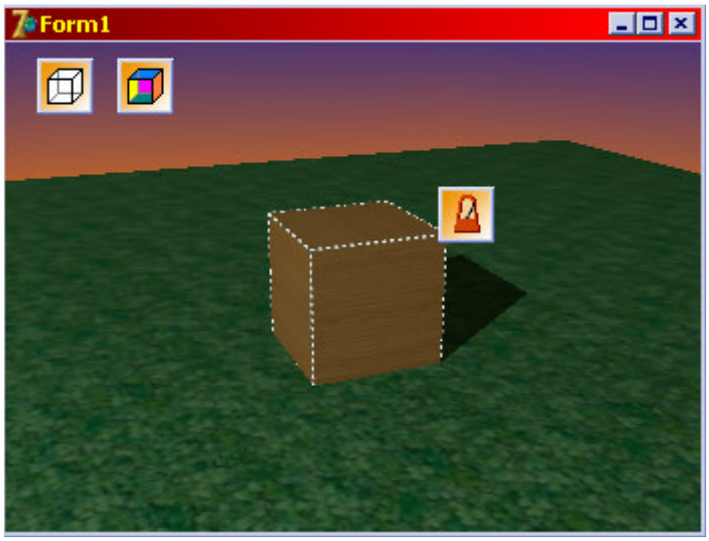


2- Una vez hecho esto podremos apreciar el fondo azul tras nuestra escena. Ahora modifiquemos las propiedades del cielo:
- PitchAngle = 90 (para colocarlo en la orientación que toca)
- Position = (0, -30, 0)
- SkyColor = clrNavy (o el que queramos)



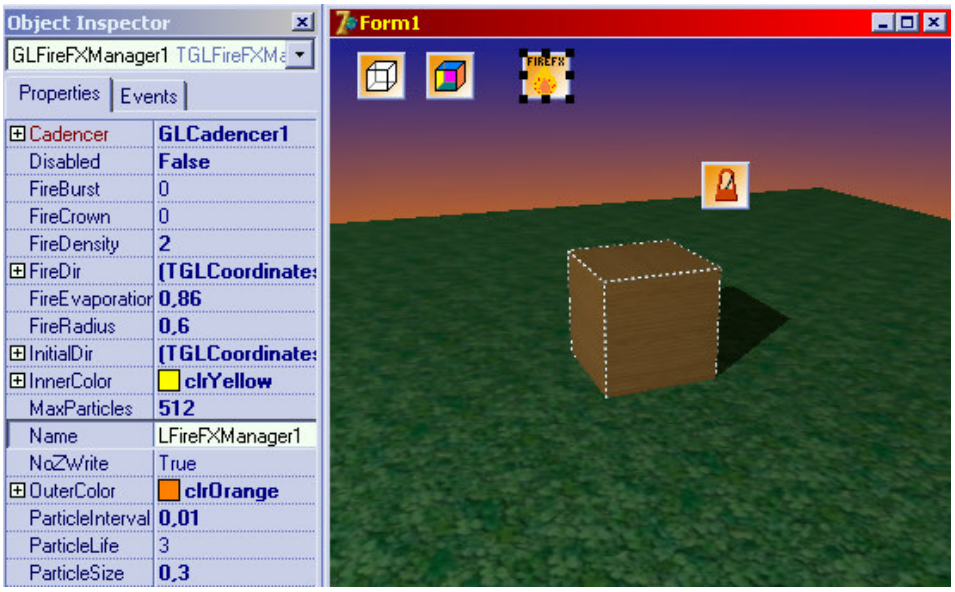
3- Ahora añadámosle al cielo unas bandas, para hacerlo menos uniforme. Cliquemos sobre la propiedad 'Bands', aparecerá una ventana en la que podremos añadirle bandas de color al cielo. Añadamos una banda con los valores: StopAngle = 20; StartColor = clrOrange; StopColor = lrNavy.

Con esto conseguimos un cielo parecido al de un atardecer (no demasiado logrado, que cada uno ponga los valores que prefiera). Se pueden añadir mas bandas de degradados simplemente definiendo rangos de ángulos, los ángulos no definidos toman el valor de SkyColor.

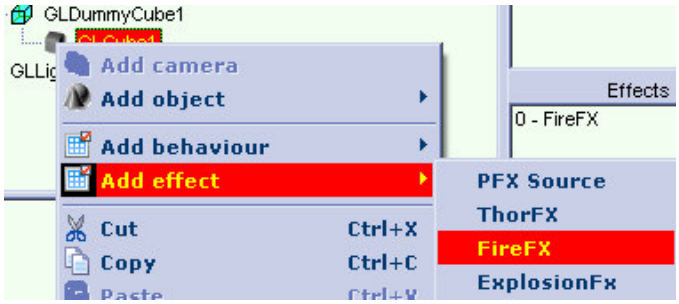


Ahora añadámosle a la escena un efecto especial, vamos a quemar el cubo. Para hacer esto añadiremos un componente GLFireFXManager, lo configuraremos, le añadiremos un efecto especial de FireFX al cubo y le asociaremos el FireFXManager, vamos con los pasos detallados:

1- Hay que añadir el componente GLFireFXManager a nuestra aplicación, y modificar sus propiedades. En la propiedad Cadencer seleccionamos GLCadencer1, en FireDensity configuramos la densidad del fuego, una densidad de 2 por ejemplo, ya que se trata de un objeto grande. En la propiedad FireRadius indicamos cual es el radio máximo en el cual aparecerán las partículas que simulan el fuego, al ser la caja de 1 de lado, 0,5 si se cuenta desde el centro, introduciremos un poco más para que parezca el que fuego sale de as paredes de la caja, por ejemplo 0,6 esta bien. En MaxParticles ponemos 512 ya que ha de ser un fuego grande, en ParticleInterval 0,01 y ParticleSize 0,3. Todos estos valores, así como el color de las llamas pueden ser configurados a gusto de cada uno.



2- Después de configurar el efecto hay que asignárselo a la caja, para ello seleccione GLCube1 en el editor de la escena, pinchar sobre el con el botón derecho y seleccionar 'Add Effect' y 'FireFX'. Aparecerá en la lista de efectos el nuevo efecto que hemos introducido, lo seleccionamos y en las propiedades de este indicamos como 'Manager' al objeto GLFireFXManager.



Con esto ya esta el efecto del fuego asociado a la caja, solo hay que ejecutar la aplicación para ver el resultado.