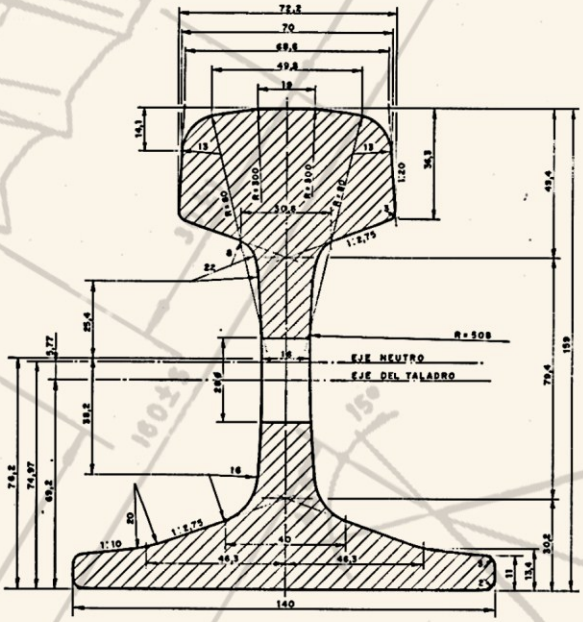
TUTORIAL: PROFUNDIZANDO EN 3DS MAX Y RAILWORKS

TS11M003



Pere Comas

Abril de 2011

trensim.com

Presentación

Este tutorial es continuación de aquellos denominados "Tutorial para iniciarse en 3ds Max y RailWorks" y "Tutorial avanzado en 3ds Max y RailWorks", y pretende ser una profundización de conocimientos para quien desea usar técnicas especiales en el modelado para RailWorks.

Se recomienda la lectura y ejecución de los tutoriales mencionados para quienes no tengan experiencia en el manejo de 3ds Max, pues conceptos ya desarrollados en ellos no volverán a describirse. En esta ocasión se incidirá en la explicación de los shaders especiales que provee RailWorks para efectos especiales.

El Tutorial presenta una visión personal del programa de diseño y de la experiencia en su manejo.

Por último tampoco quiero olvidar, que aunque el tutorial se presenta narrado en primera persona, éste es fruto de la colaboración y experiencias de varias personas:

- javierfl
- LBA
- blas_dani
- jjlor
- Divi4p
- y Marc

Así como la inestimable dedicación en el testeo del mismo de:

- edsolis
- milcien
- y Francesc SV

Contenido del Tutorial

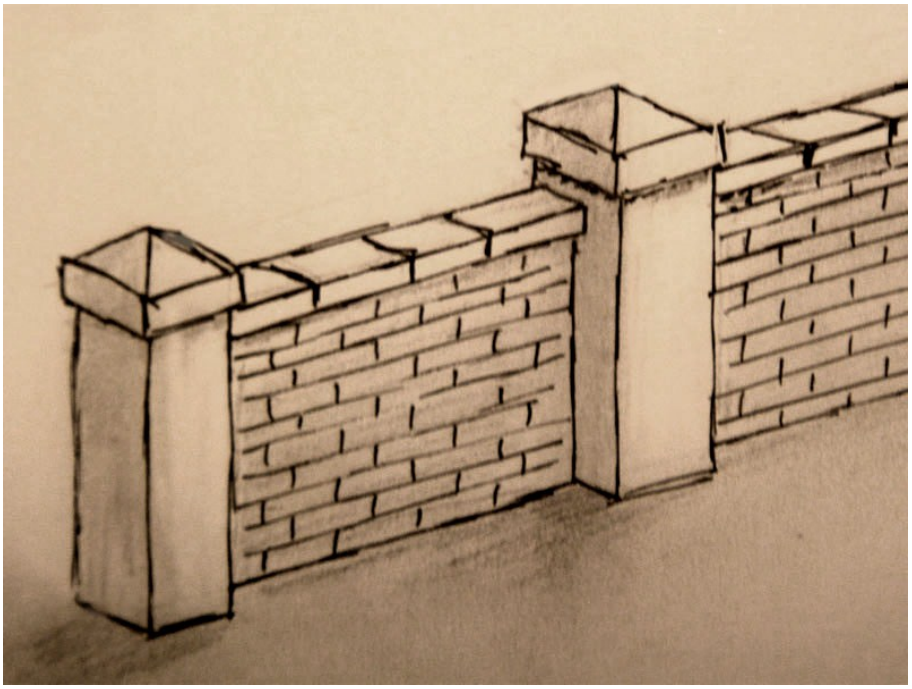
1. Objetos procedurales

Un objeto procedural es un elemento 3D del cual tan solo se informa al simulador de su perfil, procediendo el motor de la simulación a generar el volumen del objeto por extrusión de dicho perfil, ajustando dicho desarrollo a un perfil recto, curvo o mixto que se determina en el momento de situarlo en una escena del simulador (o lo que viene a ser una ruta).

Un objeto procedural se puede completar con un elemento fijo para cada uno de sus extremos, y otro objeto fijo que se puede repetir a lo largo del perfil de desarrollo a espacios regulares predeterminados.

Típicamente, con estos elementos, se construyen: Muros, vallas, andenes, carreteras, caminos, riachuelos, acequias, vías, catenarias, puentes, terraplenes, desmontes, etc...

En la presente ocasión vamos a reproducir un muro, con algún elemento decorativo adicional. Veamos una representación de lo que pretendemos obtener:



El muro, conceptualizado en el dibujo adjunto (no necesariamente a escala), estará formado por una pared de ladrillo, rematada en su parte superior por unas losas de piedra. Estará enmarcado y sustentado por unas columnas de hormigón, rematadas también en hormigón por una pirámide achatada de base cuadrada.

El muro es puramente imaginario, y la mezcla de materiales de construcción: ladrillo, piedra y hormigón, se ha forzado para mostrar las posibilidades de texturado. No obstante, este elemento nos permite explorar las diferentes posibilidades de los objetos procedurales, y, por tanto, permitirá entender las claves para la elaboración de cualquier otro elemento similar que se desee recrear.

1.1. Texturas

Para este ejemplo no se ha buscado un muro existente, al cual se podría haber fotografiado para obtener así sus texturas, por preferir mostrar otra forma de obtener texturas.

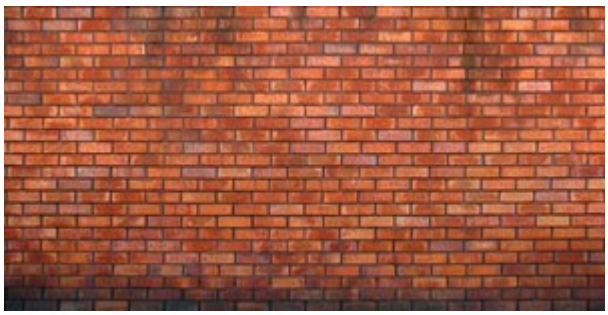
Este apartado que dedicaremos a las texturas no es específico de los objetos procedurales, pero aprovecharemos la oportunidad para introducir las posibilidades de la web CGTextures <http://www.cgtextures.com>. Como dice el manifiesto de intenciones de su autor, Marcel Vijfwinkel:

> *CGTextures se esfuerza por ser el mejor sitio de texturas. Siendo yo mismo un artista 3D, sé lo difícil que es crear texturas de materiales sin una buena foto. La realización de texturas debe ser una tarea de creatividad, no de pasar horas buscando la imagen apropiada en Internet. Espero que mediante el suministro de texturas de buena calidad, de una manera organizada, su trabajo cotidiano sea más fácil y más agradable.*

Procederemos a la localización de las texturas que nos parezcan más apropiadas para el caso que nos centra. En concreto necesitaremos:

- Una textura de ladrillos para el muro.
- Un textura de losas de piedra para el remate superior del muro.
- Una textura de hormigón para los soportes del muro.

Accedemos a la web CGTextures y observamos que en la página principal ya se nos muestra las diferentes tipologías de texturas que allí se pueden encontrar. La tercera de las tipologías, **Brick**, nos llama la atención, pues se trata de texturas de ladrillos. Pulsamos sobre ella y observamos que en la web las texturas de ladrillos están clasificadas en 7 subcategorías, de las cuales nos decantaremos por **Modern Small**. Esta tipología nuevamente se divide en hasta 13 subcategorías, de ellas nos puede interesar la primera, **Brown**. Aquí encontramos más de 200 texturas diferentes de ladrillos modernos pequeños marrones, cualquiera de las cuales nos puede interesar, pero a mí me llama la atención la primera de la última fila de la página, sobre la cual procedemos a pulsar con el ratón.



Observamos que esta textura se denomina BrickSmallBrown0064, es una contribución de Jacobo Cortés Ferreira y la tenemos disponible en tres resoluciones posibles. Elegiremos la menor de 640x320, pues será suficiente para nuestro propósito.

Retrocedemos hasta el menú de la categoría **Brick** para esta vez acceder a la subcategoría **Modern Large** donde elegiremos el tipo **Blocks** y nos fijaremos en la sexta de las texturas que aparecen, de la que usaremos una de las hileras de bloques para texturar las losas de piedra que rematarán el muro de ladrillos. En esta ocasión la textura está marcada con la palabra SET, ello indica que no se trata de una textura aislada, sino de un conjunto de texturas (dos o más) sobre un mismo tema u objeto.

Al acceder a la textura observamos que se denomina BrickLargeBlocks0007, es una contribución del autor de la página web, se trata de un conjunto de dos fotografías del mismo muro y están disponibles en tres resoluciones. Nos interesará la segunda de las texturas del conjunto, puesto que deseamos dar un acabado más sucio al muro, y nos bastará con la resolución más baja.

Por último buscaremos una textura de hormigón para las columnas. Regresaremos a la página principal de la web y seleccionaremos la categoría **Concrete** y la subcategoría **Bare**. La séptima textura, en la tercera fila, nos parece que bien pudiera adecuarse a nuestro modelo, más en esta ocasión la textura, además de estar marcada con la palabra SET, también está marcada con la palabra TILED, ello indica que el autor de la textura la ha adecuado para que esta pueda ser acoplada en mosaico con el fin de recubrir un área mayor, si es necesario, sin que se produzcan líneas de unión entre repeticiones de la textura.

Accedemos a ella y observamos que se denomina ConcreteBare0022 y que la tercera del conjunto es la que nos parece más apropiada. Por tanto la descargaremos como las anteriores.

Con estos archivos descargados vamos a preparar las texturas para el modelo. En particular, crearemos dos archivos de textura:

- Uno para texturar la columna (el elemento fijo de nuestro muro).
- Y otro para el muro de ladrillo y el remate de piedra (el elemento procedural del muro).

Para el primero usaremos la textura de cemento, que no necesita demasiada preparación. Bastará escalar dicha textura a un tamaño de 256x256, pues el original quizás es excesivo, y la guardaremos en formato .ace con un nombre que nos sea significativo. Podemos nombrarla según su contenido (p.e. TxCemento01.ace) o bien según su uso (p.e. TxMuroTut01.ace).

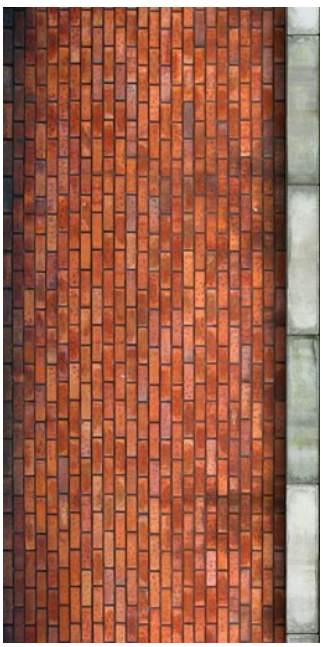
TxMuroTut01.ace

Para el segundo crearemos un archivo de 512x256 píxeles (512 de ancho y 256 de alto). En él copiaremos en la parte alta una hilera de bloques de la textura de bloques de piedra, que deberá ocupar el 10% de la hoja de textura en altura (26 píxeles) y todo el ancho. El resto, la zona inferior de 512x230, la ocupará la textura de ladrillos.

Importante:

El conjunto lo deberemos rotar 90º, pues los objetos procedurales toman la repetición de la textura, a lo largo del elemento generado, en el eje Y de la misma (norte-sur).

El resultado será similar a la imagen adjunta y la denominaremos TxMuroTut02, en formato .ace.



TxMuroTut02.ace

Como ya se indicó en su momento, crearemos una estructura de directorios para nuestro trabajo. Por tanto, deberemos crear una estructura de directorios, dentro de la carpeta **Source** del RailWorks, del tipo:

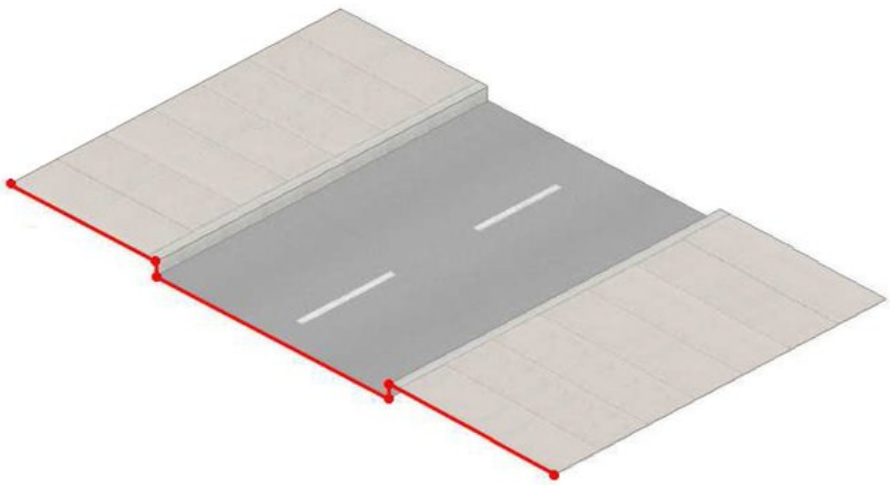
```
----Source
    |
    +---Pere (o el nombre de creador que adoptemos cada uno)
        |
        +---Renfe (o quizás "Objetos", porque éste es un
            |    elemento escénico independiente de la compañía)
            |
            +---Scenery
                |
                +---Procedural
                    |
                    +---ES_Muro_Ladrillo
                        |
                        +---Textures
```

y colocaremos en el directorio "Textures" los archivos de textura generados. Para más información podéis repasar el capítulo "*5. Guardando el trabajo - Estructura de directorios para RW*" del "Tutorial para iniciarse en 3ds Max y RailWorks".

1.2. El componente procedural. Shader LoftTexDiff.fx

Ya podemos abrir el 3ds Max, y empezaremos por construir en primer lugar el elemento que constituirá el componente procedural de nuestro muro.

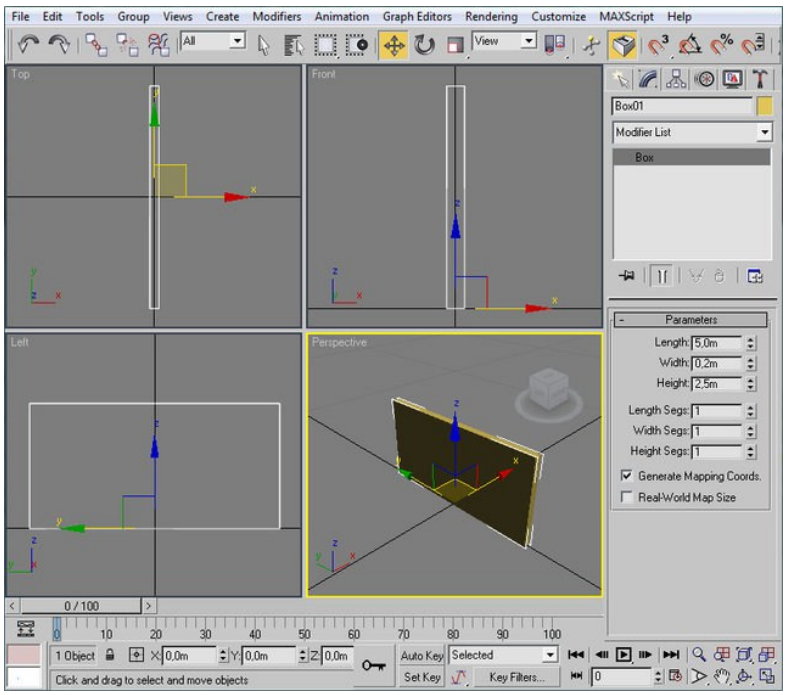
Ya hemos comentado que un procedural, técnicamente, es un perfil. Este perfil, en el simulador, determinará una forma 3D porque el motor gráfico del juego lo extrusionará en base a unas texturas que hemos informado:



En esta carretera que muestro el componente procedural es el spline en rojo. Pero para informar de cómo se deberá texturar la forma que se genere deberemos proceder también nosotros a dar esa tercera dimensión y texturarla.

Empecemos con nuestro muro.

En el viewport **Top** crearemos un cubo de las dimensiones siguientes:



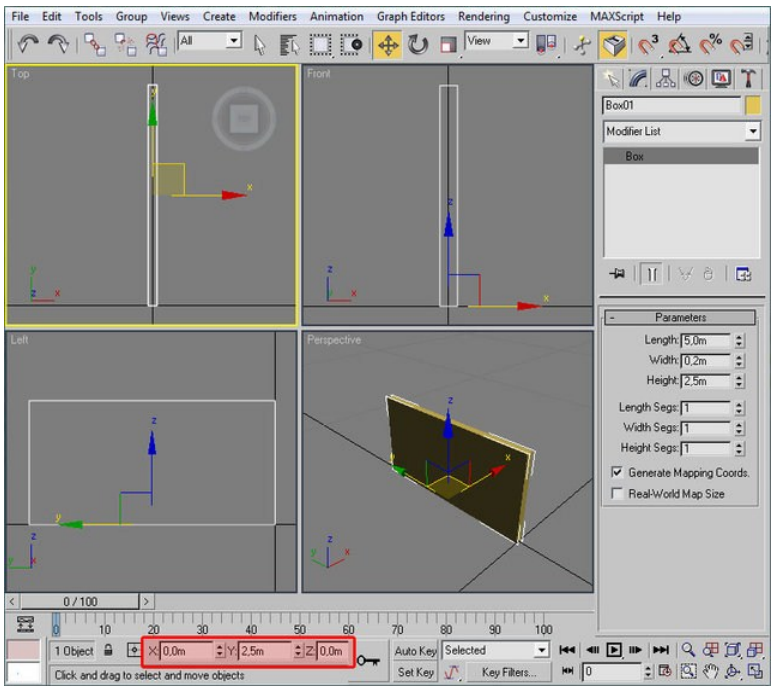
Length = 5m - Esta es la longitud que tomaremos como referencia para el texturado, es decir, la textura se repetirá cada 5 metros de longitud "real" del objeto procedural. Si quisiéramos que la textura se repitiera cada 12 metros, por ejemplo, esta sería la dimensión a dar en longitud.

Recordemos que la longitud del objeto en el simulador será la que desee el creador de rutas que use nuestro objeto, no esta que indicamos.

Width = 0,2m - Este es el ancho que daremos al muro, 20 centímetros. Podría ser otro, claro.

Height = 2,5m - Esta es la altura del muro. Un muro que rodeará instalaciones protegiendo su acceso y su contenido a las miradas de curiosos.

El cubo lo desplazaremos a:



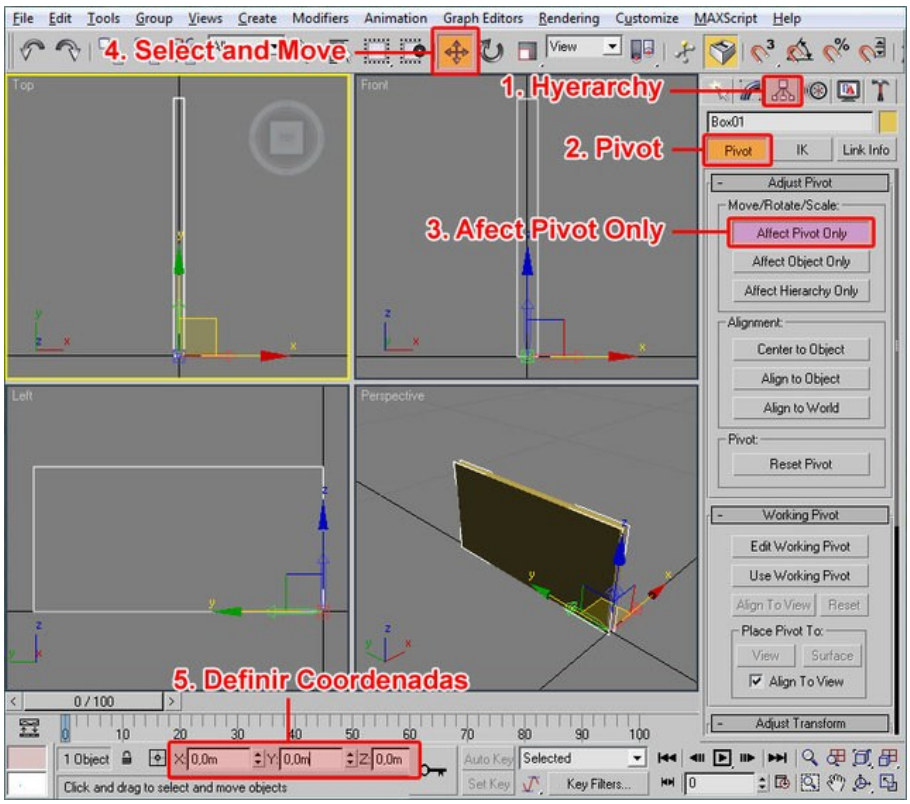
X = 0,0m - Para centrarlo respecto a su eje (salvo que quisiéramos crear un procedural descentrado, aunque no veo en este momento ningún caso que lo requiera).

Y = 2,5m - En este eje debemos desplazarlo la mitad de su longitud, para conseguir de esta forma que el extremo posterior del objeto se sitúe exactamente en el plano definido por la coordenada Y=0. Las aristas de nuestro objeto que coincidan con este plano serán las que definan el elemento procedural (las que os mostré en rojo en el ejemplo de la carretera).

Z = 0,0m - Para enrasarlo respecto al suelo. En realidad quien debe estar enrasado con el suelo es el pivote del objeto, pudiendo estar el objeto por encima (una catenaria por ejemplo) o por debajo (en el caso que quisiéramos alargar nuestro muro bajo tierra y así, en terrenos irregulares, evitar que el muro "levite" en algún tramo).

Le daremos nombre estándar de RW, por ejemplo: "1_1250_Muro". El nombre seguirá las convenciones generales de nombres para RW.

Por último desplazaremos el pivote al origen de la escena:

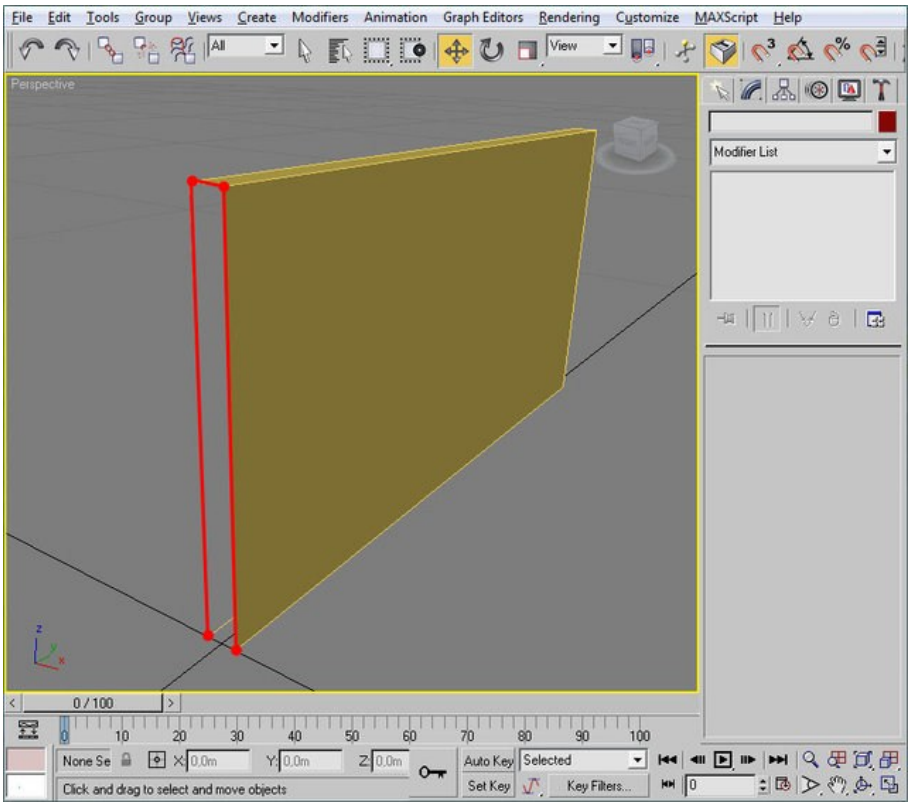


1. Vamos a la solapa Hierarchy.
2. Comprobamos que estamos en modo Pivot.
3. Pulsamos el botón Affect Pivot Only para asegurarnos que desplazaremos únicamente el pivote del objeto, dejando éste tal como está.
4. Seleccionamos la herramienta de mover.
5. Introducimos las coordenadas 0, 0 y 0 para situar el pivote en el punto de origen de la escena.

Ahora optimizaremos el elemento eliminando aquellos polígonos del cubo que no nos interesan:

- Siempre, las caras anterior y posterior, porque no participan del elemento procedural (la famosa línea roja).
- Y en este caso, la cara inferior, porque al descansar el muro sobre el terreno no será visible.

Este es el aspecto final que esperamos:

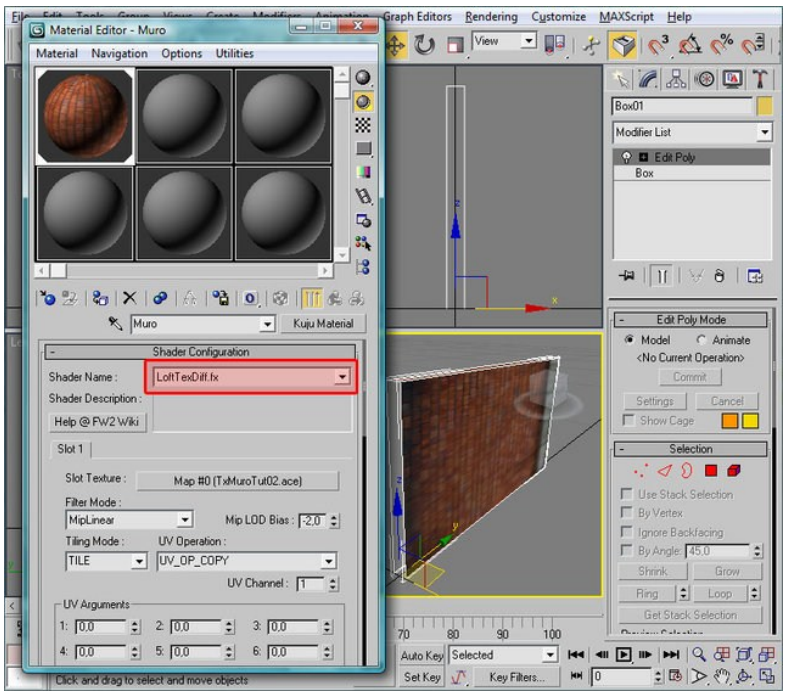


Donde he señalado en rojo el procedural, el cual recuerdo que debe tener todos sus puntos con la coordenada Y=0 (coordenada en 3ds), el pivote centrado en la escena y la extrusión de dicho procedural orientada en la dirección Y del pivote del objeto (con la longitud que deseemos para las texturas).

Si hemos creado el cubo en el Viewport Top, tal como se ha comentado, y se han realizado estas sencillas operaciones nuestro cubo tiene todos los requisitos para que se renderice en el simulador como un objeto procedural sin problemas.

Tan sólo resta mapear aquellas texturas que deseemos para nuestro objeto. En el presente caso he creado la hoja de texturas TxMuroTut02, que antes he mostrado.

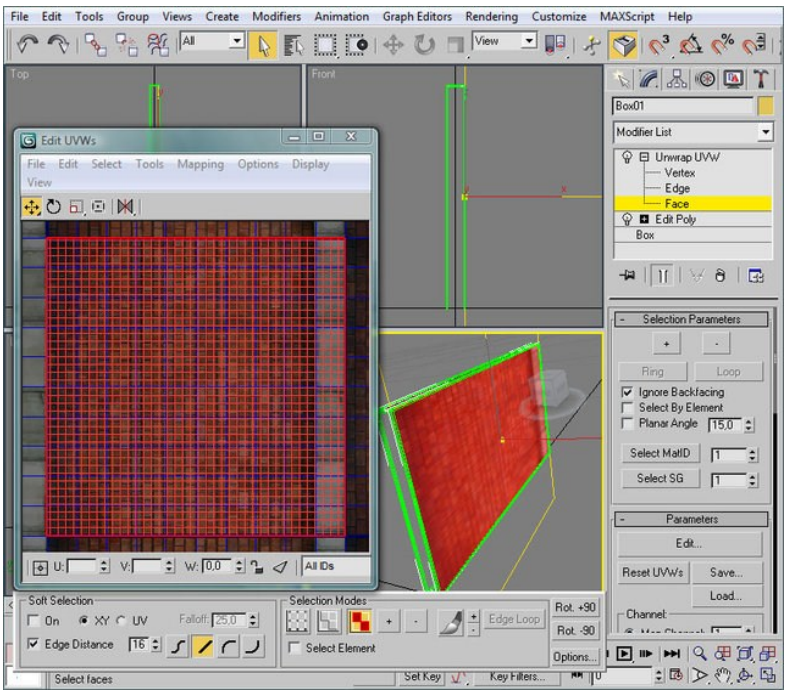
Crearemos un material al cual hay que asignar uno de los shader que empiezan por "Loft...". En particular "LoftTexDiff.fx" actúa como el shader TexDiff, pero en procedurales.



Observemos que todos los materiales "Lofts" sólo pueden realizar el efecto de mosaico de las texturas en la dirección "V", es decir, cuando se mira la textura en Photoshop el mosaico se lleva a cabo únicamente en la dirección vertical, razón por la cual la hemos girado 90º.

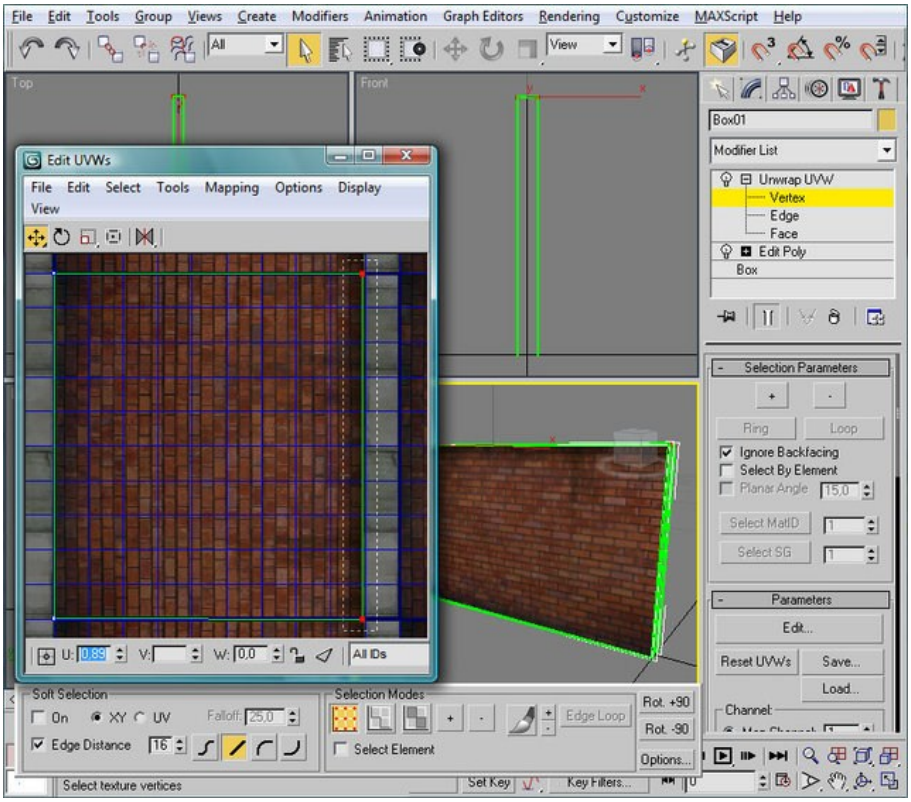
Por tanto, ahora editaremos el mapeo del muro:

- Con el muro seleccionado, añadiremos un modificador "Unwrap UVW".
- Seleccionaremos "Faces".
- Pulsaremos el botón "Edit".
- Una vez abierto el editor UVW pincharemos en un viewport (el de perspectiva será ideal) sobre un lateral del muro.
- Al seleccionarse el polígono en el editor pulsaremos el botón "Rot.+90" o "Rot. -90", según el caso, para enderezar el mapeo de la textura.
- Cerraremos el editor para ver el efecto del giro.[/list]



Repetiremos esta acción para el otro lado del muro. Una vez ambos lados tengan el mapeo correcto, nuevamente entraremos en el editor UVW para ajustar el mapeo a la zona de ladrillos exclusivamente:

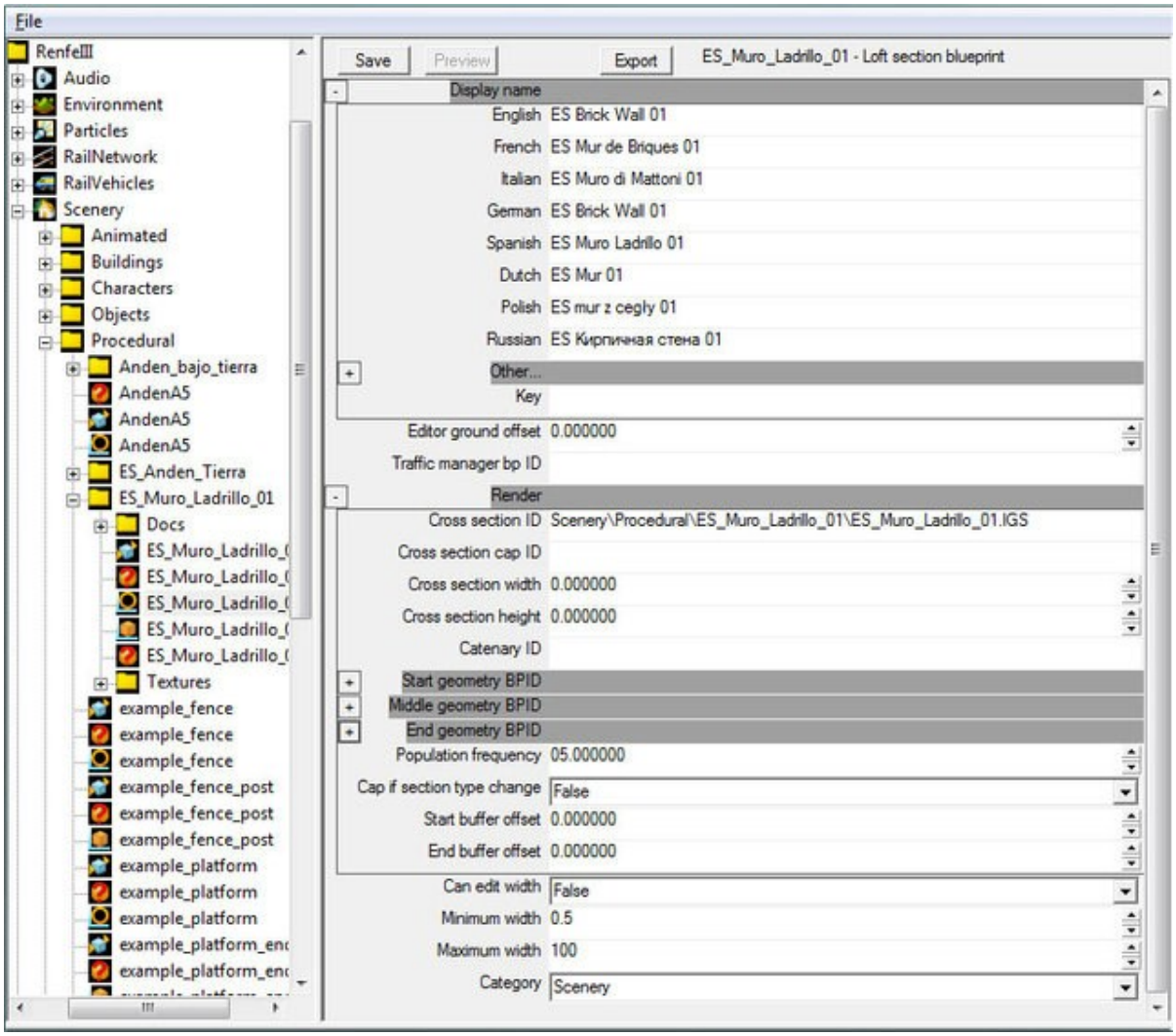
- Seleccionamos "Vertex".
- Creamos con el ratón una selección que incluya los vértices superiores del muro (son los que quedan a la derecha de los mapeos una vez girados).
- En el campo U damos el valor 0,89 y pulsamos "Intro", puesto que la textura de ladrillos la creamos en el 90% (0,9) de la zona izquierda de la textura.
- Ya podemos cerrar el editor UVW.



Con todo esto no hemos terminado el muro, pero vamos a exportar lo realizado para comprobar que de momento todo funciona y poder dar por bueno el trabajo realizado hasta ahora. Rápidamente haremos:

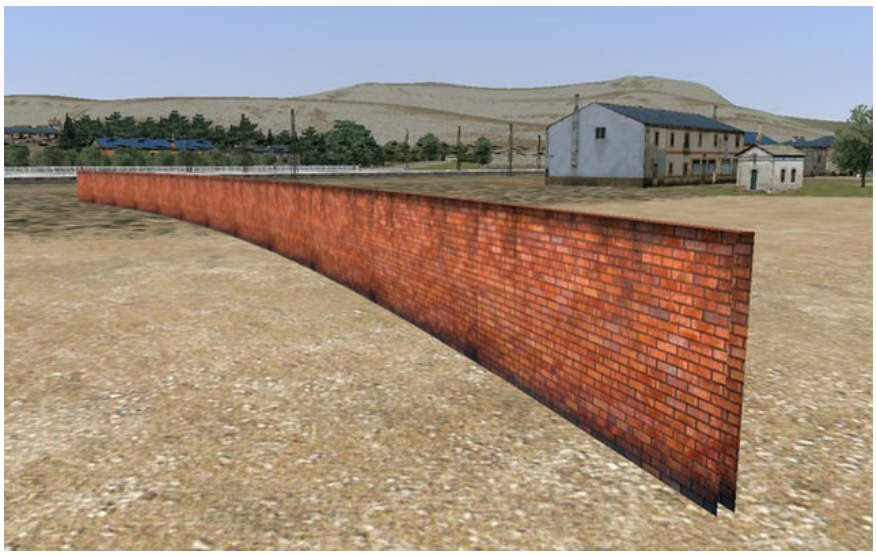
- "Collapse All" al stack del muro, así nos quedará una "Editable Poly" limpia.
- "Export" con el formato IGS, como si de cualquier otra pieza se tratase.

En el Blueprint Editor crearemos un nuevo "blueprint" de tipo "Loft section blueprint", donde básicamente cumplimentaremos:



- El nombre que daremos al muro en los diferentes idiomas. El Traductor de Google (por ejemplo) nos ha quitado la excusa de no saber idiomas.
- En "Cross section ID" informaremos del nombre del archivo IGS resultado de la exportación, junto con su ruta.
- En "Category" seleccionaremos "Scenery".

Con estas pocas acciones podemos exportar e ir al simulador a ver como luce nuestro muro:

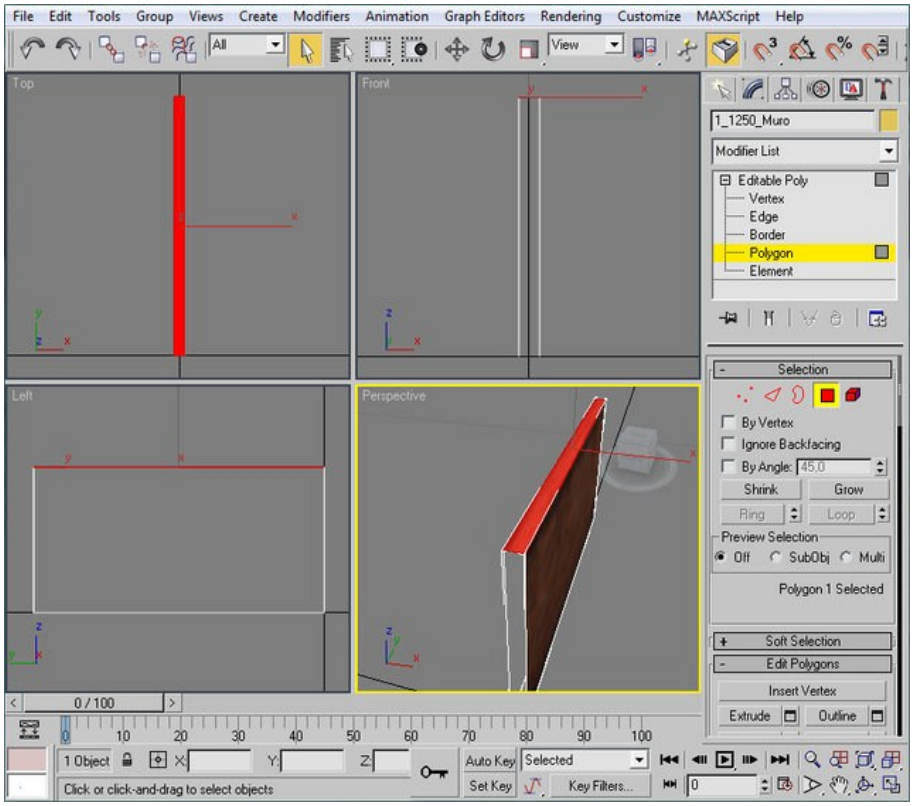


Observaremos que no es todavía lo que queremos, pero que el simulador lo entiende y genera (yo le he dado una ligera curva) en la longitud solicitada. La textura se repite cada 5 metros, como queríamos, con independencia de la longitud generada. En el extremo se aprecia perfectamente la forma del procedural definido. Y por último, cabe observar que los shaders "Loft..." todos ellos generan el mapeo por ambos lados de los polígonos, por lo que en procedurales verticales no es necesario (según el caso) más de un polígono.

Esta es la base para cualquier elemento procedural, de momento descansaremos. La explicación ha sido detallada quizás en exceso, pero así no me descuido nada ;) En un nuevo capítulo le añadiremos el remate superior y las columnas de soporte para dejar el muro tal como hemos deseado al principio.

Vamos a continuar con el trabajo del componente procedural añadiendo el remate superior de piedra.

En primer lugar seleccionaremos el polígono superior del muro que tenemos para eliminarlo, dado que al cubrirlo con la piedra no será visible.



Una pulsación en la tecla "Supr" y listos.

Ahora crearemos otro cubo en la vista Top, y le daremos las dimensiones:

Length = 5m - Esta es la longitud que tomaremos como referencia para el texturado, y será la misma que para el muro.

Width = 0,35m - Le daremos unos 15 centímetros más que al muro.

Height = 0,1m - Con esta altura (grosor) será suficiente para el caso que nos centra.

Y el cubo lo desplazaremos a:

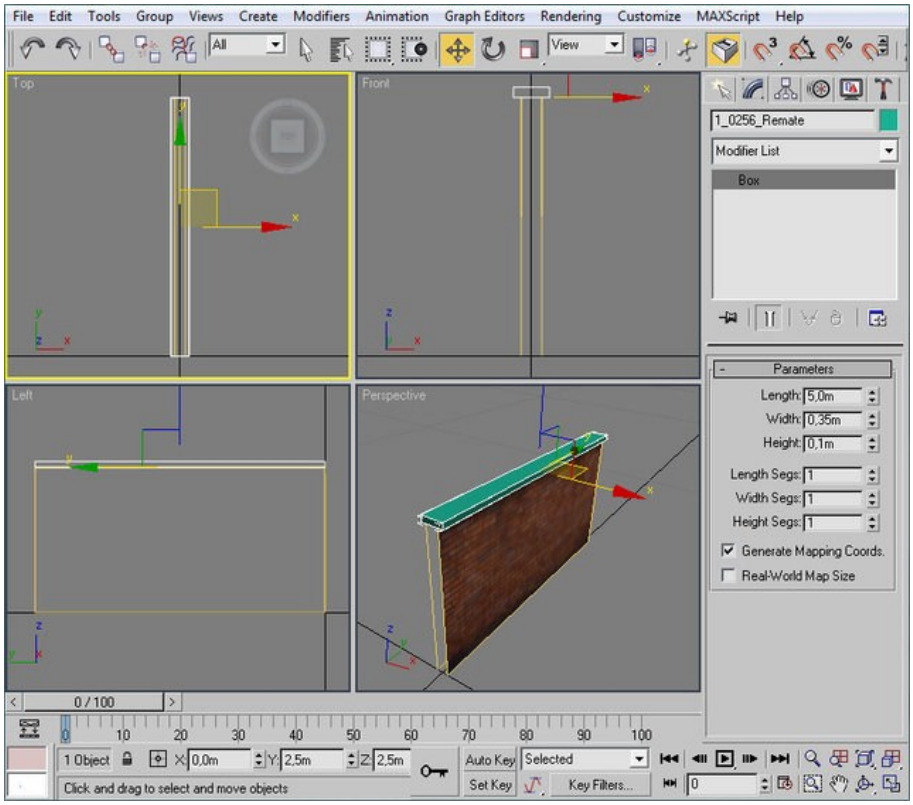
X = 0,0m - Para centrarlo respecto a su eje.

Y = 2,5m - La mitad de su longitud, para conseguir que nuevamente el extremo posterior del remate se sitúe exactamente en el plano definido por la coordenada Y=0.

Z = 2,5m - Para enrasarlo sobre el muro.

Le llamaremos 1_0256_Remate, por ejemplo.

Como de costumbre le aplicaremos un modificador Edit Poly y suprimiremos los polígonos anterior y posterior, que ya dijimos que no participarán del procedural.



Por último desplazaremos el pivote al origen de la escena, pues también este elemento del objeto debe estar en el plano Y=0:

- Vamos a la solapa Hierarchy.
- Comprobamos que estamos en modo Pivot.
- Pulsamos el botón Affect Pivot Only para asegurarnos que desplazaremos únicamente el pivote del objeto, dejando éste tal como está.
- Seleccionamos la herramienta de mover.
- Introducimos las coordenadas 0, 0 y 0 para situar el pivote en el punto de origen de la escena.

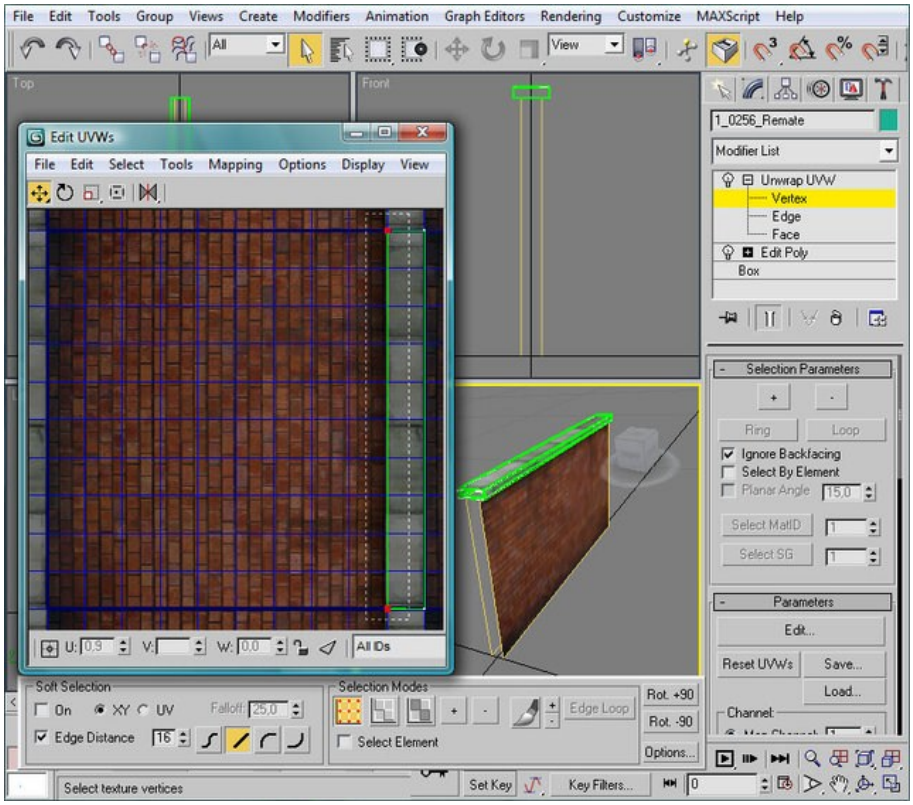
Abriremos el editor de materiales y le asignaremos el mismo material que al muro, dado que en esta textura también tenemos las losas de piedra y que el shader debe ser también uno del tipo "Loft...".

Al igual que en el caso del muro, el mapeo de los dos polígonos laterales lo debemos girar 90º. En el caso de las caras superior e inferior esto no es necesario dado que 3ds, por defecto, para estas orientaciones produce mapeos también verticales. Por tanto, ahora editaremos el mapeo del remate:

- Con el remate seleccionado, añadiremos un modificador "Unwrap UVW".
- Seleccionaremos "Faces".
- Pulsaremos el botón "Edit".
- Una vez abierto el editor UVW pincharemos en un viewport (el de perspectiva será ideal) sobre un lateral del remate.
- Al seleccionarse el polígono en el editor pulsaremos el botón "Rot.+90" o "Rot. -90", según el caso, para enderezar el mapeo de la textura.
- Cerraremos el editor para ver el efecto del giro.

Repetiremos esta acción para el otro lado del remate. Una vez ambos lados tengan el mapeo correcto, nuevamente entraremos en el editor UVW para ajustar el mapeo a la zona de las losas de piedra exclusivamente:

- Seleccionamos "Vertex".
- Creamos con el ratón una selección que incluya los vértices inferiores del remate (son los que quedan a la izquierda de los mapeos una vez girados).
- En el campo U damos el valor 0,9 y pulsamos "Intro", puesto que la textura de piedras la creamos en el 10% (0,1 de anchura entre U=0,9 y U=1,0) de la zona derecha de la textura.
- Ya podemos cerrar el editor UVW.

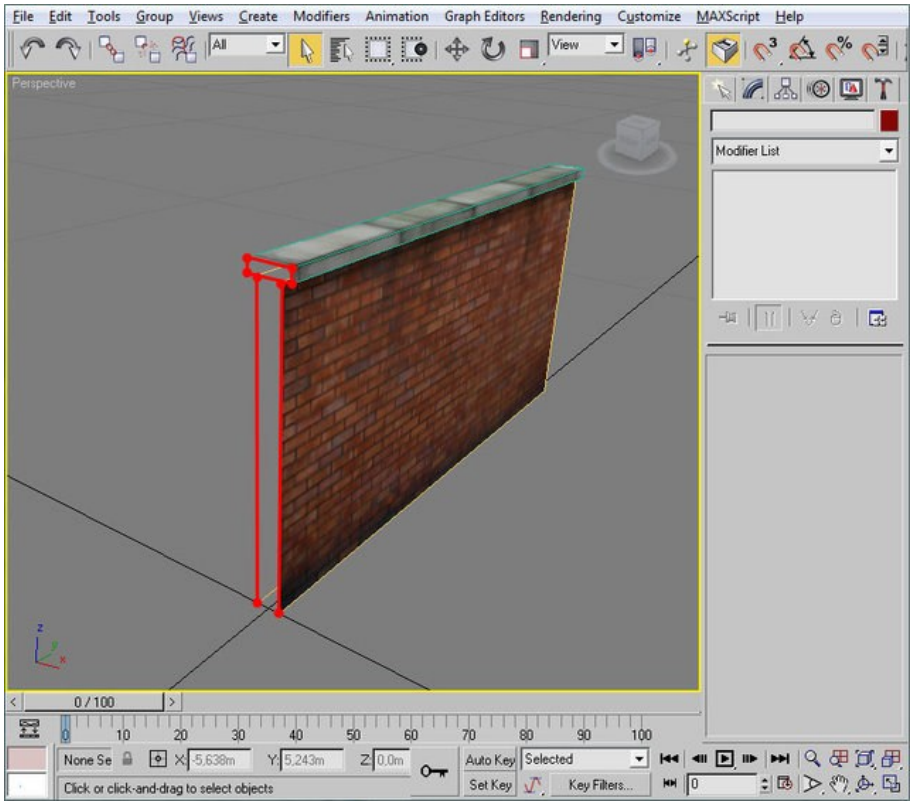


Hemos terminado el remate, y con él, también el elemento procedural. Para poder exportarlo haremos, como de costumbre:

"Collapse All" al stack del remate, así nos quedará una "Editable Poly" limpia.

"Export" con el formato IGS.

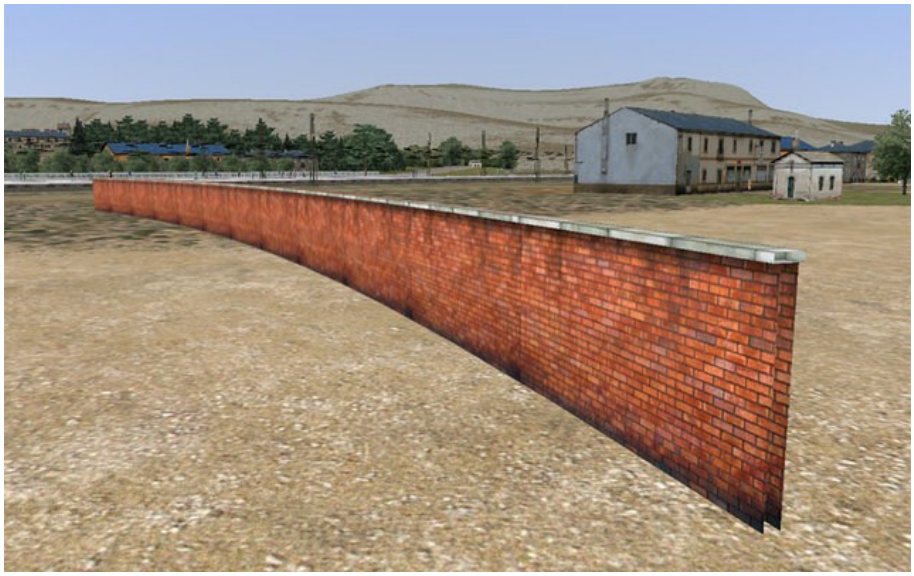
El elemento procedural nos ha quedado como muestro:



Si os fijáis en los trazos rojos (las aristas que están en la coordenada Y=0) este elemento está compuesto por dos trazos paralelos y separados, para el muro, y por una línea cerrada cuadrada, para el remate. Es decir, el elemento procedural no es necesario que esté formado por un trazo continuo ni único para poder ser representado, incluso puede contener LODs de cualquiera de los tipos que vimos para la casilla, con sus respectivas jerarquías. Lo único que no podemos incluir es un elemento sombra, dado que todos sus componentes "deben" tener asignados materiales con shaders de tipo Loft.

Lo que sí podemos incluir sin ningún problema son texturas estacionales para el procedural

Como ya hemos generado una exportación del muro con su remate, podemos ir al blueprint editor para exportar a su vez el objeto al simulador y ver cómo queda en su estado actual:

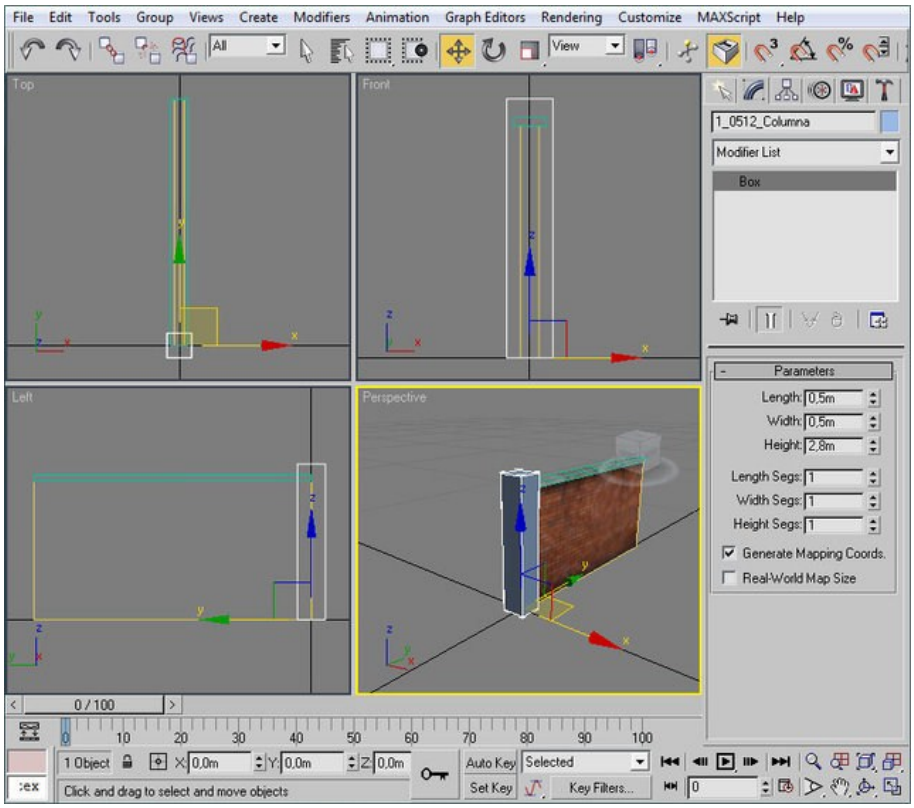


1.3. El componente fijo

Nos resta tan sólo realizar el elemento fijo para el procedural, en este caso: la columna de sustentación. Crearemos un cubo de las dimensiones:

- Length = 0,5m
- Width = 0,5m
- Height = 2,8m

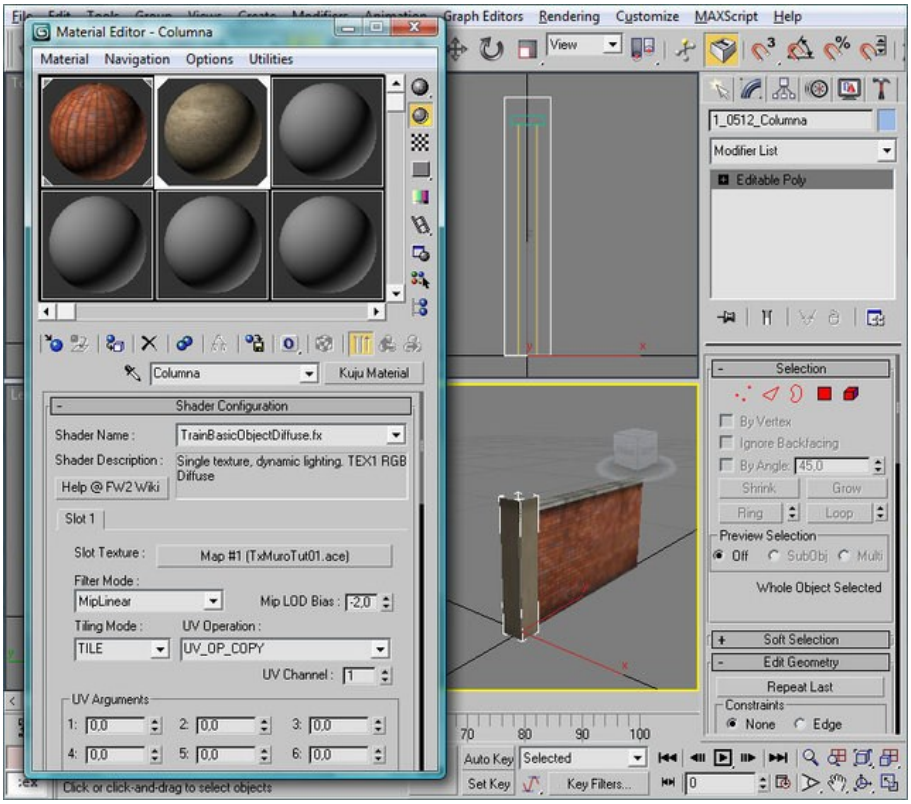
Le denominaremos 1_0512_Columna, y lo centraremos en el origen del muro, moviéndolo a la posición X=0, Y=0 y Z=0.



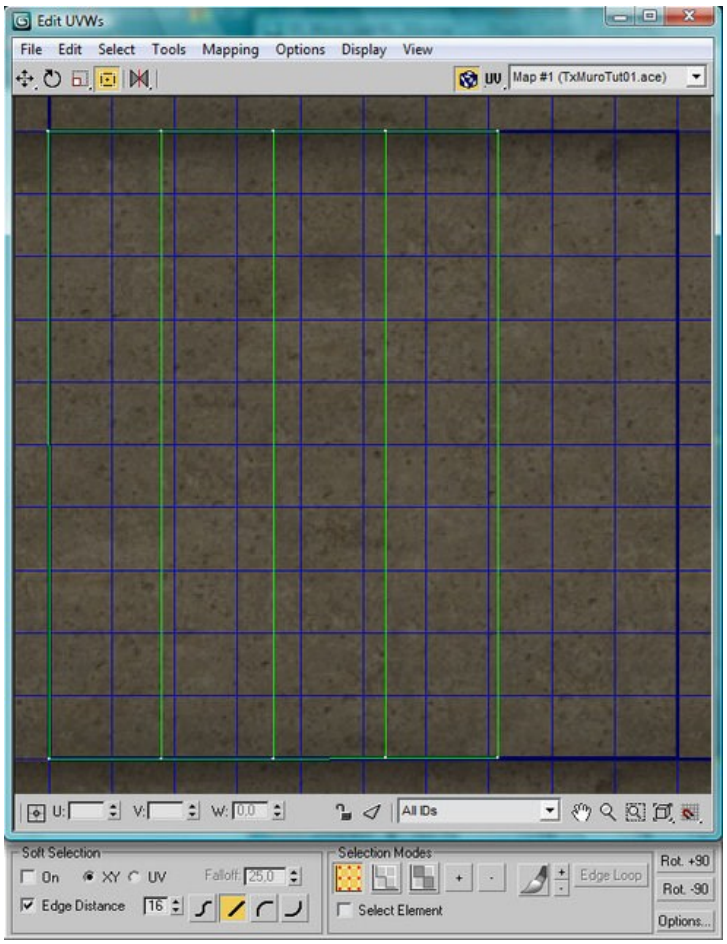
Lo convertiremos en Editable Poly y le suprimiremos las caras inferior y superior, dado que no se verán.

En el editor de materiales crearemos un nuevo material al que asignaremos un shader "normal" TrainObjectTexDiffuse.fx, es decir no precisa ser un shader "Loft" [1] puesto que la columna será un elemento escénico normal y corriente (aunque participe en la generación del muro procedural). A este material le asignaremos la textura de hormigón que preparamos en su momento "TxMuroTut01.ace":

[1] Como "curiosidad" he de comentar que he probado a asignar un shader "Loft..." a un objeto escénico (no a un procedural) y éste se renderiza perfectamente con este shader también. O sea, mientras que los shaders "Loft" son obligatorios para los elementos procedurales, en el caso de su uso en objetos NO procedurales no están en absoluto prohibidos .



Asignaremos el material a la columna y procederemos a añadirle un modificador Unwrap UVW para ajustar el mapeo del hormigón, y que éste sea proporcional al elemento.



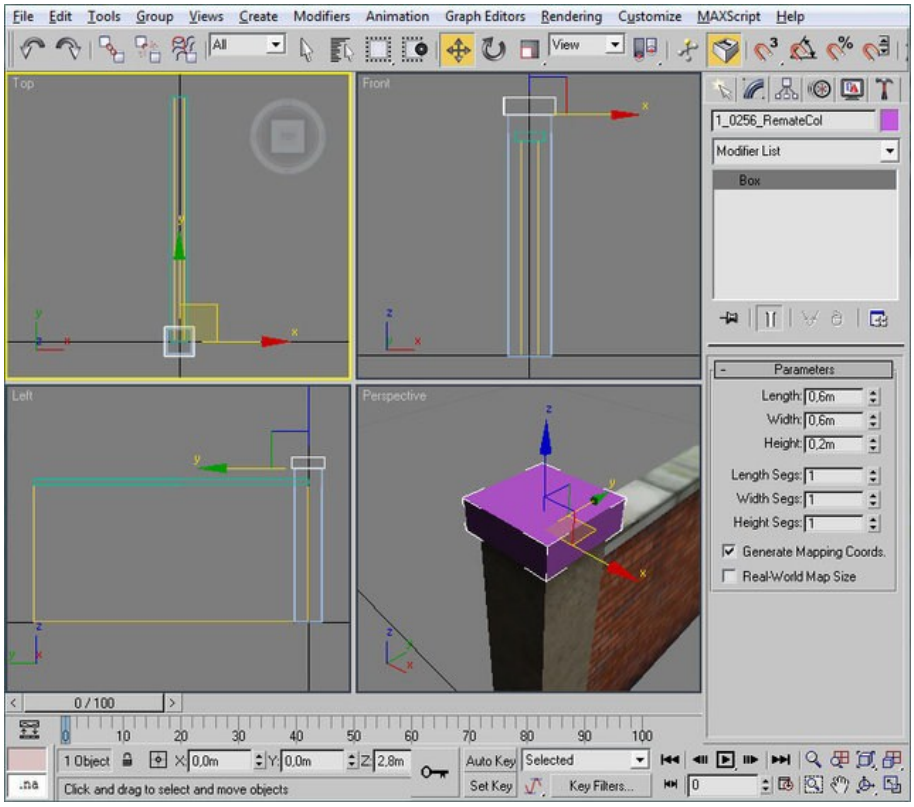
- Con la columna seleccionada, añadiremos un modificador "Unwrap UVW".
- Seleccionaremos "Faces".
- Pulsaremos el botón "Edit".
- Una vez abierto el editor UVW seleccionaremos todas las caras de la columna con "Control + A".
- Seleccionaremos Mapping -> Flatten Mapping..., reduciremos Spacing a 0 y aceptaremos. Las caras de la columna han tomado la proporción adecuada en alto y ancho, y están todas ajustadas una al lado de la otra. Observemos que las columnas han ocupado algo más del 70% de la textura en anchura, dejando libre el 25% derecho.
- Cerraremos el editor UVW.

Podemos colapsar el stack de la columna para aceptar todas estas modificaciones.

Ahora crearemos el remate de la columna mediante un nuevo cubo con las dimensiones:

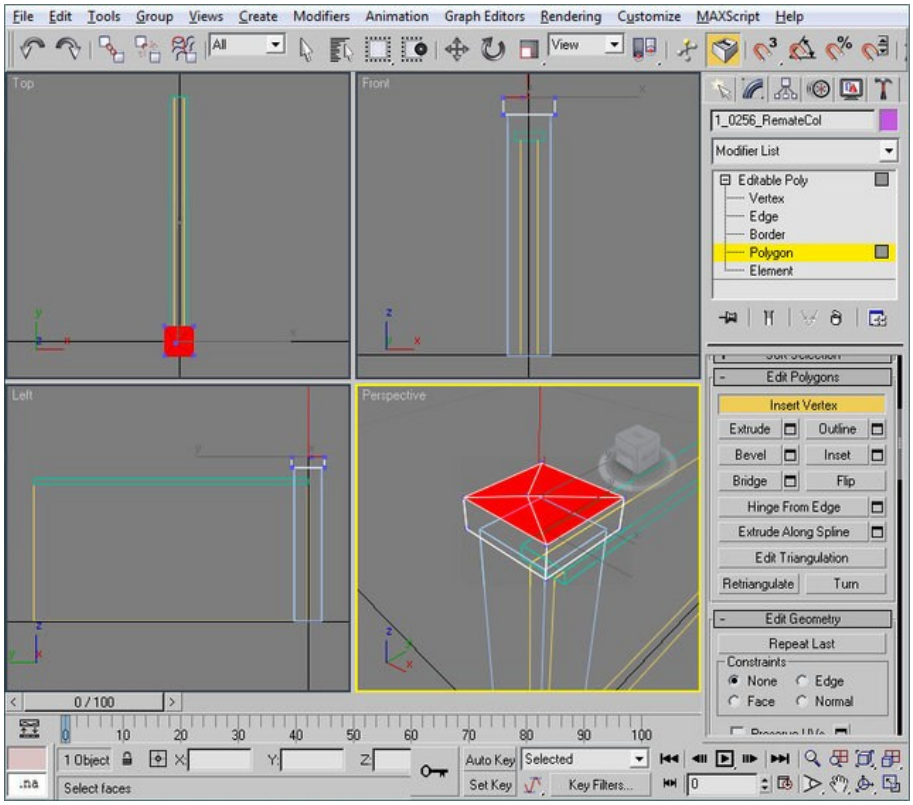
- Length = 0,6m
- Width = 0,6m
- Height = 0,2m

Le denominaremos 1_0256_RemateCol, y lo centraremos sobre la columna, moviéndolo a la posición X=0, Y=0 y Z=2,8m.

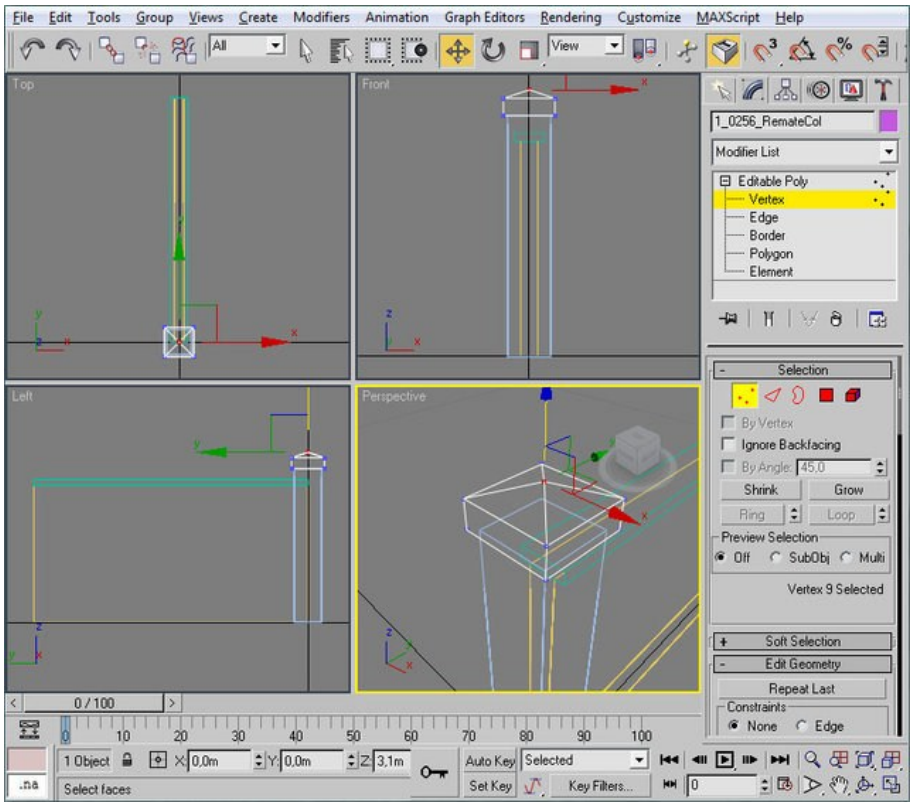


Lo convertiremos en Editable Poly y pasaremos a dar a la cara superior la forma de pirámide:

- Pasaremos a modo "Polygon".
- Seleccionaremos la cara superior del remate de la columna.
- En el grupo "Edit Polygons" pulsaremos el botón "Insert Vertex".
- Con el ratón pincharemos sobre el polígono seleccionado para crear un vértice nuevo. No es necesario que nos esforcemos en que nos quede centrado dentro del polígono, porque no lo conseguiremos. Si os fijáis yo le he dado una situación francamente descentrada para verle bien.



- Pasaremos a modo "Vertex".
- El vértice que acabamos de crear estará seleccionado (y si no lo estuviera lo seleccionamos). Pulsamos "Select and Move".
- Moveremos el vértice a las coordenadas X=0, Y=0 (para centrarlo en el polígono) y Z=3,1 (elevándolo 10 centímetros sobre su posición anterior).

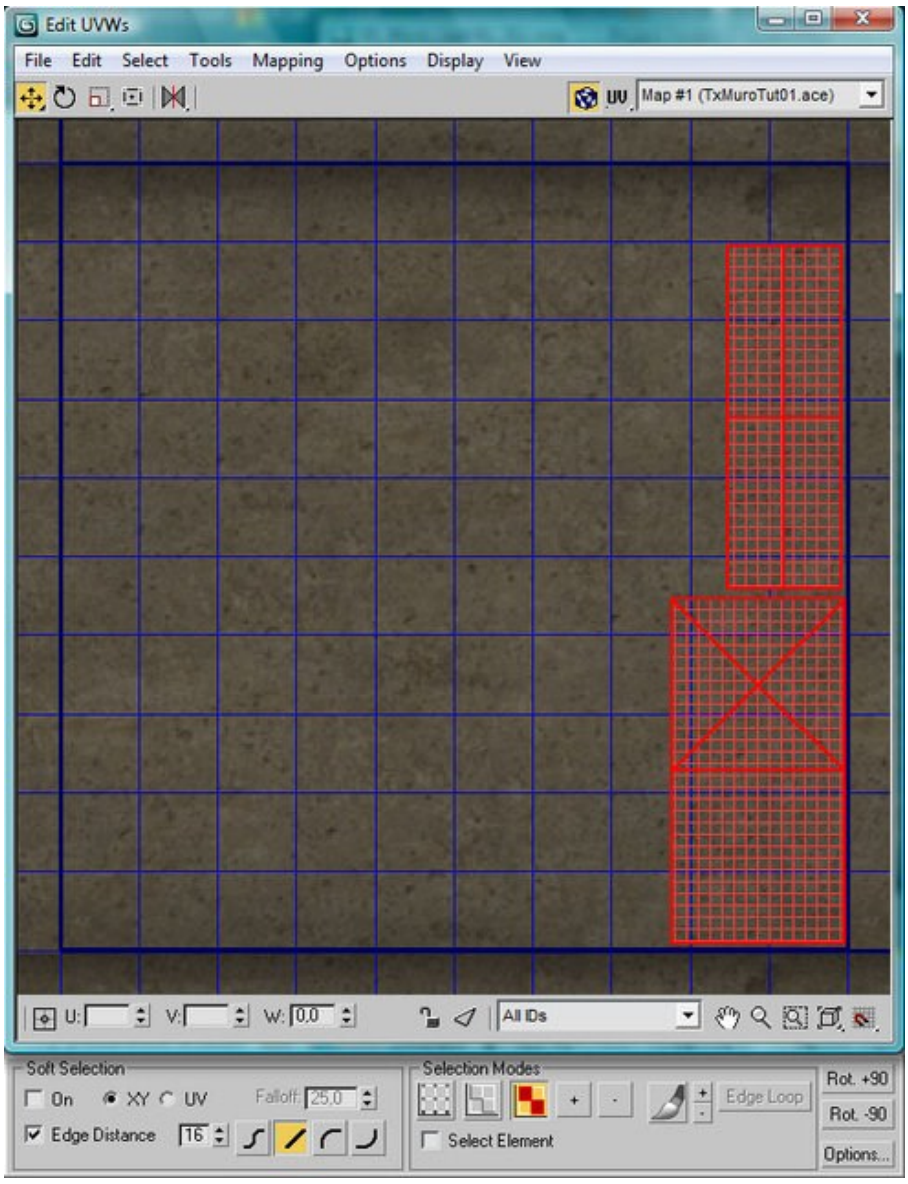


Con esta operación le hemos dado al remate la forma piramidal deseada.

En modo "Polygon" deberemos seleccionar todas las caras del remate (Control + A) y en la sección "Smoothing groups" ejecutar un "Clear All" para evitar que se suavicen las aristas superiores del remate.

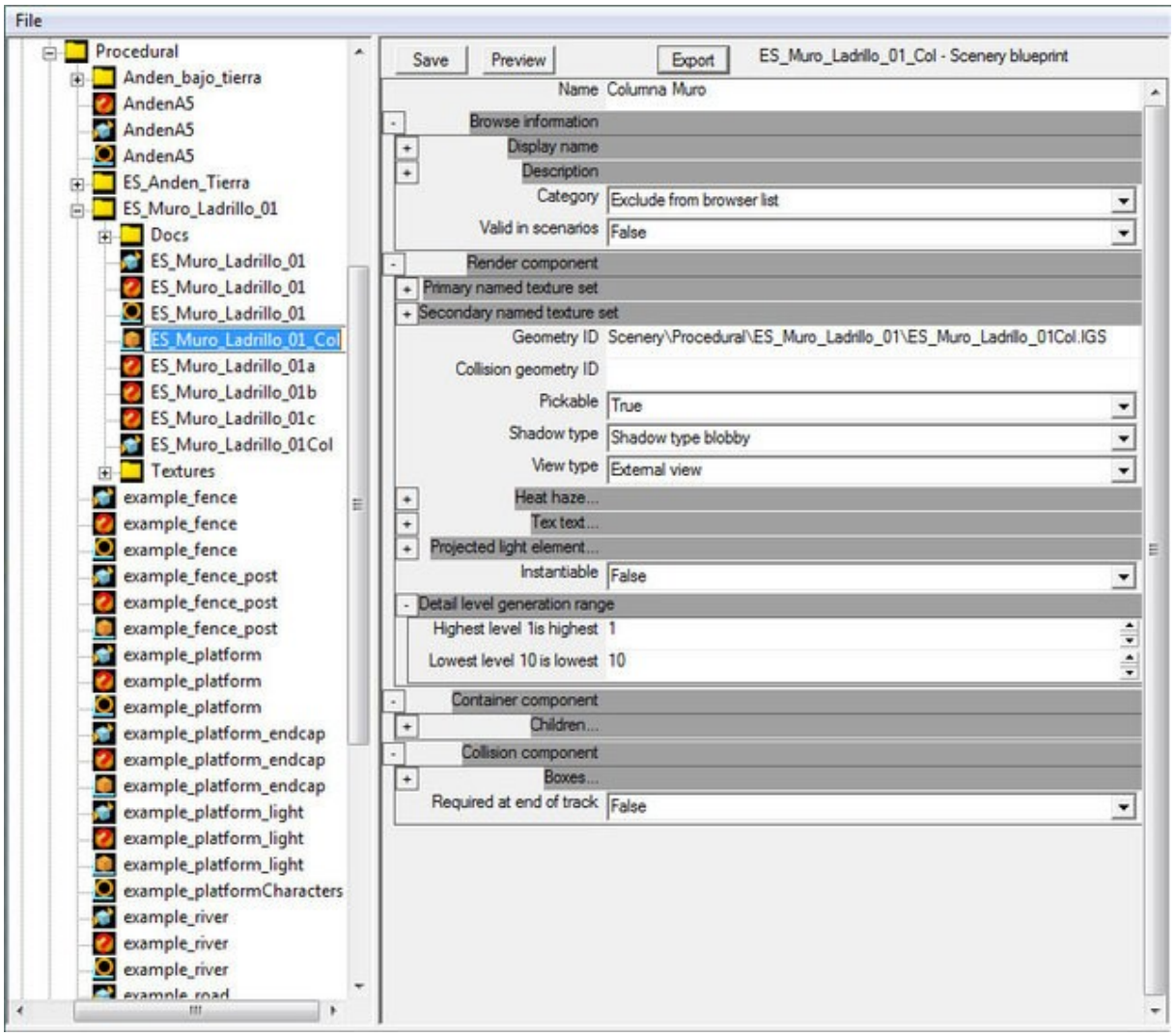
Asignaremos al remate el material de hormigón que hemos creado anteriormente para la columna y procederemos a añadirle un modificador Unwrap UVW para ajustar el mapeo de este elemento.

- Con el remate de la columna seleccionado, añadiremos un modificador "Unwrap UVW".
- Seleccionaremos "Faces".
- Pulsaremos el botón "Edit".
- Una vez abierto el editor UVW seleccionaremos todas las caras del remate con "Control + A".
- Seleccionaremos Mapping -> Flatten Mapping..., reduciremos Spacing a 0 y aceptaremos. Las caras se han distribuido de forma uniforme, no obstante las desplazaremos y escalaremos para que ocupen el 25% del margen derecho de la textura.
- Cerraremos el editor UVW.



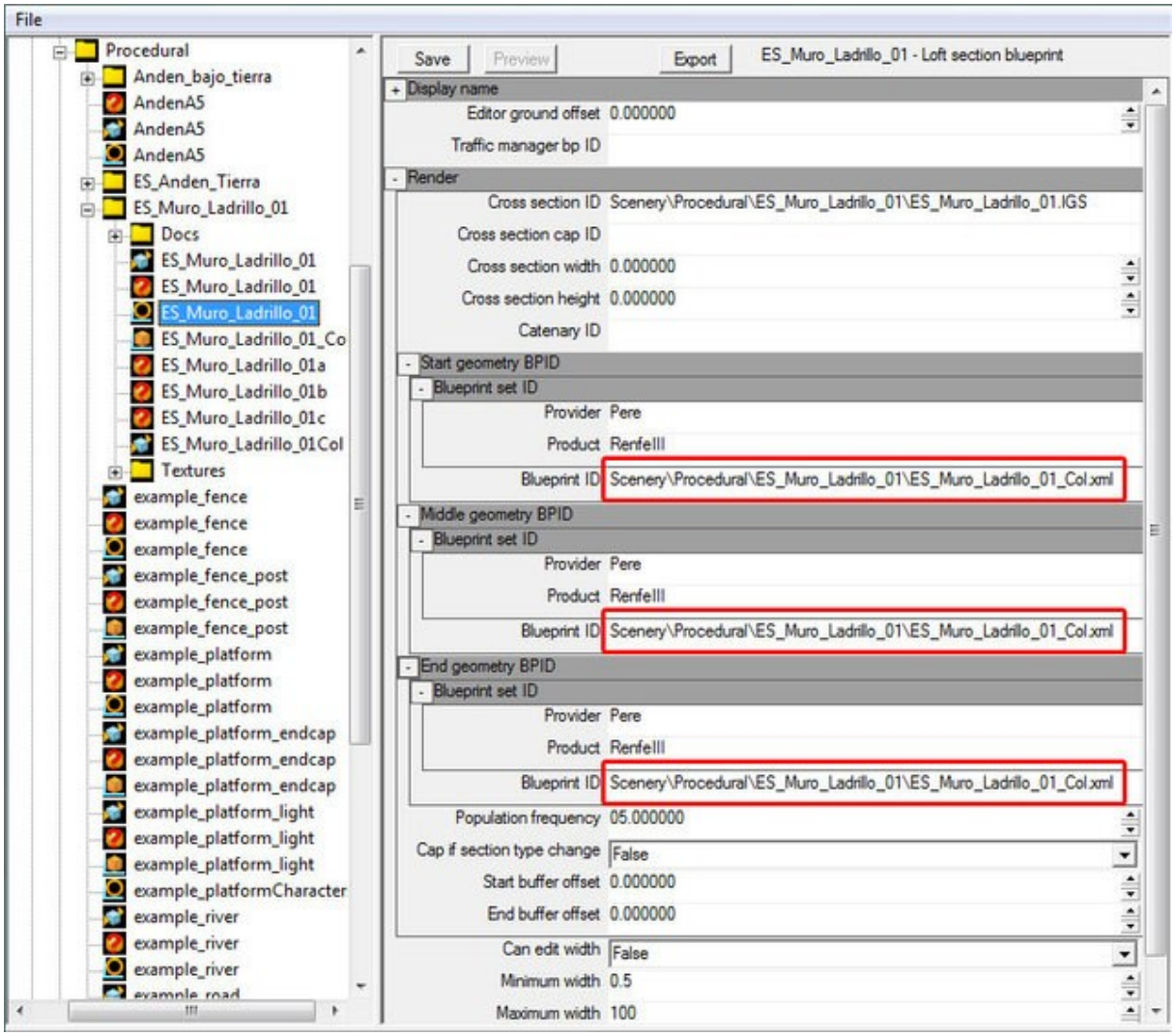
Podemos colapsar el stack del remate de la columna para aceptar todas estas modificaciones y proceder a exportar la columna como "ES_Muro_Ladrillo_01Col.IGS", para lo cual debemos de tener presente que estamos en una escena que incluye también el muro procedural y su remate, por lo que esconderemos (Hide) dichos elementos procedurales y así al exportar el elemento fijo no nos molestarán (los podríamos haber borrado, pero de esta forma los mantenemos todos en una única escena más manejable).

A la columna del muro deberemos crearle un blueprint del tipo "Scenery blueprint". Efectivamente, este elemento es un objeto normal a todos los efectos como lo puede ser un edificio. El blueprint lo cumplimentaremos como de costumbre:



Y exportaremos el elemento.

Ahora estamos en disposición de incorporar la columna al muro procedural anterior, para lo cual abrimos su blueprint y modificaremos tres parámetros del mismo:



En las secciones "Start geometry PBID", "Middle geometry PBID" y "End geometry PBID" le informaremos del Provider, Product y la ruta del blueprint de la columna que acabamos de crear. Por supuesto que, como podéis observar, estos tres elementos fijos del procedural (principio, medio y final) pueden ser diferentes entre sí si es preciso, aunque en nuestro caso no ha sido necesario.

Además, en "Population frecuency" informaremos de la distancia, en metros, a la cual el elemento "Middle geometry" se repetirá a lo largo del objeto procedural, en nuestro caso serán 5 metros.

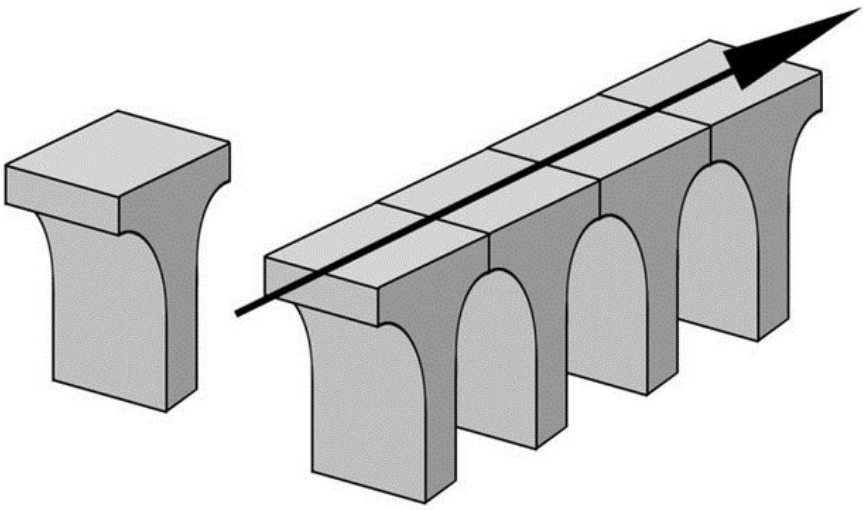
Ya podemos exportar el blueprint del muro procedural y observar el resultado en el simulador:

El elemento fijo, la columna, podría incluir una sombra, pero en este caso no sería recomendable puesto que el muro no proyectará sombra, y quedaría muy raro que las columnas del mismo sí lo hicieran. En otras situaciones, por contra, puede ser interesante incluir sombra en el elemento fijo de un procedural: en el caso de un andén que incluya farolas cada equis metros es hasta deseable que estas proyecten sombra sobre el andén u otros objetos en la cercanía.

Y este ha sido todo el misterio de un objeto procedural. Un perfil Loft a extrursionar y unos elementos escénicos que se pueden colocar al principio, en medio y al final, simple ¿no? 

Muchos elementos están construidos de esta forma. Carreteras y calles, por ejemplo, ¿y qué podríamos poner de elemento fijo que se repite a lo largo de ellos? ¿Farolas quizás?

Un simple "truco" de procedurales: el perfil lo reducimos al mínimo y además le hacemos una textura con transparencias, totalmente transparente  , de esta forma obtenemos un elemento procedural invisible. Ahora, si para el elemento central repetitivo modelamos un par de semiarcadas de puente, correctamente ajustadas en distancia, obtendremos un viaducto modular de longitud y curvaturas variables listo para adaptar a nuestros trazados:



Si hemos modelado un chalé de una urbanización podemos, además, por este mismo método crear un procedural que genere una hilera de chalés que podremos situar en alineaciones rectas o curvas, para llenar rápidamente una zona Simple ¿no?

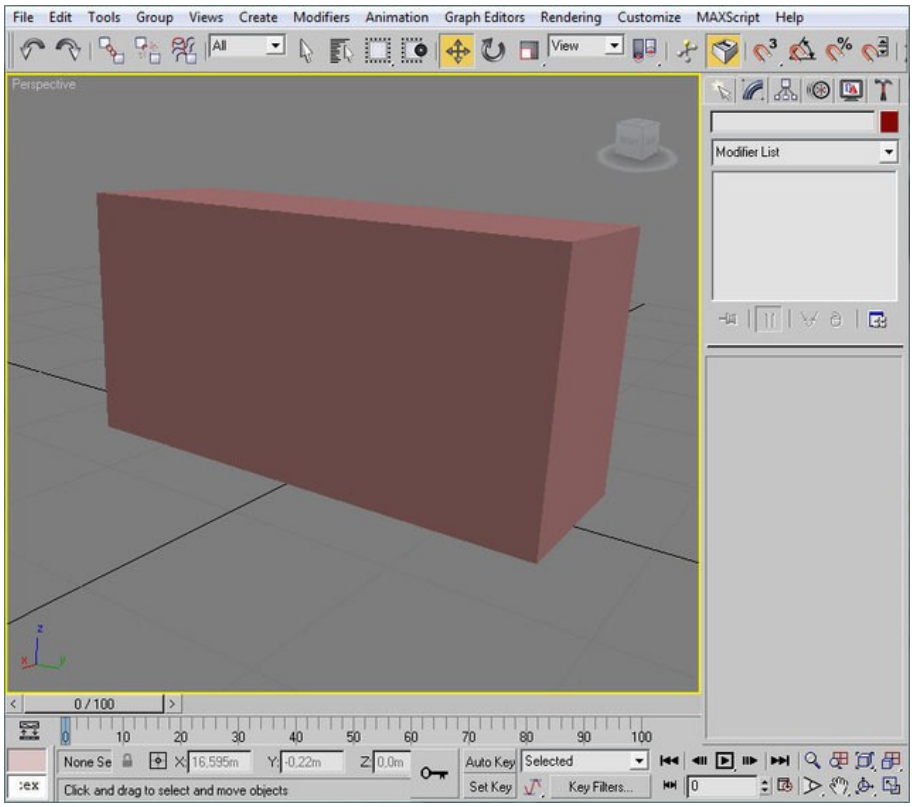
Por último, algunos de los elementos básicos del simulador son procedurales, como por ejemplo las vías. El perfil a extrusionar es el balasto y los carriles, los elementos fijos al inicio y al final de la vía son las toperas, y el elemento que se repite regularmente a lo largo del procedural pueden ser las traviesas y sus elementos. Simple ¿no?

2. Dando relieve a las superficies.

Nuestros modelos están compuestos por superficies planas. Es el caso del muro que hemos realizado.

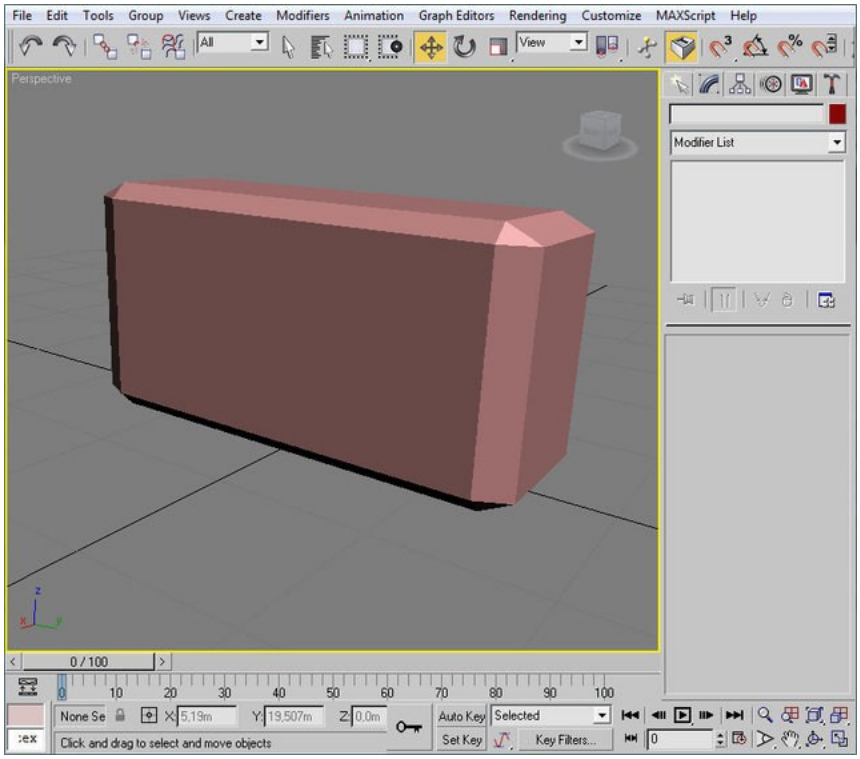
Podemos disimular más o menos este hecho mediante una textura que represente mejor o peor la superficie del polígono, pero a efectos de la iluminación éste será totalmente plano y emitirá una luz constante en toda su superficie, dado que todos sus píxeles tienen la misma orientación respecto de la luz que reciben, y por extensión que emiten como reflejo de la primera. Pensemos un momento en esto último: ¿Cómo vemos cada punto de la superficie de un objeto? Por la luz que refleja ya sea en cantidad (brillo) o calidad (color y saturación).

Cojamos un simple cubo:



Al cubo en su totalidad de he dado ese color teja pálido, por tanto, cada una de las tres caras visibles por la cámara de este cubo tiene el mismo color y la misma saturación (calidad). Pero en cambio en su renderización los tonos de cada una de las tres caras son diferentes y esa diferenciación es lo que nos da la sensación de volumen y profundidad, y eso es debido a que el motor gráfico interpreta que cada cara debe reflejar una cantidad de luz (brillo) diferentes según sea la orientación del polígono respeto a la fuente de luz y la cámara.

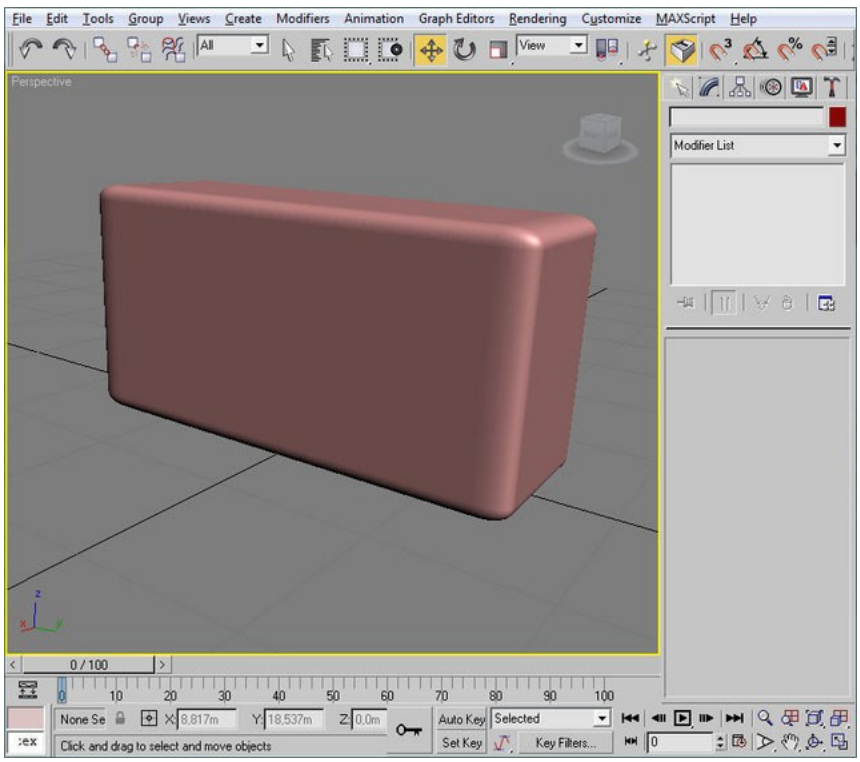
Para apreciar mejor la diferencia aplicaremos un achaflanado a todas las aristas de este ladrillo:



Las nuevas caras, con nuevas orientaciones, aun siendo del mismo color que las anteriores, presentan brillos (cantidad de luz) diferentes, más intensos en las caras orientadas hacia la luz y menos en aquellas casi perpendiculares a la fuente de luz.

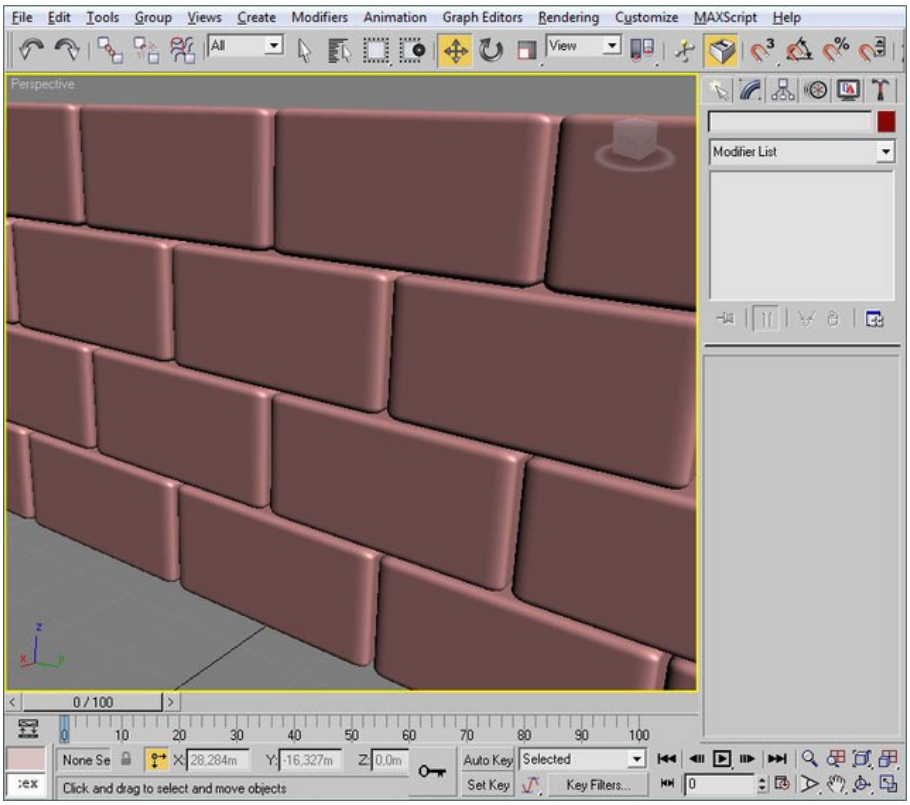
Cabe aclarar que a la dirección de un polígono, que es perpendicular a la superficie de éste, se le denomina "**normal**" del polígono, y se define por un vector en base a su magnitud en las tres coordenadas X, Y y Z (como todo en nuestro mundo 3D).

Incrementemos la cantidad de achaflanados de las aristas para producir un suavizado (redondeado) de éstas:



Se aprecian las diferentes cantidades de luz que se representan en cada arista redondeada, y esto es debido a que he subdividido dicha arista en muchos pequeños polígonos y cada uno de estos tiene una ligera variación en su orientación (en su normal), lo cual produce este efecto de volumen redondeado.

Al igual que hemos realizado este "ladrillo" podríamos modelar todos y cada uno de los ladrillos de un muro, consiguiendo una sensación de relieve del mismo muy acusada:

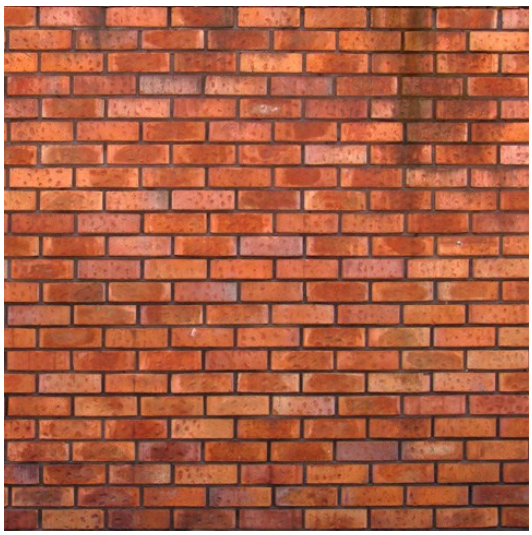


El problema con el que nos encontramos es que cada uno de estos ladrillos tiene él sólo más de 7000 polígonos debido a que he realizado un achaflanado de todas las 12 aristas del cubo subdividiéndolas en 30 segmentos cada una (por si lo queréis probar). Y nosotros queremos que cada panel del muro de ladrillos sea un único polígono.

En nuestra ayuda vienen el mapa de relieve y el mapa de normales. Veamos en qué consisten estos "mapas".

2.1. Mapa de relieve y mapa de normales.

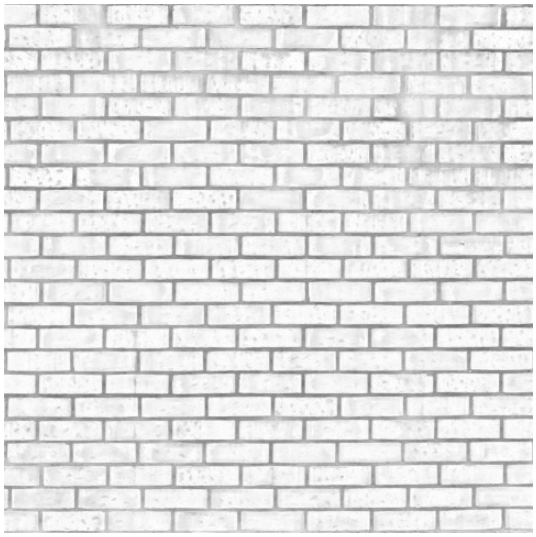
Partiremos de una textura Diffuse de los ladrillos de nuestro muro. Esto de la "textura diffuse" es la manera de nombrar de forma pomposa en este mundo del 3D a la textura de toda la vida que hemos tratado hasta ahora.



Ahora veamos el mapa de relieve o "Bump Map" de este "Diffuse Map".

El mapa de relieve es una hoja de textura con un único canal (gris) en la cual cada píxel informa de la "altura" que tendrá el punto del polígono sobre el que se aplique. Esta altura se entiende sobre la definida en el polígono con una variación en más o menos en la forma: el valor 128 (tono gris medio) no altera la altura del punto del polígono, los valores inferiores hasta 0 (color negro) representan variaciones en profundidad o puntos hundidos en la superficie del polígono, y los valores superiores hasta 255 (color blanco) representan variaciones en relieve o puntos elevados en la superficie del polígono.

Este sería un Bump Map del anterior:



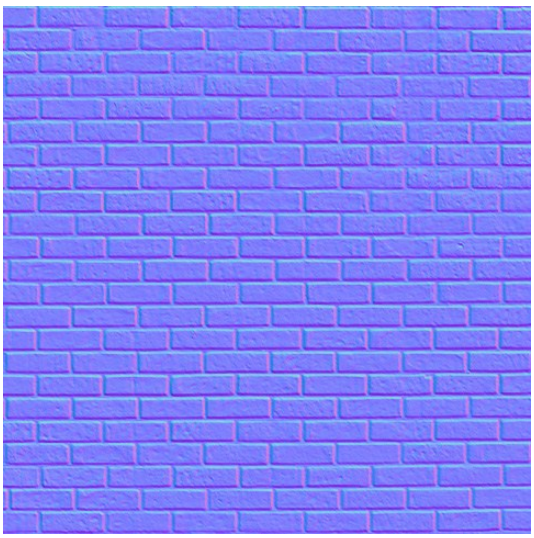
Se observan las superficies más claras (elevadas) y las más oscuras (hundidas).

Por último veamos un mapa de normales o "Normal Map" de esta misma textura de ladrillos.

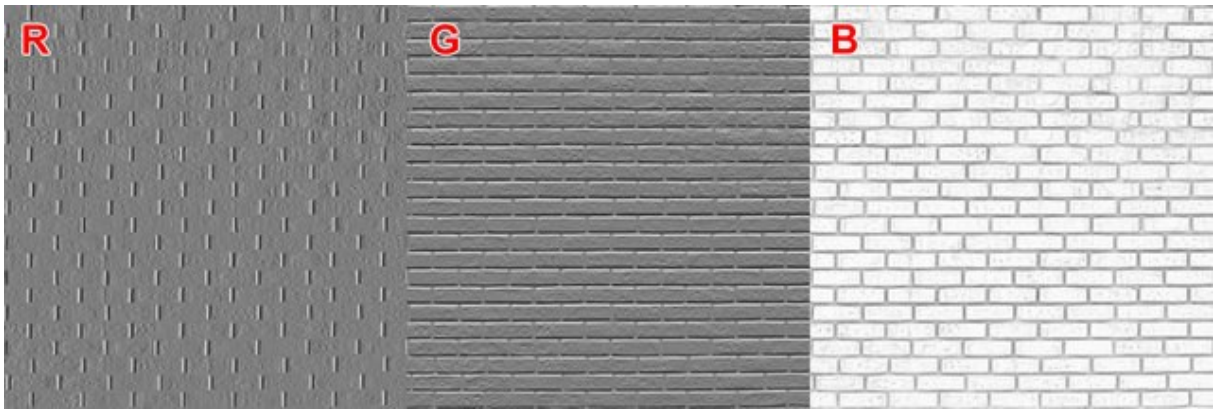
El mapa de normales es una hoja de textura con los tres canales RGB (rojo, verde y azul) de toda textura, en la cual cada píxel informa, no ya de la "altura", si no de la dirección en los tres ejes de un punto de la textura, o sea de la normal del punto. Dicho en palabras llanas: hacia donde mira el punto de la textura. Esta información se expresa en una cantidad en cada uno de los tres ejes de coordenadas, y donde el canal rojo de la textura (R) tiene la información de la dirección en el sentido Este-Oeste de la textura, el canal verde (G) tiene la informa-

ción de la dirección en el sentido Norte-Sur de la textura, y el canal azul (B) tiene la información en "altura" en el sentido hacia el espectador (el equivalente a un mapa de relieve).

Este sería un Normal Map de los ladrillos:



Y para apreciar el cómo está almacenada la información nos basta con observar los tres canales RGB por separado de este mapa:



Es decir, que un mapa de normales tiene la misma información que un mapa de relieve (la altura) complementada por la información de la dirección del punto (su normal). El motor gráfico de RW trabaja con mapas de normales, aunque popularmente se les denomina BumpMap aún y no ser esta denominación estrictamente adecuada.

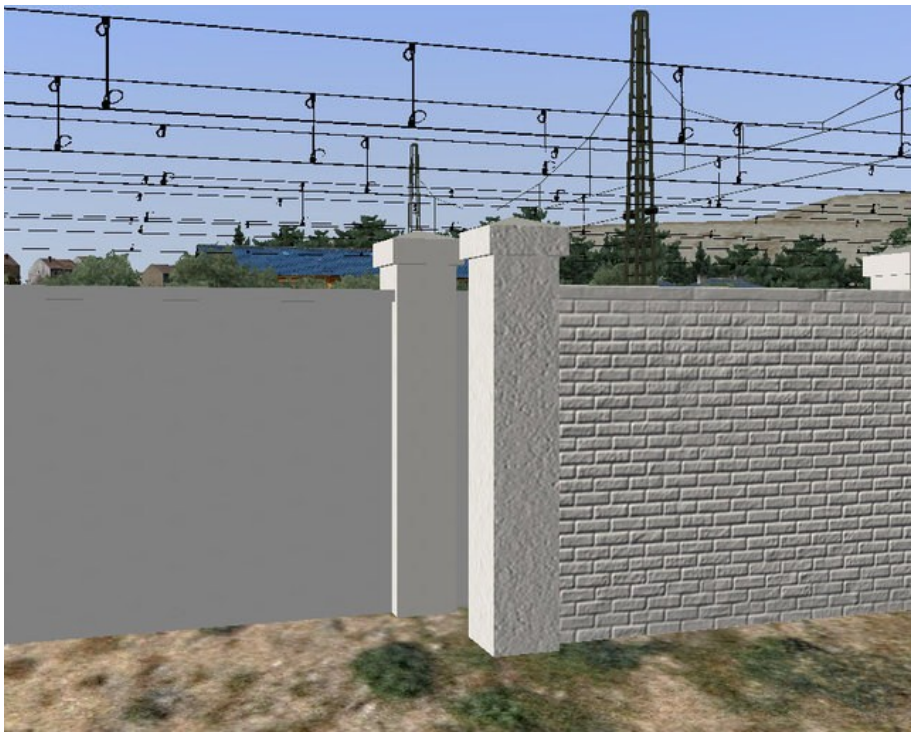
A título de curiosidad:

En un mapa de normales, mientras que para los canales rojo y verde se toma el valor 128 como neutro (0) y el rango de estos varía entre 0 y 255 (-128 y +127), para el canal azul el rango se admite únicamente entre 128 y 255 (entre 0 y +127) pues la normal define una semiesfera sobre el polígono base y los valores de altura negativos no tienen sentido. Esto le da a estos mapas el característico tono azulado, dado que el canal azul como poco siempre está presente en un valor mínimo de 128.

¿Y para qué quiere el motor gráfico estos mapas? Sencillo. Con un mapa de normales el motor gráfico puede variar la cantidad de luz que refleja un punto de una superficie plana según la hipotética orientación de este punto que está informada en el mapa de normales. Recordemos que en realidad estamos frente a un polígono totalmente plano y la geometría del mismo

no generará dicha información. Es decir, que podemos tener un único polígono para toda la pared de ladrillos (y no los miles y miles de polígonos que nos salían de haber modelado con detalle estos ladrillos) y sin embargo su aspecto ser muy semejante. ¿Increíble? Hasta cierto punto. Os muestro un ejemplo.

Este es el muro de ladrillos con una textura diffuse totalmente uniforme en gris. Evidentemente está construido con un sólo polígono plano y su luz es por tanto constante en toda su superficie. Frente al mismo he colocado otro muro idéntico, también con un sólo polígono, también totalmente plano, al cual además de la misma textura diffuse en gris le he aplicado un mapa de normales como el mostrado más arriba:



Aun siendo totalmente plano el muro de ladrillos, el motor del juego altera la cantidad de luz emitida por cada punto del mismo en virtud de la normal que está informada en el mapa de normales. También he generado mapa de normales para la columna de hormigón y para el remate de losas de piedra. El modelo, aún y no tener las texturas diffuse que representan ladrillos u hormigón, presenta satisfactoriamente la sensación de superficie rugosa o con hendiduras, y hasta podemos apreciar los ladrillos, sus irregularidades y diferentes alturas (hay ladrillos más hundidos en el muro y otros que sobresalen un poco más que el resto).

Dejémonos de teoría y pasemos a ver como podemos crear estos mapas y como implementarlos en el simulador... pero esto será en un nuevo capítulo.

2.2. Creando un mapa de normales.

Veamos como podemos obtener el mapa de normales de una textura que tengamos. En primer lugar deberemos proveernos de alguna de las utilidades que existen y que automatizan el proceso. Voy a mencionar tan sólo alguna de las más habituales, aunque cada cual podrá adaptarse a la que le proporcione mayor comodidad:

- NVIDIA publica unos plugins para Photoshop entre los cuales se encuentra el **NVIDIA Normal Map filter**.

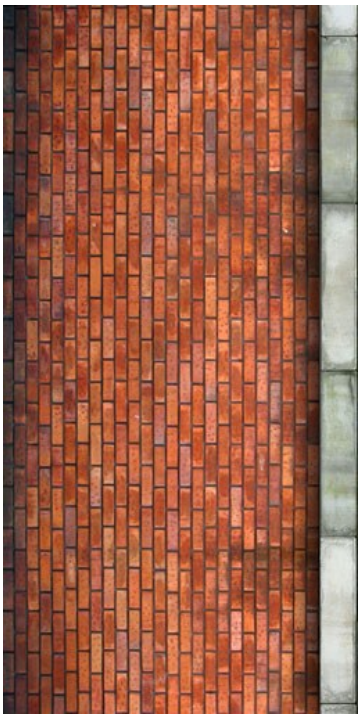
- Para GIMP también existe un plugin con la misma finalidad: **GIMP normalmap plugin**
- Por ultimo mencionaré **xNormal** una potente utilidad para tratamiento en general de texturas en modelado 3D, y cuya instalación incluye también plugins para Photoshop.

Personalmente uso el primero de los mencionados, aunque creo que xNormal tiene un gran potencial que aún no he descubierto.

Una vez instalada la utilidad que escojamos, su uso es sencillo en cualquier caso. No obstante, este ejemplo lo mostraremos sobre "NVIDIA Normal Map filter".

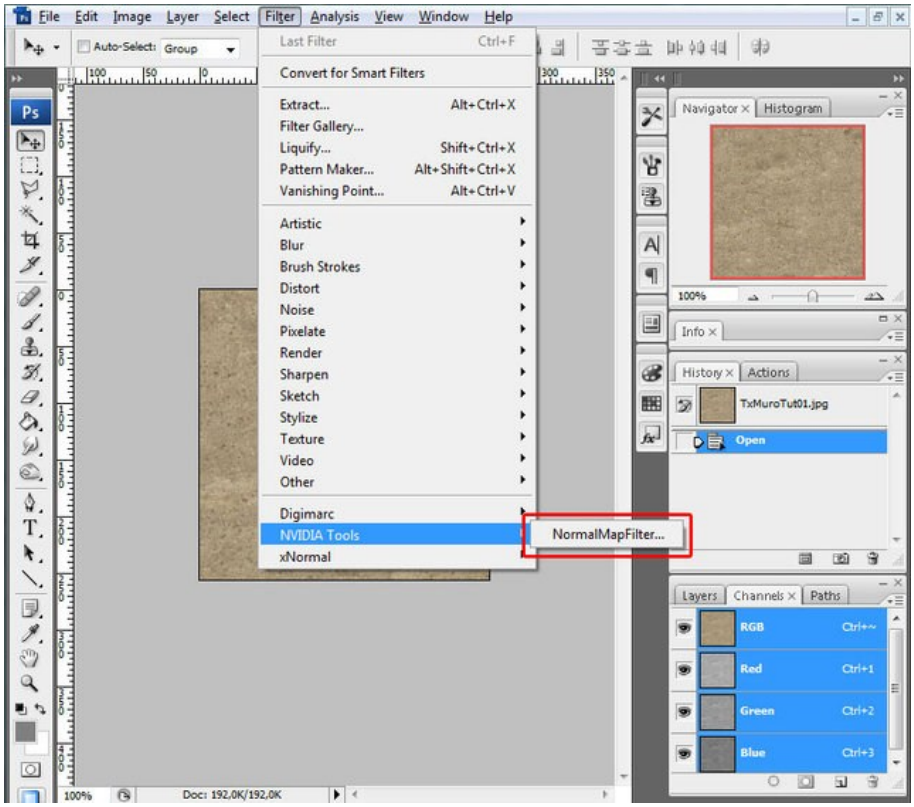
Vamos a plantearnos dar relieve a las superficies del muro que acabamos de crear, tanto al elemento procedural (el muro de ladrillos) como al elemento fijo (la columna de hormigón), con lo cual los métodos empleados nos cubrirán el aplicar relieve a cualquier superficie de cualquier objeto.

Recordemos las texturas que teníamos en el muro:

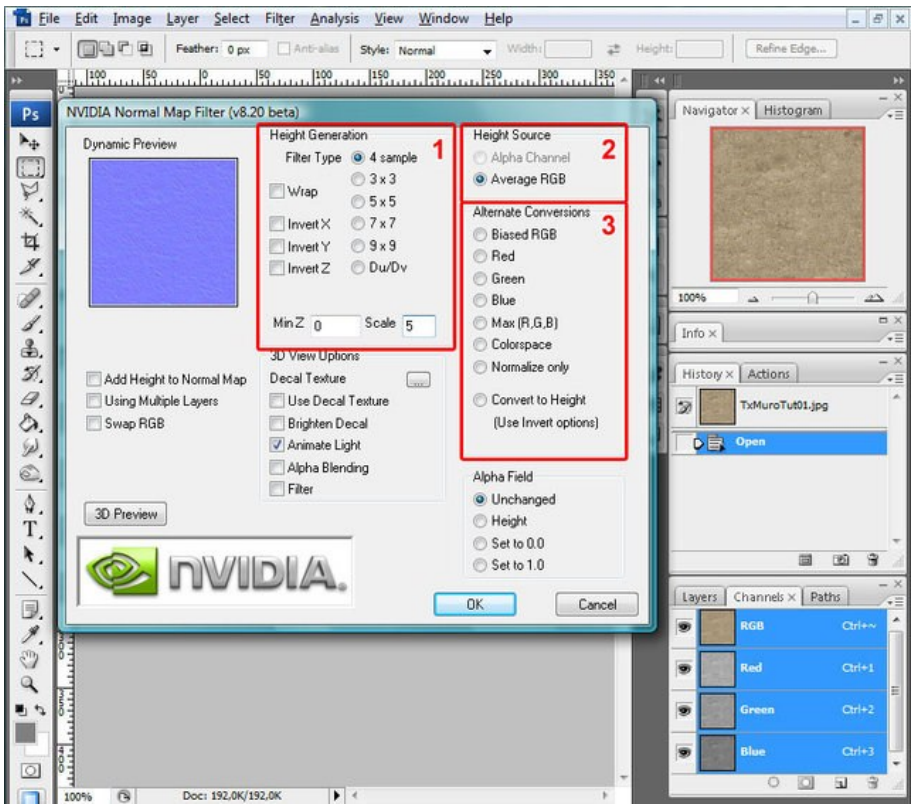


Empezaremos por una cualquiera de ellas, por ejemplo la de hormigón.

Abriremos la textura "TxMuroTut01.ace" con el editor de imágenes para, con la textura como imagen activa, encontrar el acceso al plugin. En mi caso, en Photoshop, lo encontraré bajo el menú "Filter", dentro del grupo "NVIDIA Tools":



El plugin actúa como un filtro más del editor sobre la imagen que la alterará en consecuencia. Lo ejecutaremos y aparecerá una ventana desde donde podemos parametrizar el efecto del plugin:



Observamos que, de izquierda a derecha, en primer lugar aparece una ventana con la previsualización del resultado. Claro que, como la ventana es muy pequeña, la visualización es apenas suficiente, pero no importa. Podemos pulsar el botón "3D Preview" para que nos abra

una nueva ventana donde se visualiza un polígono con el mapa de normales aplicado, pero tampoco perdáis mucho tiempo con ella, mejor pasamos a ver que modificaciones son interesantes en los parámetros proporcionados.

El primer grupo, "**Height Generation**" tiene bastante interés. En él encontramos

- "Filter Type", que nos da a escoger entre 6 opciones que determinarán el tamaño de la área de trabajo que usará el filtro para calcular las direcciones de los puntos. El tamaño ideal variará dependiendo a su vez del tamaño de la textura. Para una textura pequeña como la nuestra la opción "4 sample" será la mejor. Si escogemos un tamaño mayor, p.e. 7x7 obtendremos un mapa de normales suavizado, semejante al efecto de aplicar un filtro "Blur" (desenfoque) a una imagen. Ahora bien, si nuestra textura fuera de un tamaño grande, digamos 2048 píxels, al escoger un tamaño grande, como 7x7, no sólo no nos aparecería este efecto de blur, si no que ayudaría a eliminar un cierto "ruido" de fondo en la imagen resultante y por tanto sería aconsejable.
- "Invert X", invierte la dirección de las normales calculadas para el canal R (este-oeste). No suele ser necesario, pero si la imagen de partida no tiene una iluminación uniforme y nos parece que no se genera correctamente esta orientación podemos usar esta opción.
- "Invert Y", igual que el anterior, invierte la dirección de las normales calculadas pero para el canal G (norte-sur). No suele ser necesario.
- "Invert Z", invierte el canal B (altura) provocando que las zonas hundidas sobresalgan y viceversa. No suele ser tampoco necesario.
- "MinZ", fija el valor mínimo para el canal B. El valor estándar es cero, pero en algunos casos especiales ciertos motores gráficos admiten valores inferiores hasta -128. Deberemos dejarlo en 0.
- "Scale", determina un factor de falsa altura a generar en el mapa de normales. Variará según el efecto de volumen que queramos conseguir, más o menos acusado, aunque el valor 5 suele ser suficiente. Si queréis podéis probar con 10, aunque para nuestra textura de hormigón no sería lo apropiado (los agujeros de su superficie no son muy profundos).

Bajo este grupo que hemos visto encontramos el grupo "3D View Options", que afecta únicamente a cómo se genera el "3D Preview" del plugin. No deben importarnos por tanto sus valores.
El segundo grupo, "**Height Source**" determina el origen de la información para obtener el mapa de normales, y se combina con el tercero, situado bajo él. En él encontramos

- "Alpha Channel", que indica al filtro que tenga presente la información del canal alpha de la imagen para la generación de las "alturas". Pero como en nuestro caso la imagen de partida no dispone de canal Alpha, no está disponible.
- "Average RGB", determina que las "alturas" se calculen a partir de un valor promedio de los tres canales de color de la imagen original (R=rojo, G=verde y B=azul). Dicho promedio, pasado a una escala de grises, determinará que los puntos que presenten un valor bajo (oscuros) estén más profundos que los puntos que presenten un valor alto (claros).

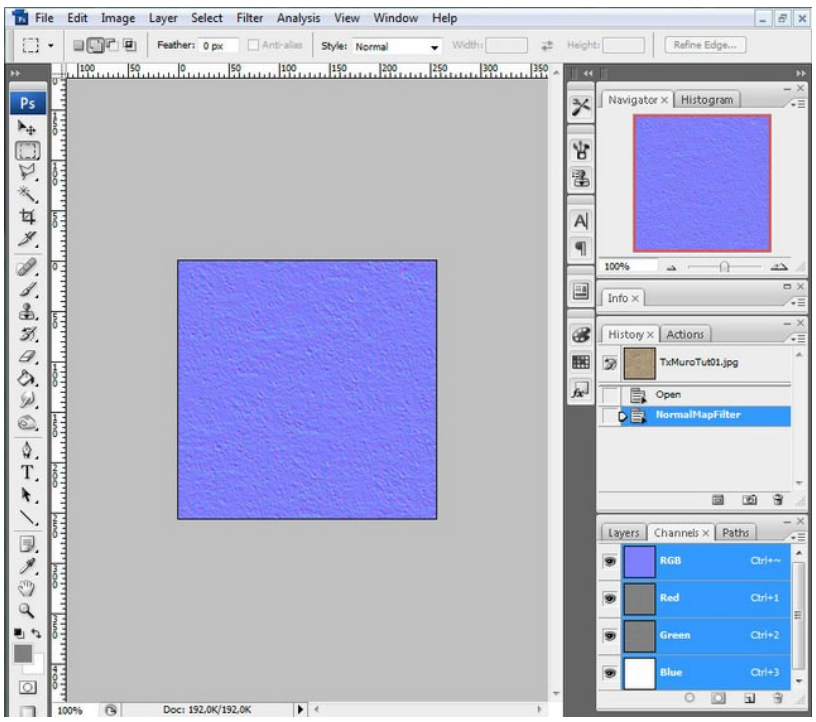
El tercer grupo, "**Alternate Conversions**" desactiva las opciones anteriores y permite calcular las alturas en base a otros factores:

- "Biased RGB", usará los tres canales de color de la textura, pero normalizándolos (suavizando) sus diferencias antes de calcular su promedio. Esto suele dar lugar a mapas de normales más uniformes, sin demasiadas diferencias.
- "Red", usará tan sólo la información del canal rojo de la imagen original para obtener el mapa de normales. En el presente caso de la textura de hormigón, que tiene un color muy neutro, no tiene mayor importancia. Pero para la textura de la pared de ladrillos, en que domina claramente el color rojo, con esta opción se obtendrá un mapa de normales más preciso.
- "Green", actúa como en el caso anterior, pero usando tan sólo el canal verde para obtener el mapa de normales.
- "Blue", actúa como en los casos anteriores, pero usando tan sólo el canal azul para obtener el mapa de normales.
- "Max (R,G,B)", para cada punto selecciona el mayor de los valores de los tres canales RGB (rojo, verde y azul) para obtener el mapa de normales.
- "Colorspace", añade al valor medio de los puntos la información de saturación del color como información también de la altura.
- "Normalize Only", esta opción no generará un mapa de normales, si no que modificará la imagen original para eliminar el ruido que pudiera tener y ajustar los colores a una operación posterior. La suelo usar como paso previo a la generación del mapa de normales, el cual, de esta manera, recoge mejor y con mayor definición la información de alturas. Es opcional según los gustos de cada cual.
- "Convert to Height", esta opción tampoco generará un mapa de normales, si no un mapa de alturas, pero de una forma muy pecualiar que no le acabo de encontrar la gracia. Igual la tiene.

El último grupo "Alpha Field" determina la información que el filtro cumplimentará en el canal Alpha, caso de que la imagen original lo tuviese. En nuestro caso es intrascendente.

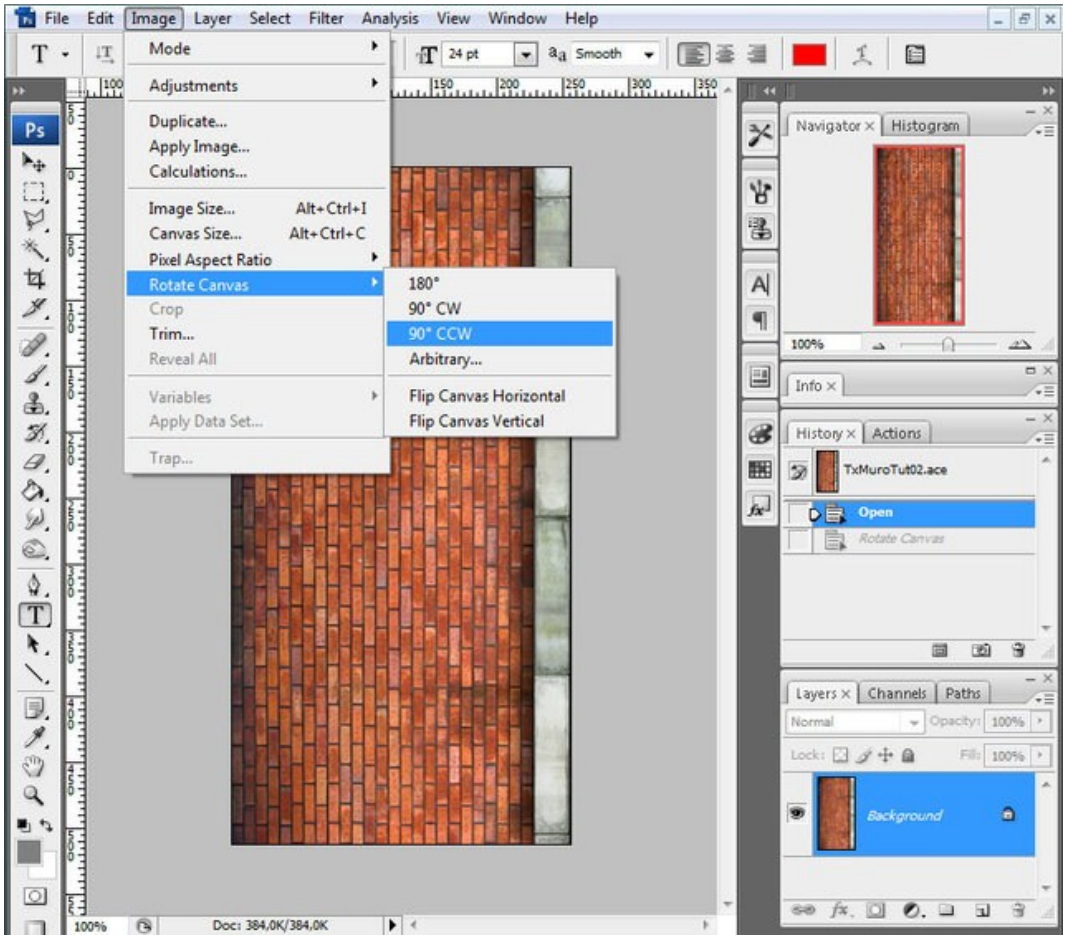
Por tanto, y visto lo visto, elegiremos los valores que os muestro en la imagen anterior y pulsaremos el botón OK. Cuesta mucho más explicarlo que ejecutarlo, por lo que os pido que no os dejéis impresionar par tanta letra puesto que la herramienta es muy sencilla de usar.

Como resultado del filtro que acabamos de ejecutar obtendremos esta imagen:

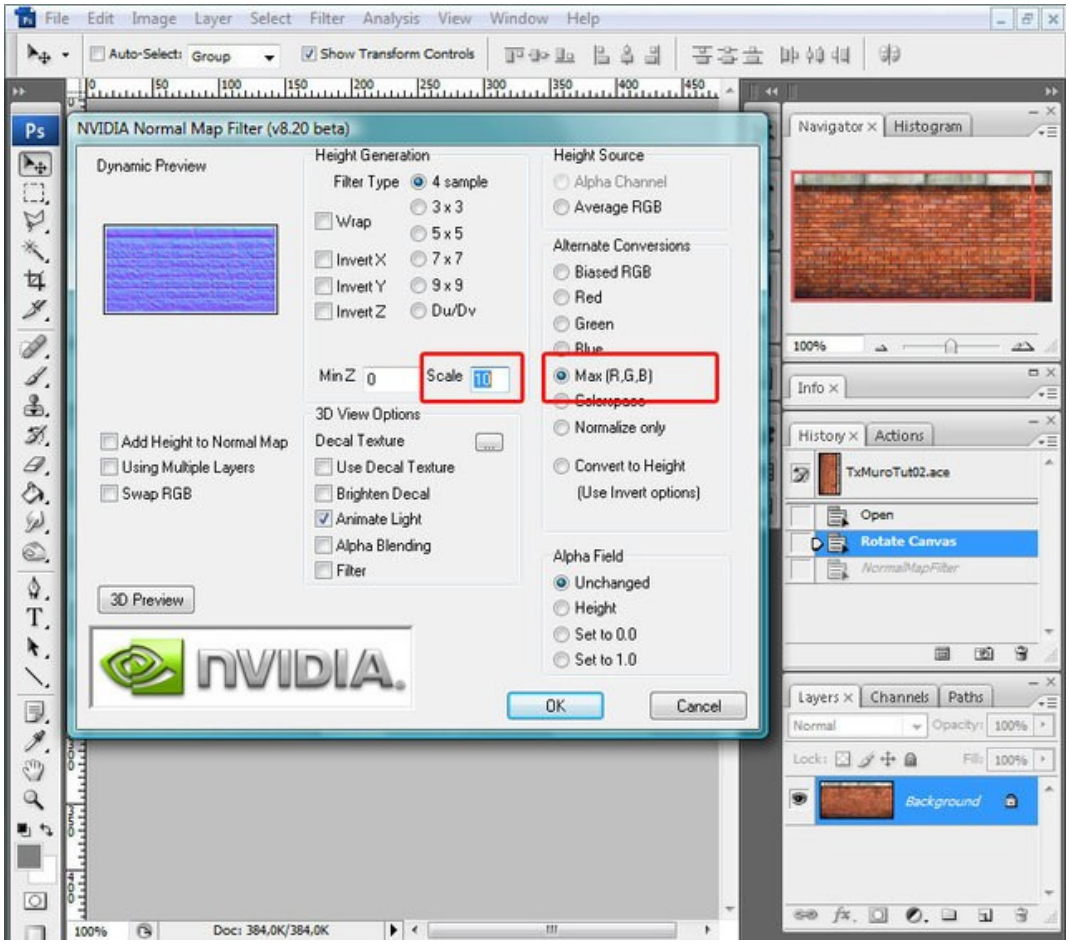


Procederemos a guardarla con el mismo nombre que tenía la textura original de hormigón, pero añadiendo el sufijo "**_nm**", abreviatura de "normal", pero que en realidad evita que el exportador realice una compresión DTX a la textura, la cual podría deteriorar la información de las normales.

Con el caso de la textura de ladrillos actuaremos de igual manera, pero con unos pequeños cambios. Como esta es una textura para un objeto Loft, recordemos que la tenemos girada 90º para su correcta aplicación en mosaico a lo largo del elemento procedural. En este caso, por tanto, una vez cargada la textura original del muro de ladrillos le deberemos efectuar un giro de 90º para recuperar la posición horizontal:

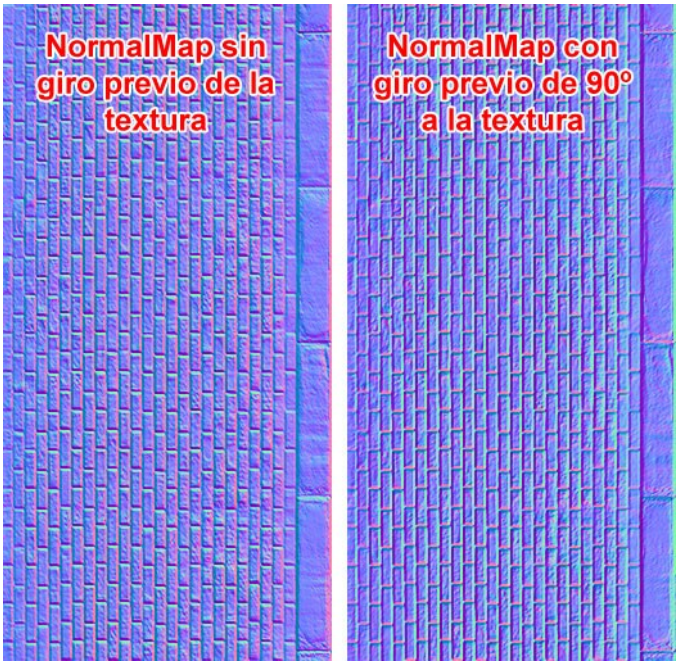


Luego podemos invocar el filtro NormalMap. Aunque los parámetros que hemos usado anteriormente con la textura de hormigón son genéricos y suficientes en muchos casos, en el presente querremos que el relieve de los ladrillos sea superior al del hormigón, y por tanto aplicaremos estos otros:



Una vez obtenido el mapa de normales de esta otra textura procederemos a girarla nuevamente de forma que coincida en su posición con la textura diffuse del muro y la guardaremos con el nombre "TxMuroTut02_nm.ace", es decir añadiendo el sufijo _nm como en el caso anterior.

¿Es necesario este giro a la textura para generar el mapa de normales en una posición determinada? Me he hecho esta pregunta y no tengo una respuesta del todo clara, pero he comprobado que los mapas generados no son iguales con el giro que sin el giro:

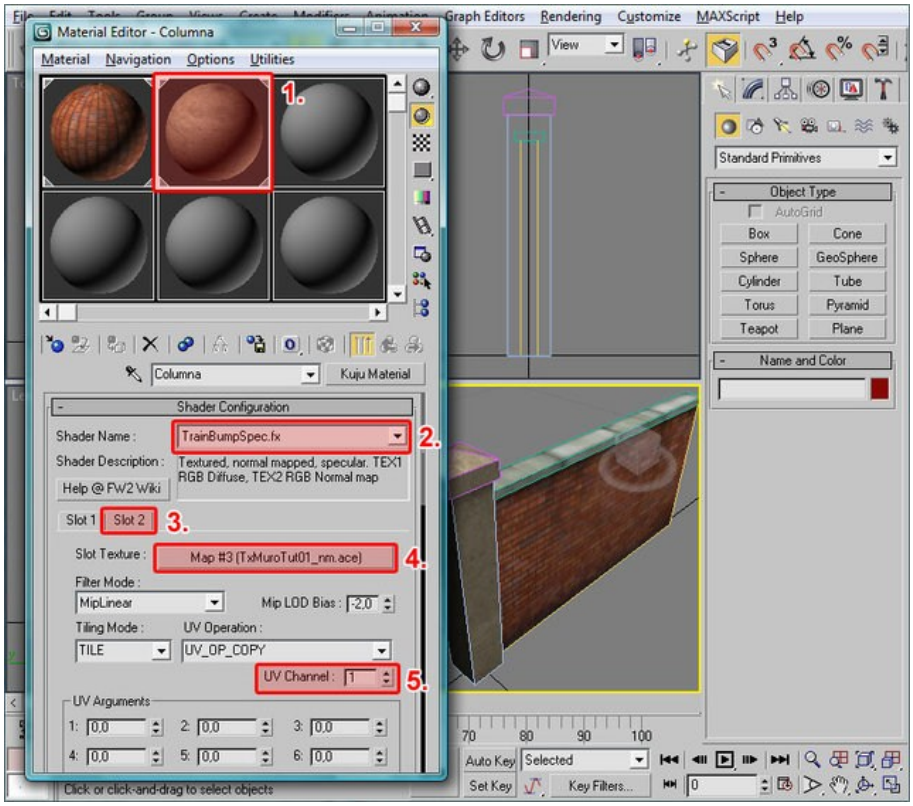


Está claro que aunque la información básica es la misma, los canales X e Y (el rojo y el verde) están invertidos en un caso respecto al otro. Como la operación cuesta muy poco mi consejo es que realicéis dicho giro, pues el resultado es satisfactorio y algo en mi interior me dice que además es necesario.

2.3. Uso de shaders con Bump en RW. TrainBumpSpec.fx y LoftBump.fx

Una vez generadas las texturas de los mapas de normales que acompañaran a las texturas diffuse, veamos como le decimos a nuestro modelo en 3ds Max que las use correctamente.

Cargaremos nuestra escena 3ds con el muro y abriremos el editor de materiales. Modificaremos en primer lugar el material "Columna" (la segunda bola) con la textura de hormigón:

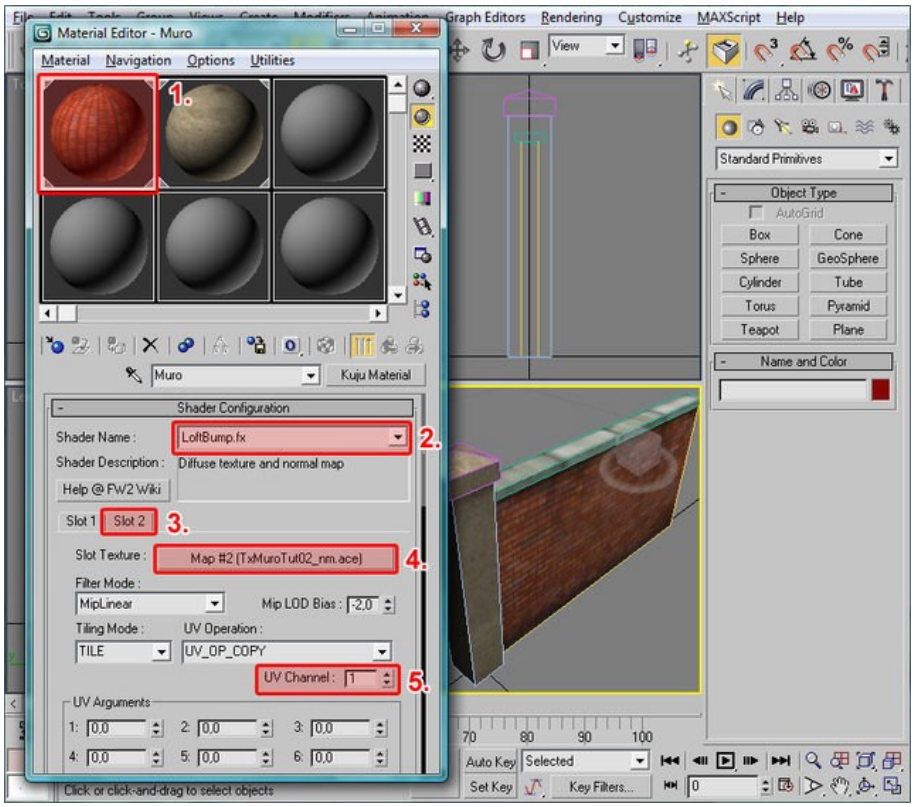


Y efectuaremos unos pequeños cambios:

1. Verificamos que tenemos seleccionado el material "Columna"
2. Cambiaremos el shader por el "**TrainBumpSpec.fx**". Este shader permite trabajar con dos mapas de texturas, en el slot 1 una textura RGB Diffuse donde dejaremos la existente "TxMuroTut01.ace", y en el slot 2 una textura con un mapa de normales. Además, este mismo shader permite añadir opcionalmente información sobre un mapa specular del material (brillo de superficies metálicas o pulidas), pero esto lo veremos en otro capítulo y no lo necesitamos en nuestra columna de hormigón.
3. Seleccionamos el Slot 2 para añadir la nueva información a este material.
4. Indicamos que este slot usará la textura con el mapa de normales del hormigón: "TxMuroTut01_nm.ace".
5. Nos aseguraremos que el canal de mapeo para esta nueva textura sea el 1, el mismo que para la textura Diffuse, de esta forma no tenemos que remapearlo.

Y esto es suficiente para un objeto genérico (edificio, vehículo, locomotora, etc...) como en este caso la columna.

Seguiremos ahora con la textura para el elemento procedural del muro:



Y efectuaremos también unos pequeños cambios, que en esencia son los mismos que en el caso anterior:

1. Verificamos que tenemos seleccionado el material "Muro"
2. Cambiaremos el shader por el "**LoftBump.fx**". Este shader [(2)], específico para elementos procedurales, también permite trabajar con dos mapas de texturas, en el slot 1 una textura RGB Diffuse donde dejaremos la existente "TxMuroTut02.ace", y en el slot 2 una textura con un mapa de normales.
3. Seleccionamos el Slot 2 para añadir la nueva información a este material.
4. Indicamos que este slot usará la textura con el mapa de normales del muro: "TxMuro-Tut02_nm.ace".
5. Nos aseguraremos que el canal de mapeo para esta nueva textura sea el 1, el mismo que para la textura Diffuse, de esta forma no tenemos que remapearlo.

Y con ambos cambios realizados podemos cerrar el editor de materiales y proceder a exportar nuevamente tanto el elemento procedural, el muro, como el elemento fijo, la columna. Tras pasar por el Blueprint Editor para generar nuevamente el objeto podemos apreciarlo en el simulador con sus flamantes mapas de normales:

[2] El shader LoftBump.fx, a diferencia de otros shaders para objetos procedurales, renderiza los polígonos por una sola cara. En algunas ocasiones, principalmente en los polígonos orientados hacia arriba, esta renderización se efectúa erróneamente produciéndose en la cara contraria a la deseada, en cuyo caso, para resolver el problema, podemos separar el polígono afectado del resto del procedural y girarlo 180º para que pueda ser visto en el simulador (hay que recordar resituar en el origen de la escena el pivote del nuevo elemento creado con el polígono afectado).

Nuevamente mantengo en la captura el muro sin mapas de normales, en segundo término, y con los mapas de normales, en primer término. Insisto que en ambos la geometría del modelo es exactamente la misma, y sus texturas diffuse también son idénticas.

En resumen, la mejora del aspecto de las texturas es evidente en lo referente al volumen de la superficie de los materiales, no habiendo incrementado en absoluto los polígonos de la malla de los objetos para conseguir esta sensación de volumen.

¿Y en qué casos es recomendable esta técnica? En superficies de objetos que estará relativamente cerca de la cámara y presentan materiales de un cierto relieve: paredes de ladrillo, muros de piedra, tejados de cualquier material, balasto en las vías, etc. Y en vehículos ferroviarios, cuando el relieve de la superficie no es excesivo: planchas metálicas, frisos de madera de coches y vagones, techos no lisos, etc. Y en general cuando queramos, lo probemos y veamos que el resultado nos merece la pena, porque aplicarlo es muy sencillo, y eliminarlo también si no nos convence.

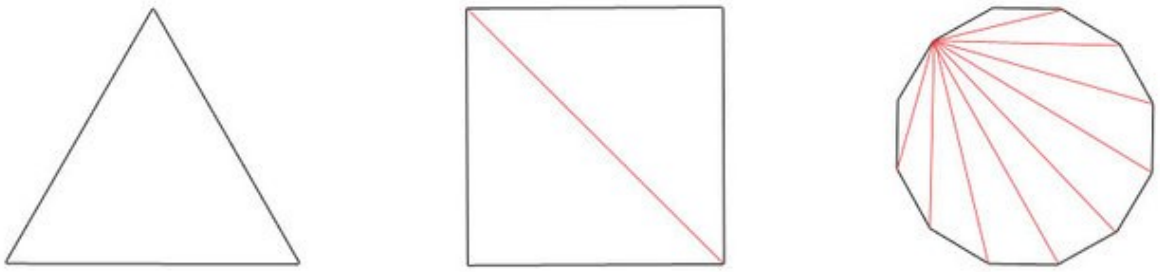
3. Poligonaje de objetos

"Un objeto complejo tiene más polígonos que varios objetos simples que definan la misma forma."

Pensemos en esta afirmación por un momento. Nuestra intención es hacer modelos que luzcan, cuanto más mejor, y con gran definición de detalles, cuantos más mejor también. Pero esto lo debemos llevar acabo sin merma del rendimiento del motor gráfico, pues estamos modelando para un simulador que debe renderizar la escena en "tiempo real", y uno de los consumidores del rendimiento del motor es el número total de polígonos de nuestro modelo. Veamos, pues, como controlar el crecimiento del número de polígonos que se produce en nuestras creaciones cuando les damos un mayor detalle.

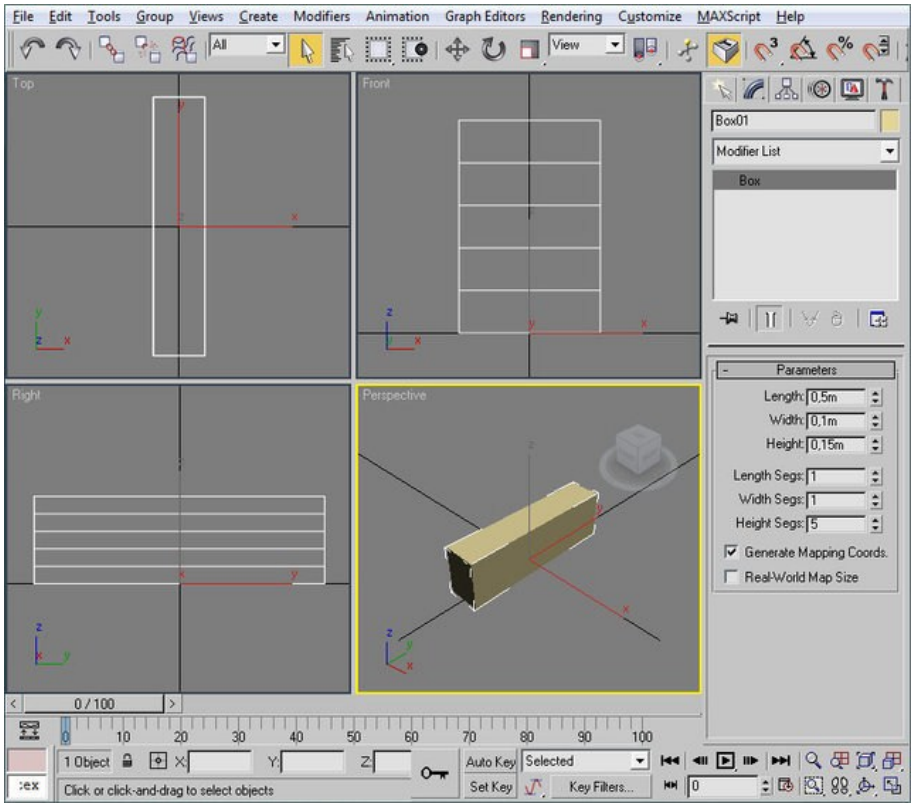
Una aclaración previa: Dada una cara de múltiples lados en un objeto, el total de polígonos (triángulos) necesario para representarla es, como mínimo, igual al número de lados de la cara menos dos. Hay que tener presente que las tarjetas gráficas trabajan con triángulos cómo unidad de representación en 3D.

Es decir, para representar un triángulo (3 lados en la cara) se necesita 3 - 2 = un polígono. Normal Para representar un cuadrado (4 lados en la cara) se necesitan 4 - 2 = dos polígonos (triángulos), y así sucesivamente nos encontraremos que si definimos un cilindro con una base de 12 lados para representar una de las bases se necesita 12 - 2 = diez polígonos.



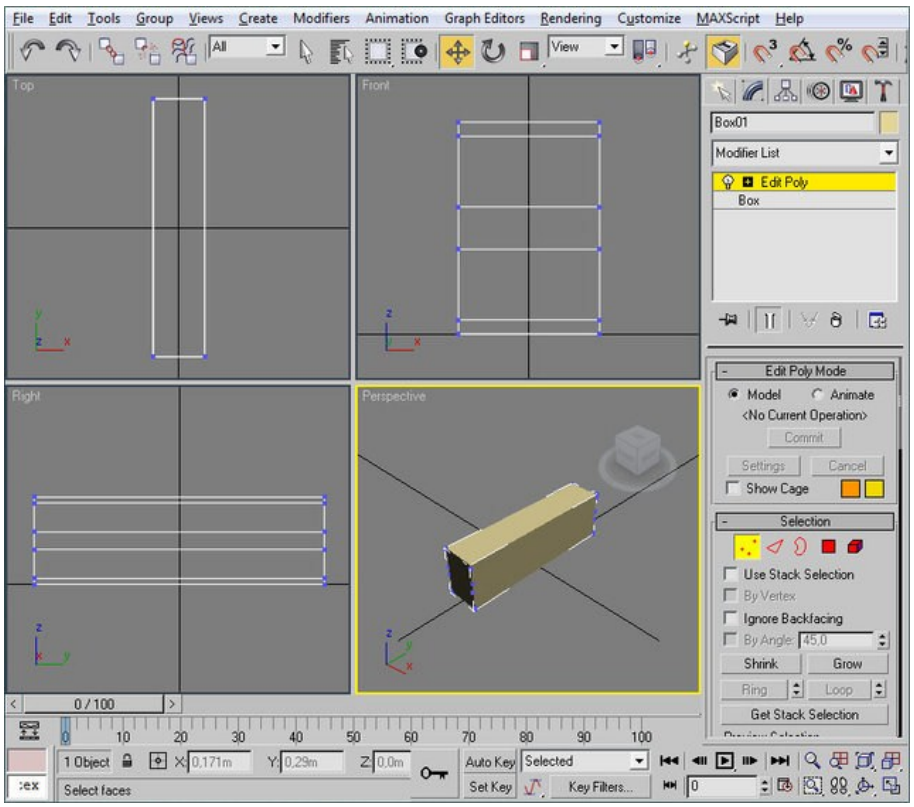
Vamos a crear ahora una viga con forma de H, de un metro de longitud y 10x15 centímetros de sección.

Podemos realizar la viga creando un cubo de 1 x 0,10 x 0,15, definiendo 5 secciones en altura:

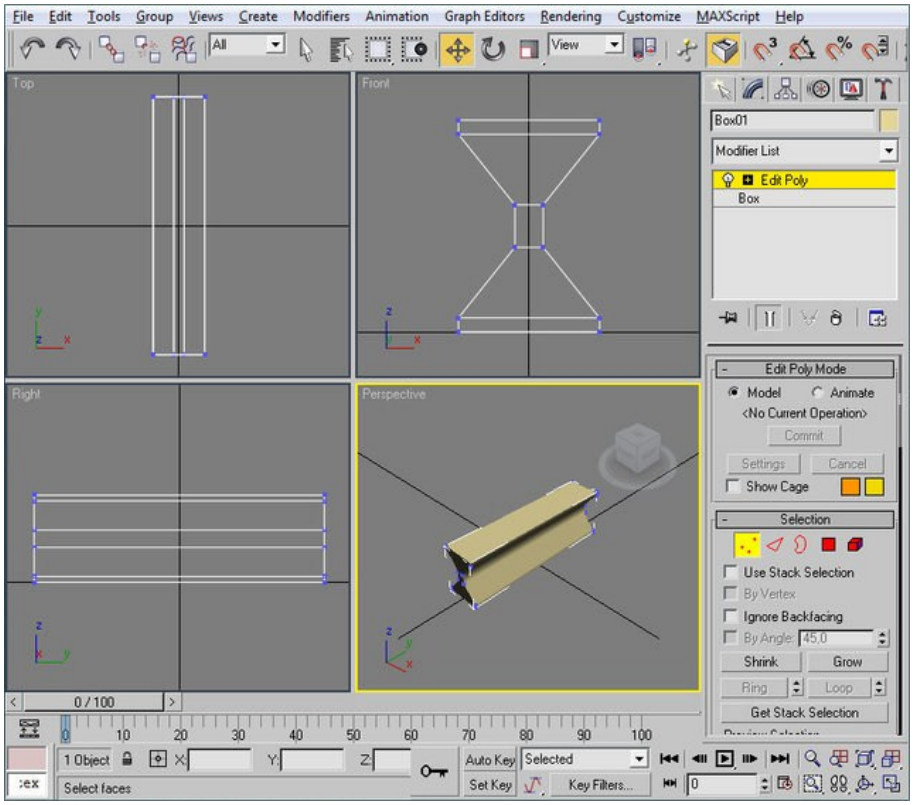


Un objeto de estas características está compuesto por 44 polígonos (triángulos): 10 en la cara anterior (cinco cuadrados de dos triángulos cada uno), otros 10 en la cara posterior, 10 más en un lateral, otros 10 en el otro lateral, 2 en la cara superior (un sólo cuadrado) y otros 2 en la inferior.

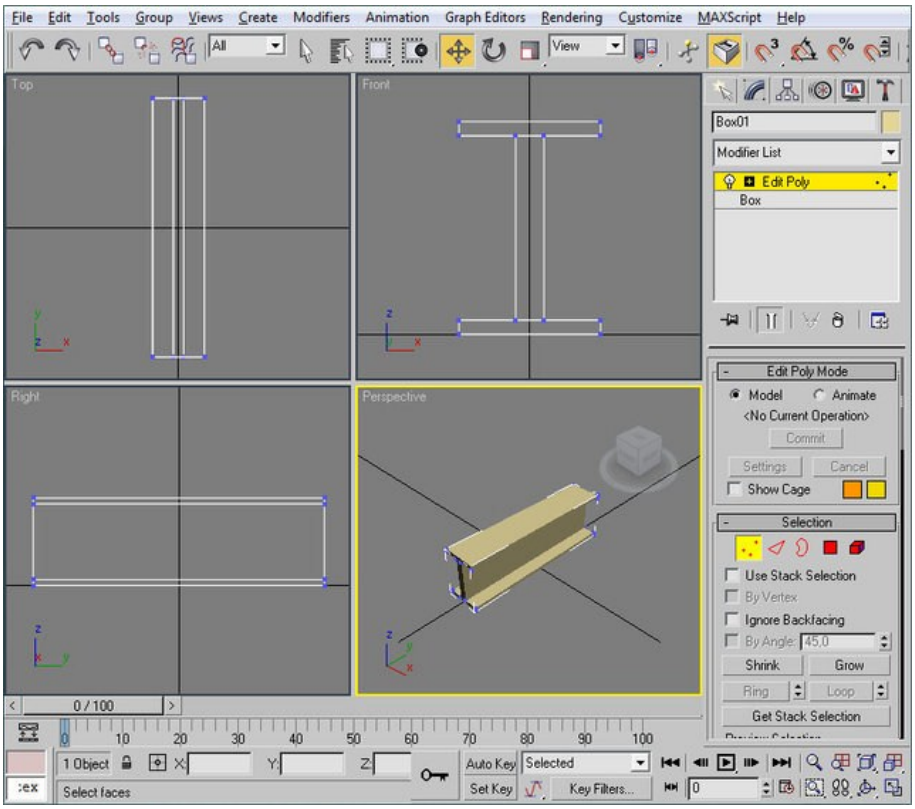
Si movemos las secciones segunda y penúltima a 1 centímetro de los extremos:



Desplazamos las dos centrales hacia el eje dejando 2 centímetros entre ellas:



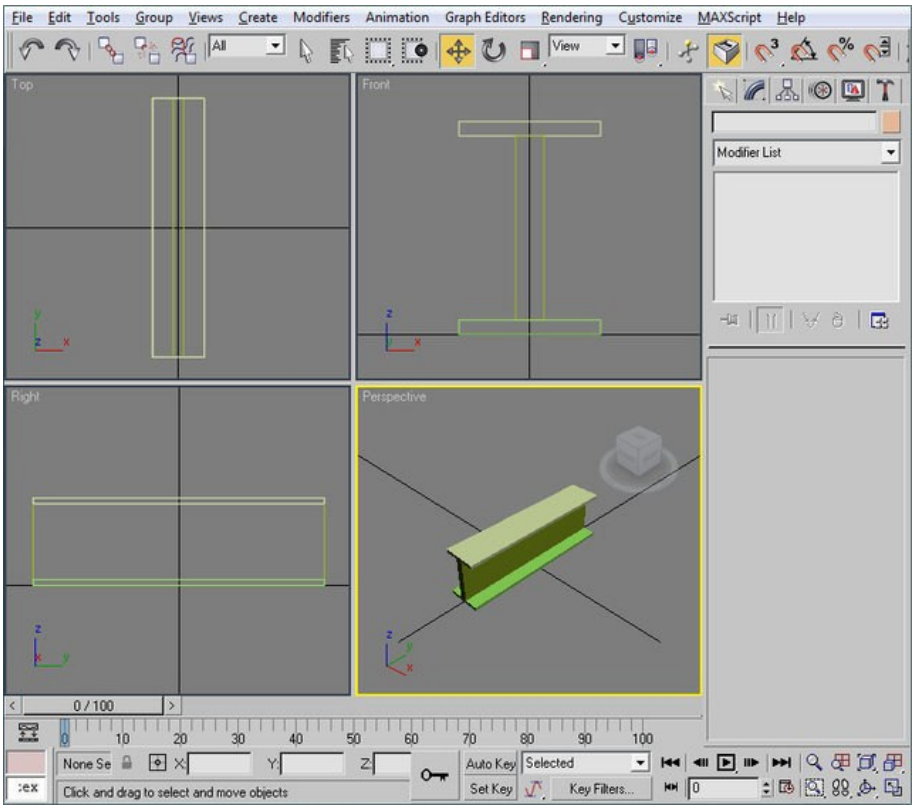
Y luego las movemos también a 1 centímetro de los extremos como hicimos con las primeras:



Obtenemos la forma definitiva que deseábamos.

Este mismo elemento lo podíamos haber obtenido por sustracción booleana de los espacios laterales de la viga a un cubo, pero el total de polígonos obtenidos al final sería el mismo, 44 como mínimo (ciertas operaciones booleanas tienden a generar polígonos extras en la forma resultante).

Ahora bien, en este caso, que realmente es muy sencillo, podríamos haber planteado la viga como un conjunto de tres cubos, en lugar de un único cubo deformado. Os muestro el caso donde los perfiles superior e inferior son dos cubos y el alma central otro cubo:



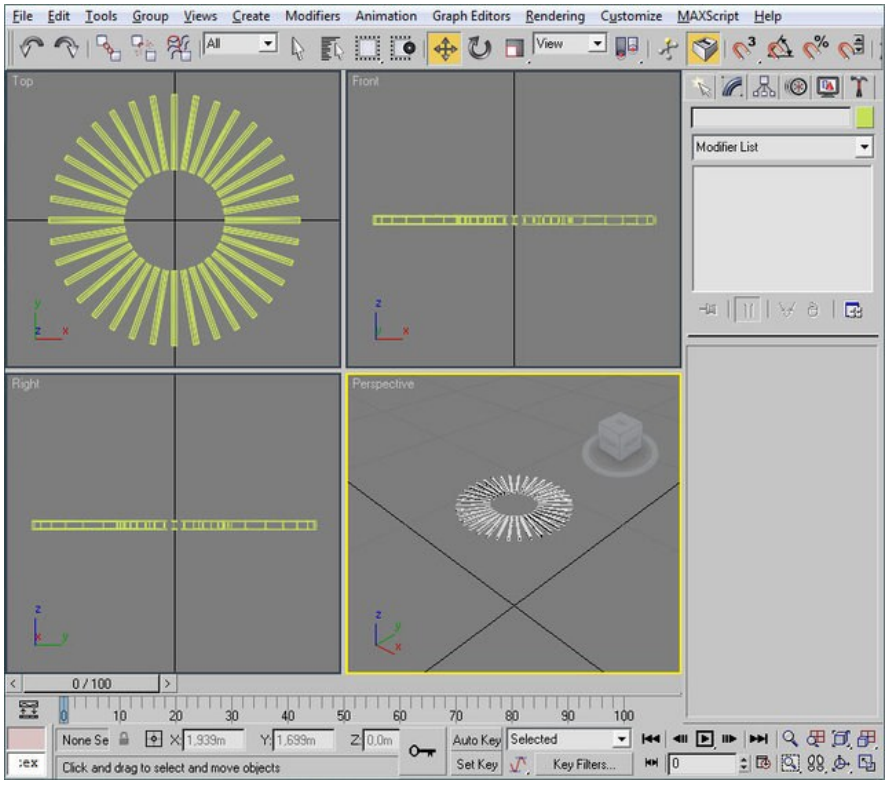
El resultado es idéntico al anterior, pero la complejidad de cada elemento es muy inferior y el número de polígonos totales también.

En efecto, un cubo simple tiene siempre 6 caras cuadradas, y por tanto 12 polígonos (2 por cara), no importando las dimensiones que pueda tener el cubo. Por tanto, la suma de los tres cubos serán 12 x 3 = 36 polígonos en total, 8 menos que en el mejor supuesto del caso anterior donde la viga era un único elemento algo más complejo. Además, en el cubo central que compone el alma de la viga podemos prescindir de las caras superior e inferior del mismo, pues no pueden ser vistas en ningún caso, y eso son 4 triángulos menos en el total, que queda reducido a 32.

Hemos reducido de esta forma las necesidades de recursos de la viga en algo más de un 27%.

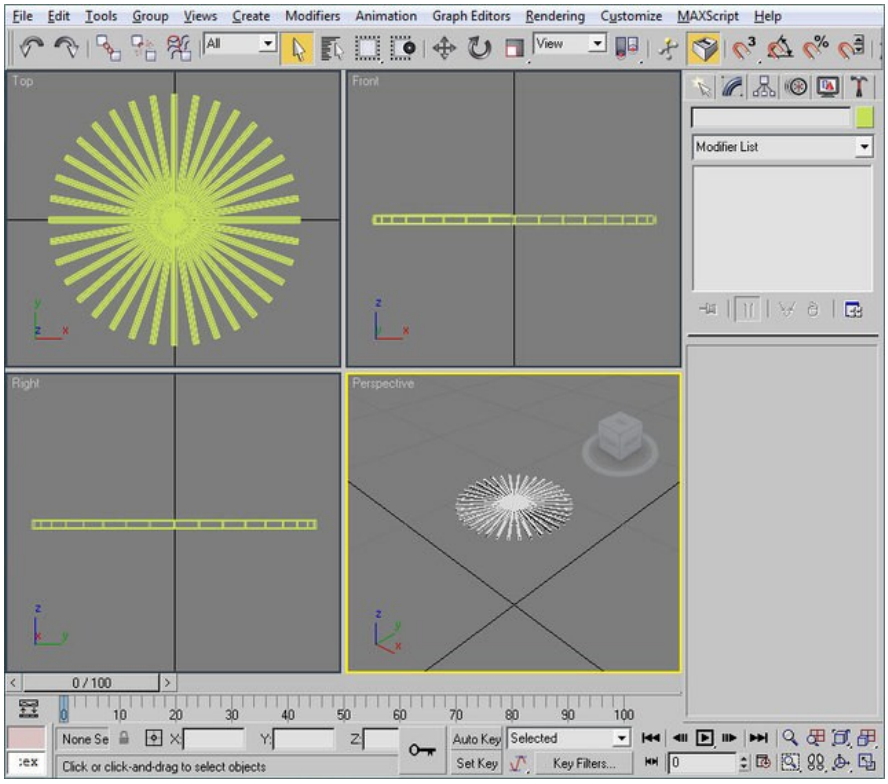
Me diréis que en total han sido 8 polígonos menos en la viga, y que aunque represente el 27% de ganancia, 12 polígonos siguen siendo despreciables, pero hemos de pensar que nuestros objetos no serán los únicos presentes en una escena del simulador, que una ruta puede estar llena de objetos y que en ella pueden estar evolucionando gran número de vehículos y composiciones. Y la renderización de todo ello ha de ser en tiempo real, o sea un mínimo de 25 imágenes por segundo, lo cual es en sí todo un reto pues si bien un segundo de renderización para toda una escena en 3ds Max es totalmente despreciable, en la simulación pasa a ser totalmente inaceptable.

Además, resulta que esta viga la necesitaba para componer con un conjunto de ellas una base formada por vigas de 2 metros de longitud situadas en círculo a razón de una cada 10º de giro, en total 36 vigas:



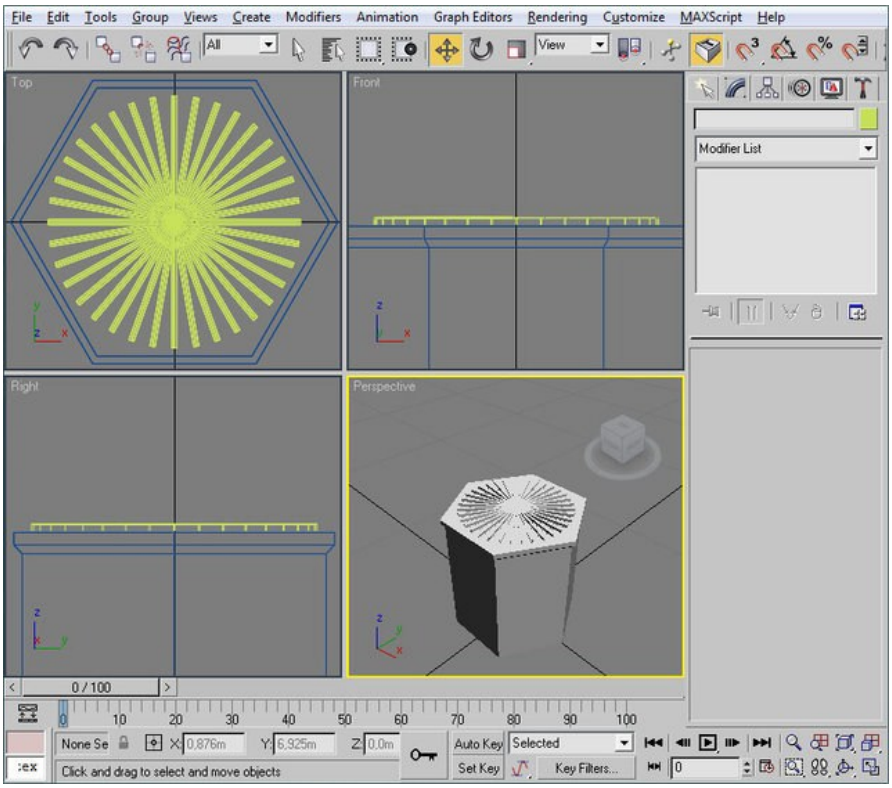
Esto da un total de 36 x 32 = 1.152 polígonos en esta base, frente a los 36 x 44 = 1.584 sin la optimización, con lo que el ahorro lo podemos elevar hasta los 438 polígonos.

Esta base de vigas la usaré en la construcción de un cubato, y por tanto irá sobre una base de mampostería y sobre ella se situará la cuba del mismo. Por tanto, como únicamente van a ser visibles los finales exteriores de las vigas, podemos simplificar el conjunto haciendo que las vigas sean de 5 metros de longitud, que es el diámetro de la cuba, y por tanto una misma viga irá de extremo a extremo de la base mostrando ambos finales:

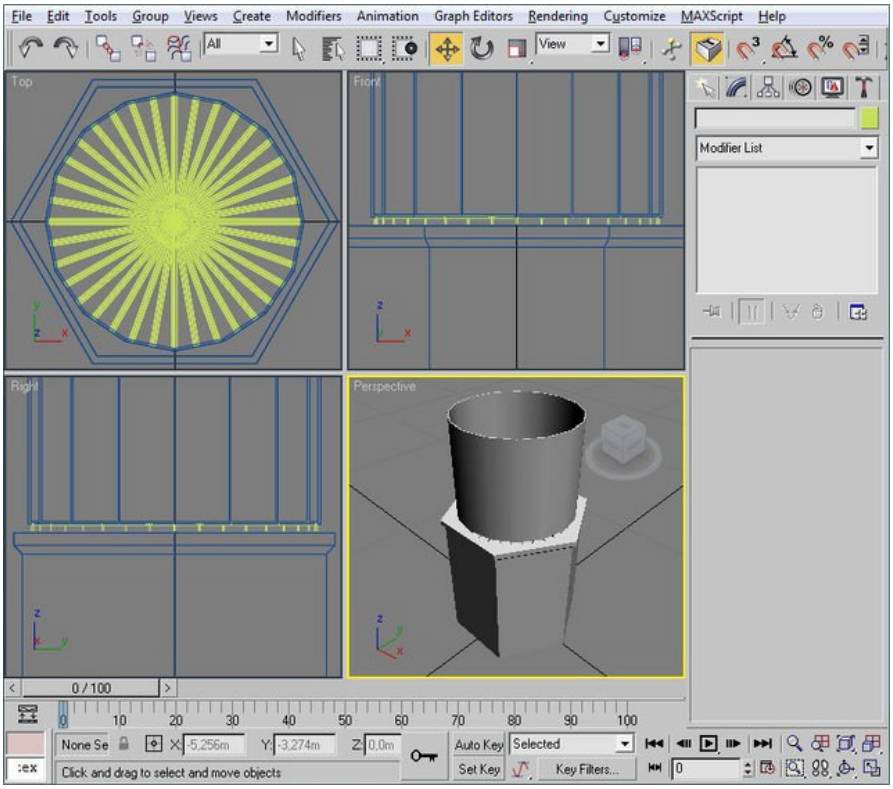


Esto nos permite reducir el número de las vigas a la mitad, dado que cada una aporta los extremos de dos de ellas, sin que el resultado visible final quede alterado, lo cual nos reduce a la mitad el total de polígonos de esta base al necesitar tan sólo 18 vigas para el mismo efecto. Y 18 x 32 = 576 polígonos totales.

Vemos, en efecto, que el interior de las vigas no será visible y por tanto no queda afectado el modelo. Ponemos las vigas sobre la base de mampostería:

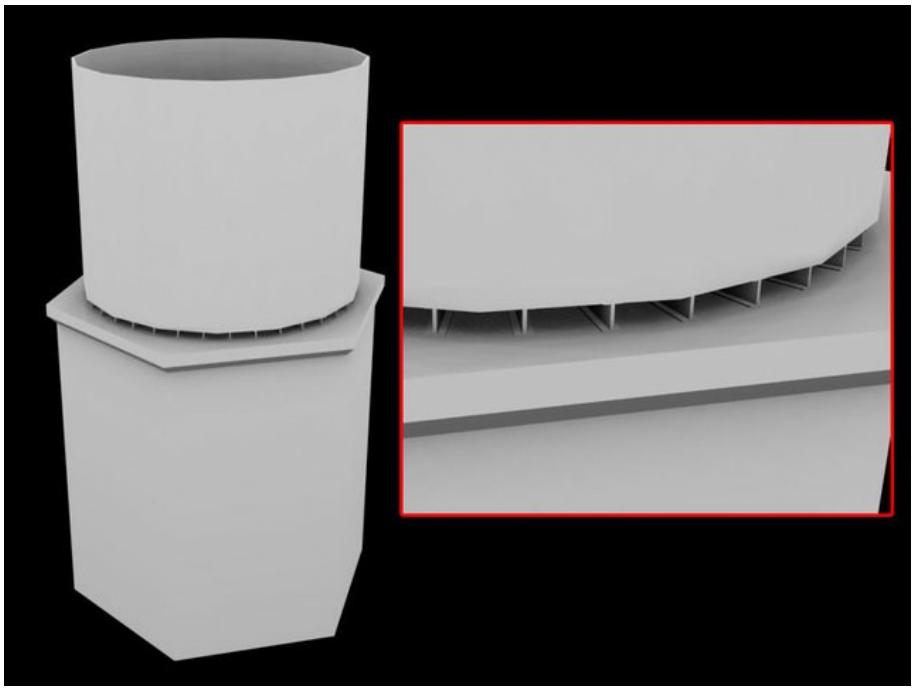


Y sobre ellas ubicamos la cuba:



Además, a este elemento podemos darle una distancia de visión de, pongamos 50 metros, de esta forma cuando nos acerquemos al cubato podremos apreciar el detalle de las mismas, pero al alejarnos, cuando éstas no podrán ser visibles, el modelo no las mostrará ahorrando así al simulador el trabajo de calcular su representación.

Este es el aspecto final que aporta dicha base al cubato:



Y a pesar del alto detalle que incluye no constituye una carga para el modelo.

Por tanto, siempre deberemos tener presente que nuestro objetivo tiene que aunar tanto el mayor nivel viable de detallado como el de una representación lo más económica posible en el simulador.

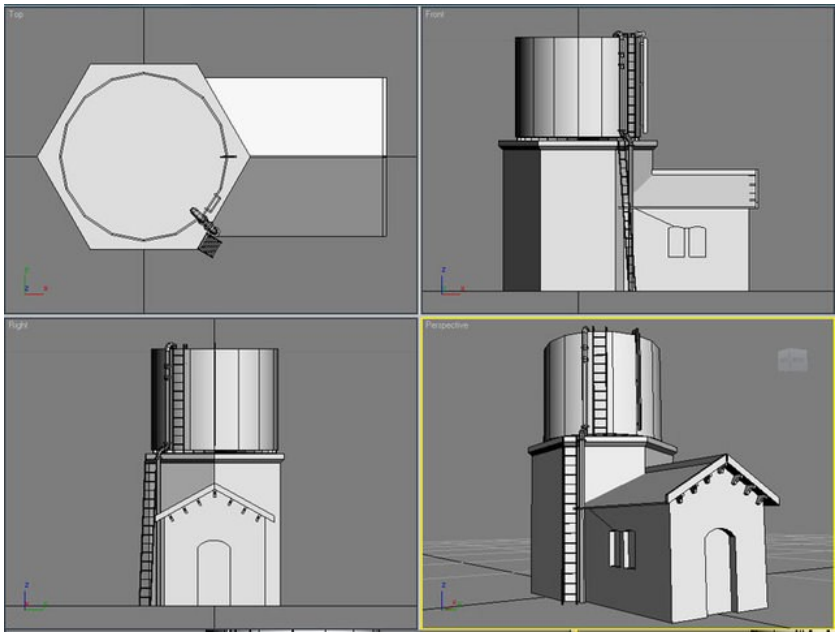
4. Mapeo de objetos (edificios) complejos.

Os voy a presentar el objeto que ha inspirado el capítulo anterior, y que me hacía gracia reproducir:

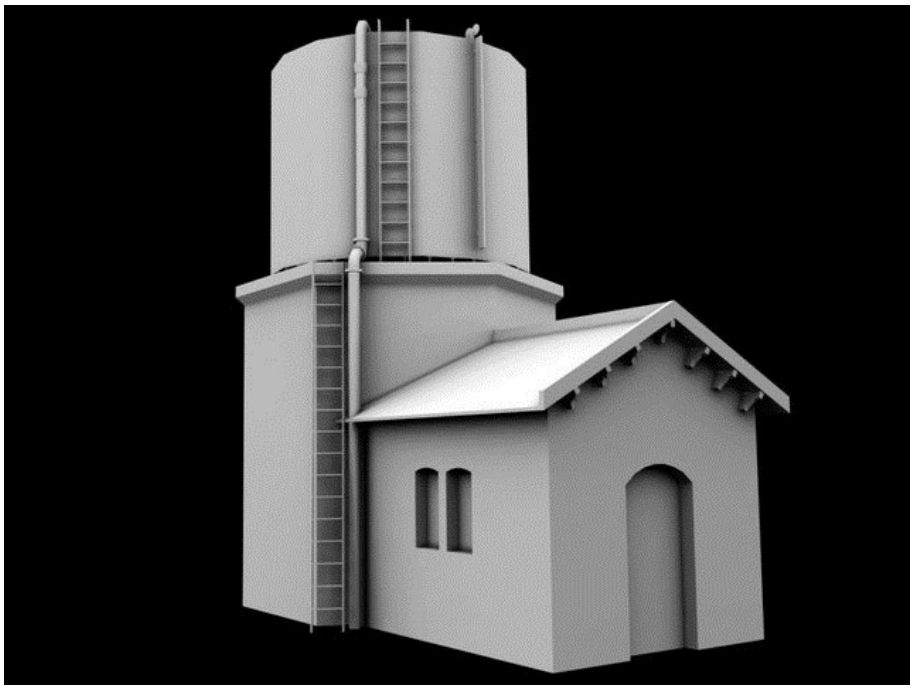
Se trata de uno de los dos cubatos presentes en la estación de Caspe, en particular el que está situado junto a la reserva de locomotoras.

Lo tenía medio empezado y ante la escasez de estos elementos Jordi Hernández me lo solicitó para la ruta "Steam of the Sierra", donde podéis verlo en una versión primitiva.

Ahora que ya me "muevo" con más facilidad por el 3ds me he puesto a realizar una versión completa. No voy a desarrollar el proceso porque no se trata de eso (para el modelado de objetos consultar los otros tutoriales), y porque en esta ocasión nos centraremos en la creación de sus texturas. Por tanto os lo presento modelado:



La vista renderizada destaca sus elementos constructivos:



Podéis observar como las sombras sobre las texturas producirán un acabado espectacular en un edificio de este tipo. Esto determina que el "Render to Texture" será imprescindible para el edificio (lo debería ser para cualquier objeto que realizásemos).

Y ello nos lleva a un pequeño, o no tan pequeño, problema. Os muestro el mapa con todos los polígonos de los elementos mapeados:

Imagen reducida en anchura de 2048 a 800 píxeles

La hoja de texturas es de 2048 x 2048 píxels, la mayor permitida por RailWorks, y, aún y teniendo en cuenta que he eliminado los polígonos que no se verán para que no ocupen innecesariamente espacio en la misma y que la separación entre mapeos la he reducido a 0,005, se observa que la resolución de los diferentes elementos es baja, principalmente en las texturas más destacadas:

El frontal de la casilla que acompaña el cubato (en rojo) ocupará una superficie de 313 x 360 píxels.

- El lateral de la casilla (en morado) 328 x 230.
- Un cuarto de desarrollo del cubato, 90º, (en azul) 232 x 328.
- Un panel de la base del cubato (en amarillo) 198 x 392.
- Y un ala del tejado de la casilla (en verde) 370 x 156.

Aun no siendo un drama, me habría gustado una resolución o texels (pixels de textura por metro real representado) sobre el doble de la obtenida, y eso representaría, en superficie, cuatro veces la mostrada. Es decir, una textura de 4096 x 4096 píxeles, o lo que es lo mismo repartir los mapeos entre cuatro texturas de 2048 x 2048 píxeles.

Pero esta solución producirá un consumo de recursos gráficos muy importante para un elemento que es modesto dentro de una ruta, puesto que no estamos hablando de la estación de Atocha.

Por otro lado, en una textura de 2048 x 1024 (la mitad de superficie de la mostrada) he podido ubicar las texturas principales del cubato en una resolución notablemente mayor, muy cercana a la deseada:

Imagen reducida en anchura de 2048 a 800 píxeles

El tamaño de las texturas más destacadas ahora es de:

- El frontal de la casilla que acompaña el cubato ocupará una superficie de 460 x 657 píxels.
- El lateral de la casilla 722 x 496.
- Un cuarto de desarrollo del cubato, 90º, 361 x 512.
- Y un panel de la base del cubato 377 x 657.

Todas ellas notablemente mayores a las anteriores.

Aquí falta la textura del tejado, que le he dejado en una textura aparte de 256 x 512 píxeles, pues en este caso voy a querer implementar bump (mapa de normales) en esta superficie

La intención es usar estas últimas texturas para mapear el edificio, teniendo presente que aplicaremos la misma textura a cada uno de los seis paneles laterales de la base, en ambas fachadas laterales de la casilla, o en el cubato repitiendo la textura cada 90º, así como en su interior. Con esto cumpliremos sobradamente con el objetivo de optimizar los recursos aumentando la resolución de las texturas a la par que disminuimos el tamaño de las mismas.

Pero seguro que alguien habrá pensado que si llevamos a cabo estos propósitos ya podemos ir olvidándonos del Render to Texture y de aplicar las sombras que nos muestra la imagen anterior a las texturas, pues la sombra que producirán las escaleras y la tubería sobre el cubato se repetiría también cada 90º, aun y no existir estos elementos, así como la regla del nivel proyectará sombra en otra sección también del cubato. Lo mismo pasará entre un lateral y el otro de la casilla, puesto que sólo uno de ellos se sombreará con la tubería y las escaleras. Y ya no digamos en los paneles laterales de la base del cubato, que los hay que reciben sombra de la casilla, de la escalera, de ambos y de ninguno.

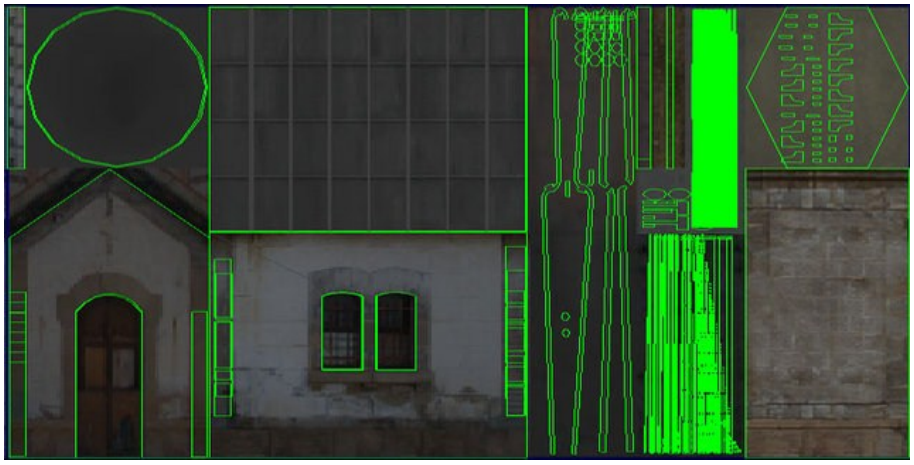
Este mismo problema nos ocurrirá en muchas ocasiones en grandes edificios, donde no nos es posible reproducir las grandes fachadas completas (so pena de perder resolución) pero por contra dicha textura se compone de módulos que se repiten a lo largo del edificio, y la aplicación de la textura repetida en módulos tiene el problema de que otros elementos del edificio pueden producir sombreados sobre alguno de estos módulos y no en otros.

Más tarde veremos cómo podemos realizar lo comentado anteriormente sin perder el sombreado del R2T pero de momento terminaremos el trabajo.

En primer lugar, mapearemos el modelo sobre la textura que hemos creado de 2048 x 1024. Algunas partes, como se ha mencionado, se superpondrán a otras que compartan la misma textura, pero no nos preocuparemos:

- Los dos laterales de la casilla usan la misma textura.
- Las seis paredes de la base comparten también textura.
- El cubato se mapeo en trozos de 180º sobre una misma zona, dos secciones exteriores (180º + 180º) y las dos interiores también.
- Para la cara superior de la base he usado una textura beige pálido, que comparten también los remates de las vigas de la casilla.
- Ambas escaleras, y sus escalones, se han mapeado en el rectángulo gris oscuro.
- Las vigas, se han mapeado en el rectángulo gris claro.
- Todas las caras laterales de las ventanas y puerta de la casilla se superponen a una parte de la fachada lateral.

Con todo ello, el mapeo nos quedará más o menos como el siguiente:



Si exportamos, una vez creados los objetos para la sombra dinámica, se puede visualizar así en el simulador:

No está mal, pero adolece de la falta de la iluminación y del sombreado del R2T (forma abreviada del Render to Texture), aún y poseer sombras dinámicas.

Veamos cómo implementarlos... en un nuevo capítulo.

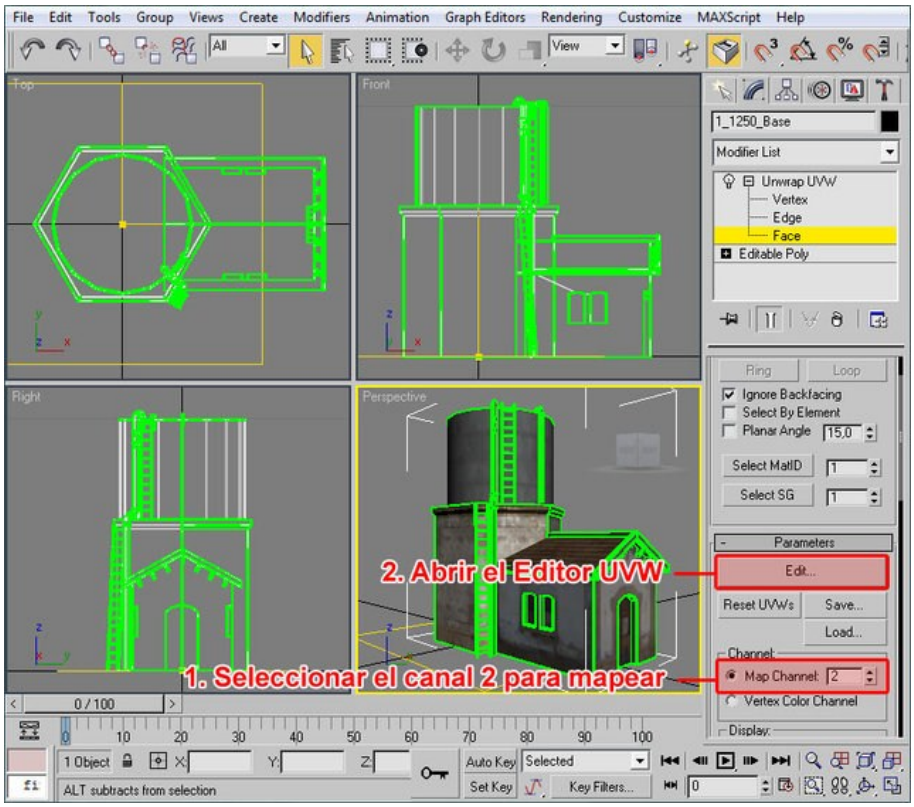
4.1. Render to Texture sobre un mapa alternativo. Shader TrainLightMapWihtDiffuse.fx

Hemos visto el modelo ya texturado, pero nos falta ver como aplicar una preiluminación a un mapeo no exclusivo de las texturas.

Para aplicar dicha iluminación y sombreado necesitamos mapear todos los polígonos bajo un esquema diferente al que ya poseen, puesto que el mapeado existente reutiliza zonas de la textura que ahora van a recibir sombreados diferentes aun y tener la misma textura diffuse.

En este caso, trabajaremos con todos los elementos "visibles" del modelo que vayan a recibir y participar de la preiluminación. Esto excluye a los elementos que constituyen la sombra dinámica y los que puedan representar LODs lejanos (los que tengan un número de LOD diferente a 1), los cuales podemos proceder a ocultar. Respecto del tejado, aunque este elemento usará una hoja de textura independiente y el shader TrainBumpSpec.fx con un mapa de normales de la textura usada, lo incluiremos en el proceso de preiluminación pues participa del mismo, produciendo sombra en la casilla y otros elementos, aunque al final él mismo no verá aplicada preiluminación (que en realidad no le hará falta al ser un elemento orientado hacia la luz, que recibirá poca o ninguna sombra).

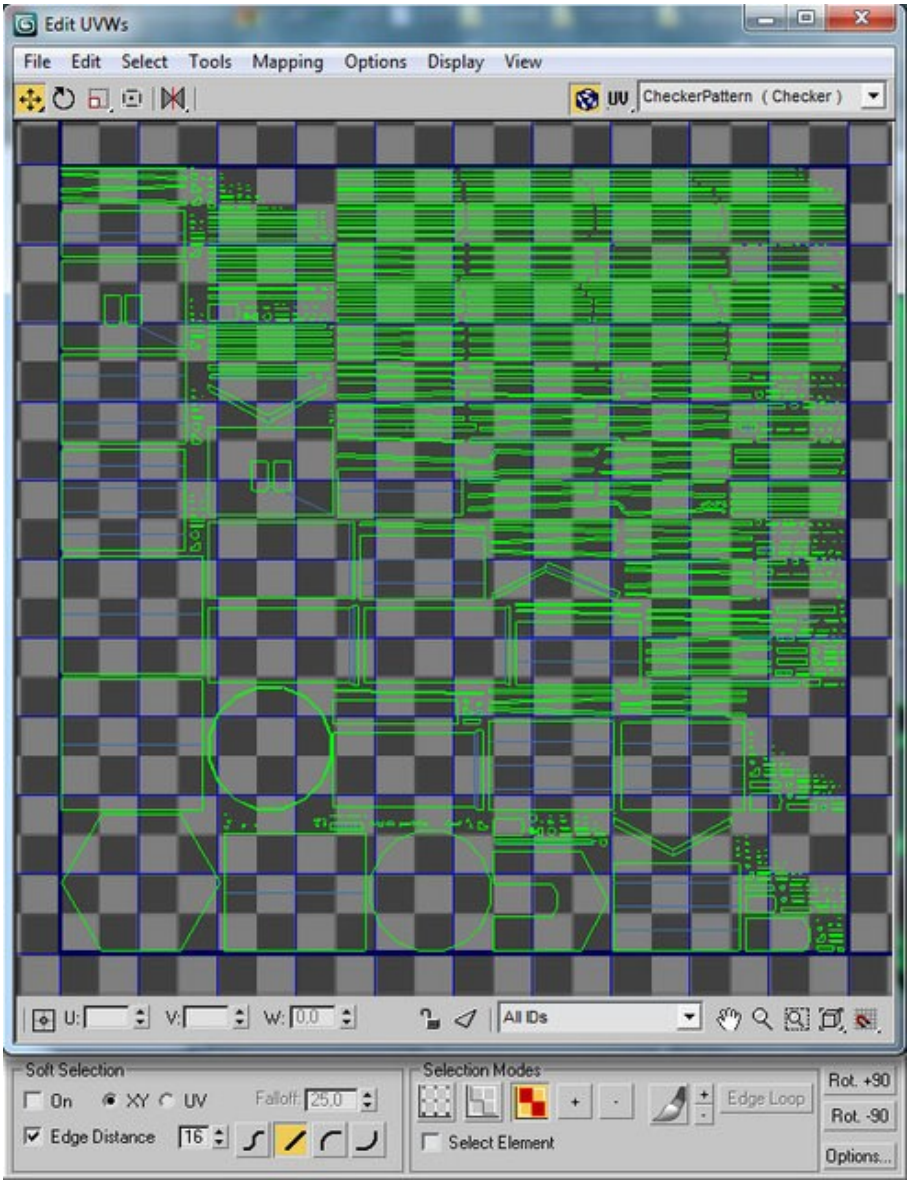
Seleccionaremos todos los elementos que han quedado visibles, y a esta selección le añadiremos un modificador Unwrap UVW y escogeremos el modo "Face":



Y dentro del grupo "Parameters":

1. Tenemos que ajustar, antes que nada, el canal de los mapeos a 2.
   Todos estos polígonos ya tienen informados unos mapeos sobre la textura Diffuse que hemos elegido, y dichos mapeos se han realizado sobre el canal 1, que es el defecto y no lo habremos modificado, pero ahora queremos introducir unos nuevos mapeos "sin solapar" para el mapa de iluminación, sin perder los anteriores. Esto lo haremos en este segundo canal.
2. Una vez seleccionado el canal podemos abrir el editor UVW.

En el Editor UVW seleccionaremos todos los polígonos (Control + A) y abriremos el menú "Mapping" y "Flatten Mapping...". En la ventana que nos aparece dejaremos el ángulo en 45º, pero el espaciado entre mapeos lo reduciremos a 0,002, para de esta forma aprovechar mejor la hoja de textura para la iluminación. Pulsaremos OK y obtendremos un nuevo mapeo de los polígonos del cubato.



Confirmaremos estos nuevos mapeos saliendo del editor UVW y colapsando el stack de modificadores.

Estos mapeos no coinciden en absoluto con los que tenemos para la textura diffuse, pero no debe preocuparnos; lo importante es que estén realizados sobre el canal 2, y eso lo podemos comprobar visualmente porque el modelo debe seguir mostrando el texturado de sus caras tal y como estaba antes de nuestra operación.

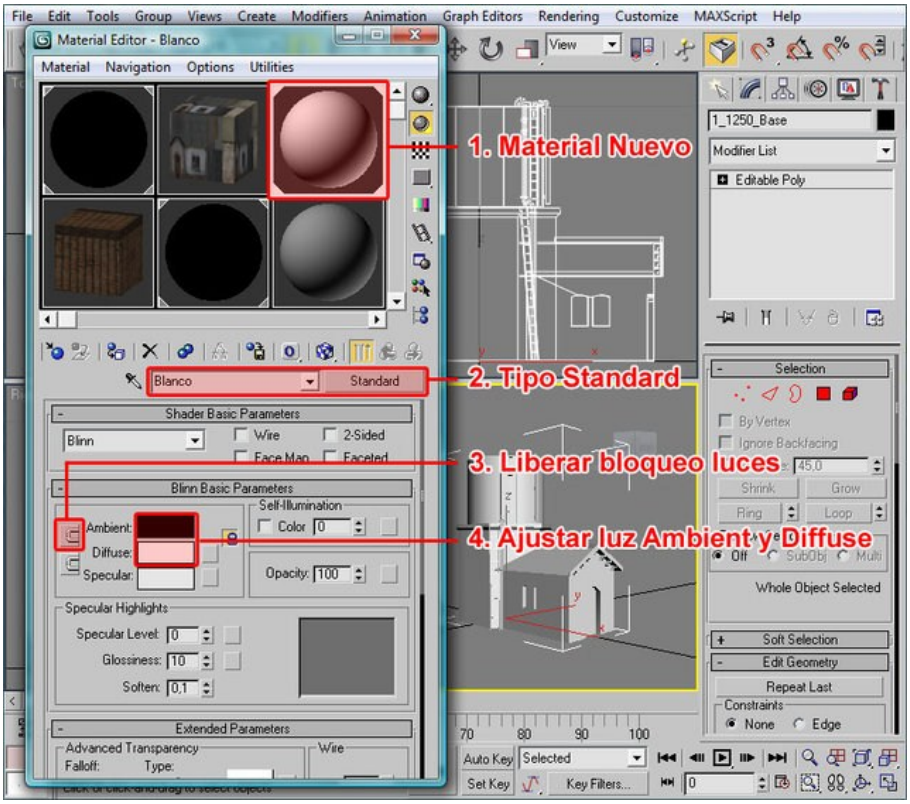
Procederemos a salvar la escena con los cambios mostrados.

Ahora debemos aplicar la preiluminación al conjunto, para lo cual guardaremos el trabajo con otro nombre de archivo; yo suelo emplear el sufijo "_Light" para diferenciar ambos archivos. Sobre esta nueva escena con el mismo cubato, realizaremos algunos cambios al modelo, tan sólo a efectos de la preiluminación (a efectos del modelo el archivo anterior es el correcto si en el futuro queremos seguir trabajándolo). Estos cambios ya se vieron en el tutorial de la casilla, pero los repito porque haremos algún pequeño cambio.

Uniremos en uno sólo (Attach) todos los objetos que vayan a recibir y participar de la preiluminación. Esto excluye a los elementos que constituyen la sombra dinámica y LODs diferen-

tes a 1, los cuales podemos proceder a borrar. Ya hemos comentado que el tejado lo dejaremos para que proyecte sus sombras sobre otros objetos.

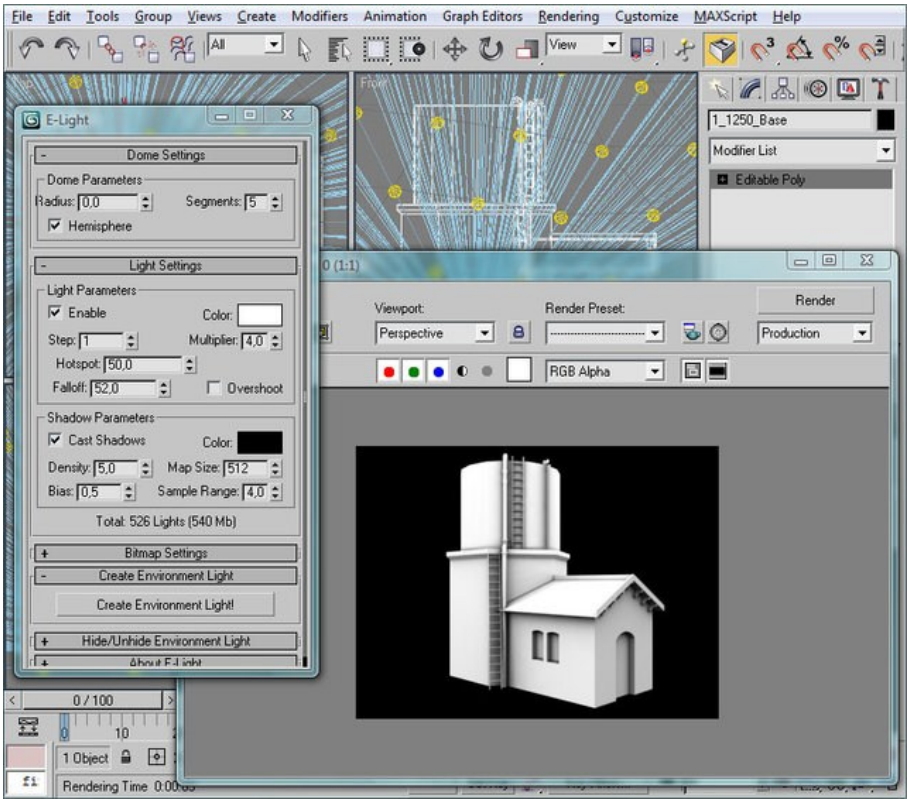
Ahora podemos prepararnos para llevar a cabo el Render to Texture. Seleccionamos el modelo y abrimos el editor de Materiales para crear uno nuevo:



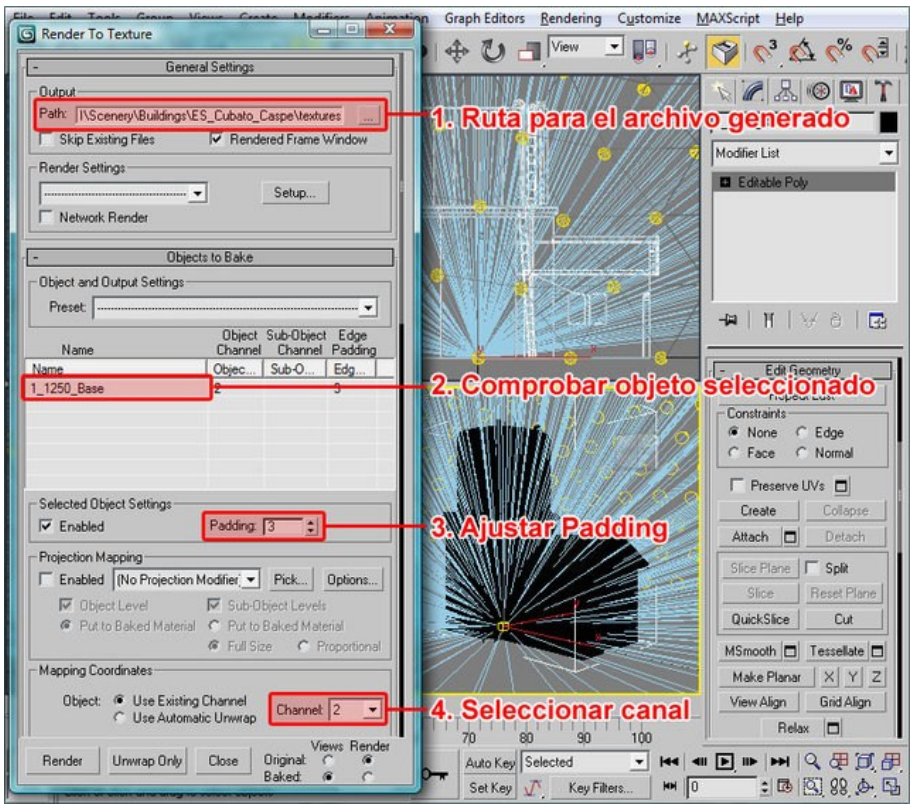
1. Elegiremos un slot de material no usado.
2. Opcionalmente le daremos nombre, y comprobaremos que el tipo de material es "Standard". Lo debe ser si el material no había sido usado, pues es el defecto.
3. Pulsaremos en el botón que bloquea las luces Ambiente y Diffuse para deseleccionarlo y liberarlas (si no ambos tipos de luces se igualan al cambiar cualquiera de ellas).
4. Ajustaremos la luz Ambient para el color negro, y la luz Diffuse para el color blanco

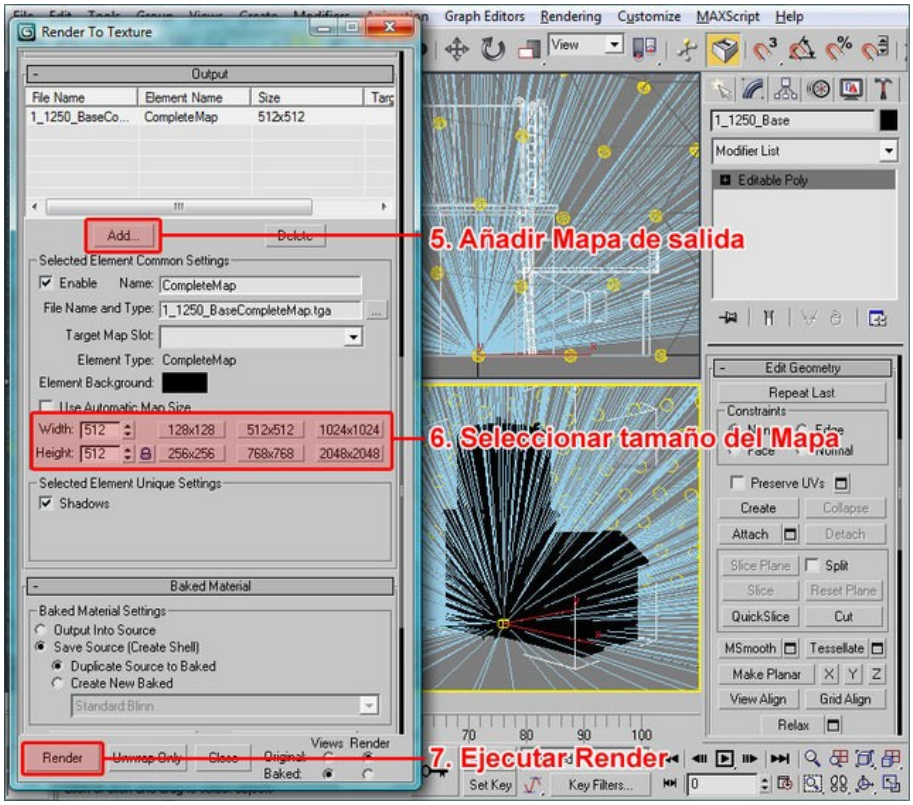
Una vez el nuevo material esté definido lo asignaremos al modelo, el cual lucirá de un blanco inmaculado.

Ahora podemos crear una cúpula de iluminación con el script e-Light (cómo vimos en el tutorial de la casilla), ajustar los parámetros de iluminación y comprobar mediante el Render que las sombras se generan en la medida que nos satisfaga:



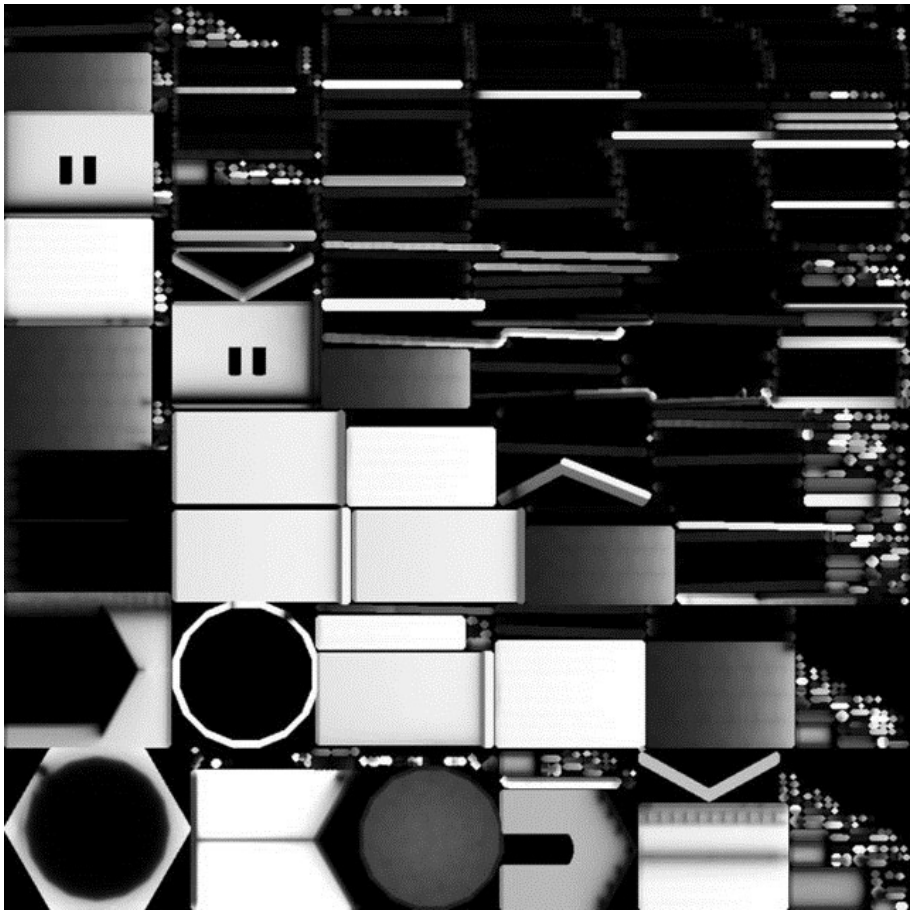
Una vez llegados al ajuste deseado, abriremos la ventana de "Render to Texture", con la tecla 0 (cero) por ejemplo:



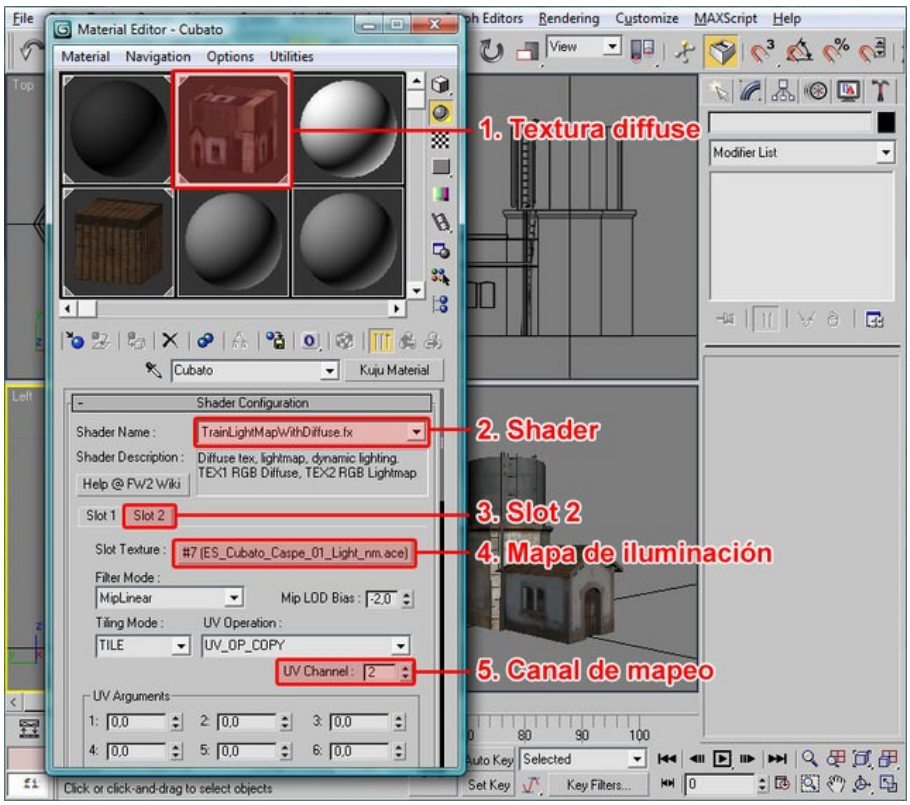


1. Indicaremos el directorio donde queramos que nos genere el archivo del mapa de preiluminación. Yo suelo indicar directamente el directorio de texturas del modelo.
2. Comprobaremos que el objeto del modelo está seleccionado. En caso contrario bastará con seleccionarlo en alguno de los viewports.
3. Modificaremos el parámetro Padding con el valor 3. El defecto, 2, puede generar una franja negra en los bordes de las texturas.
4. IMPORTANTE: seleccionaremos el canal 2 para producir el mapa de iluminación (el que no tenía solapamientos entre los mapeos).
5. Pulsaremos el botón "Add" para añadir un mapa a generar. Nos aparecerá una ventana donde podemos elegir entre varios tipos, pero en nuestro caso nos interesa el llamado "Complete Map".
6. Especificaremos el tamaño que queremos que tenga el mapa de iluminación generado. Este tamaño no tiene por que ser el mismo que el del mapa Diffuse del modelo. Un tamaño medio será 512x512, aunque para edificios más peuqeños podemos usar 256x256 y en grandes edificios incrementar el tamaño (el valor 768x768 no es aplicable en RailWorks).
7. Finalmente podemos pulsar el botón Render para obtener el resultado.

Tras pulsar "Continue" en la ventana de aviso que aparece, obtendremos el mapa de iluminación de nuestro modelo:



El archivo obtenido lo deberemos convertir a .ace, y terminada la operación volveremos al 3ds Max para abrir la escena original del modelo (no la que nos ha servido para generar el mapa de iluminación) y actualizaremos el material:



1. Seleccionaremos el material de la textura diffuse de nuestro modelo

2. Escogeremos el shader "TrainLightMapWihtDiffuse.fx", que permite añadir a la textura diffuse del modelo un mapa de iluminación.
3. Activaremos el Slot 2 de este shader. El Slot 1 deberá contener la textura diffuse.
4. Informaremos el nombre del archivo donde hemos guardado el mapa de iluminación, después de convertirlo en .ace.
5. IMPORTANTE: deberemos seleccionar que este mapa se aplicará a los polígonos mediante el canal 2 de mapeos e éstos. Obviamente, la textura diffuse informada en el slot 1 deberá tener el canal 1 como origen de los mapeos.

Una vez actualizado el material, podemos volver a exportar el cubato y proceder a visualizarlo en el simulador:

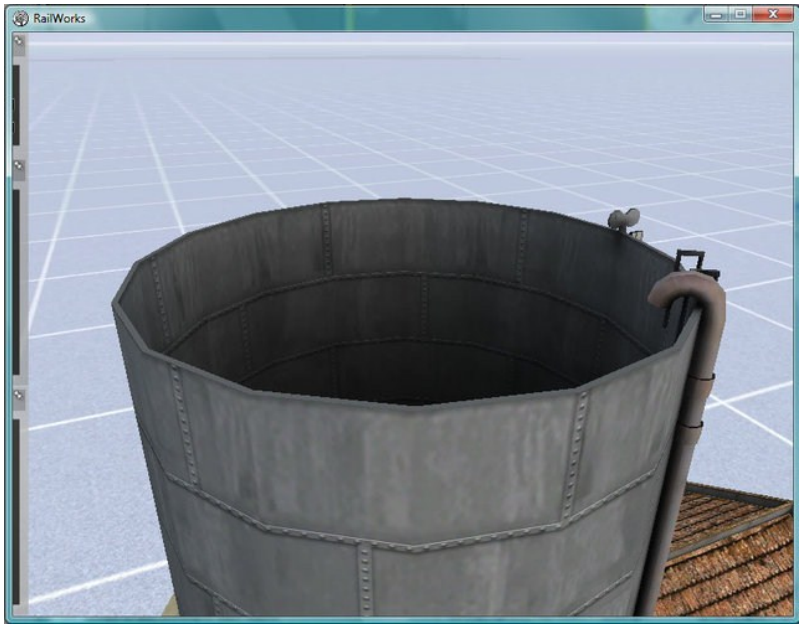
Podemos comprobar que la textura diffuse se conserva igual que antes (os muestro la imagen obtenida al final del capítulo anterior):

... pero a ésta se han añadido las sombras del mapa de iluminación donde procede, con independencia del mapeo usado en el mapa diffuse y dando un acabado netamente superior.

Hemos obtenido lo que queríamos, y esto nos será muy útil sobre todo en grandes edificios. Pero no todo es alegría, este procedimiento, aunque muy efectivo, es incompatible con los mapas de normales, de brillos o de entorno, en aquellas caras en que se usa este shader, razón por la cual su uso se generaliza en edificios antes que en material móvil. Por contra, en esos grandes edificios es una muy buena solución para mantener una resolución alta de las texturas a pesar de las grandes superficies a cubrir, soportando el mosaico de texturas, y no perder los beneficiosos efectos de la preiluminación.

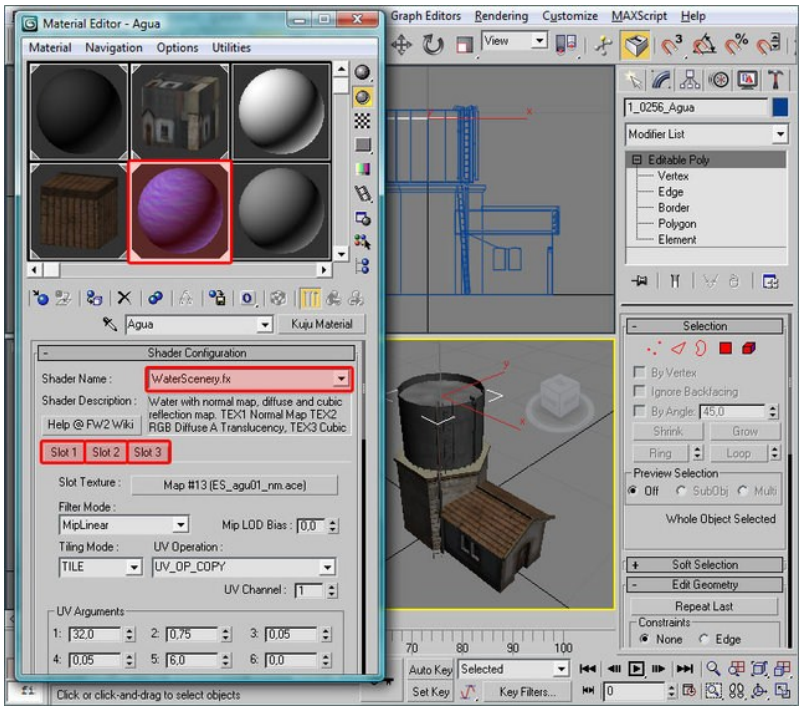
5. Modelado de una superficie de agua. Shader WaterScenery.fx

Podemos asomarnos al interior de la cuba del cubato que hemos mostrado:



¡Vacío! Terriblemente vacío. Y claro, de esta forma no podremos suministrar el preciado líquido a las locomotoras que lo necesiten. Sí, realmente será necesario llenar de agua este depósito, o de lo contrario no será en absoluto creíble

Para representar este agua contenida en la cuba dispondremos de un polígono, ajustado al contorno de la cuba, que definirá la superficie de esta agua, marcando el nivel hasta donde se encuentra el líquido. Y bastará con asignar a esta superficie un material adecuado. En este caso nos apoyaremos en el shader "WaterScenery.fx"



Este shader dispone de tres slots para texturas, y todos ellos deben estar asignados. Veamos estos slots:

- El Slot 1 debe tener informado como textura un mapa de normales que represente la irregularidad de la superficie del agua que queremos reproducir. Además en este slot se debe incluir unos parámetros adicionales para controlar la superficie del agua a nuestro antojo: su movimiento, reflejo, etc .
- El Slot 2 debe tener informado como textura un mapa diffuse del aspecto que tendrá la superficie del agua en sus canales RGB. Además, en el canal Alpha de dicho mapa diffuse se incluirá el grado de transparencia que debe presentar la superficie del agua.
- El Slot 3 se usa para informar al motor gráfico del mapa cúbico de reflexión que se aplicará a la superficie del agua. Este CubeMap es el que provoca que el cielo se refleje sobre la superficie del agua.

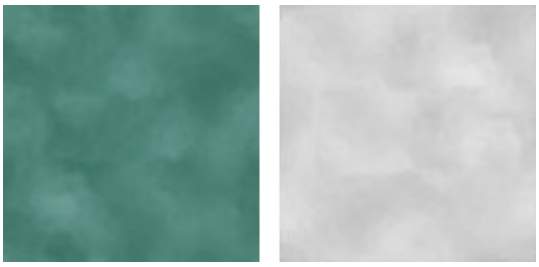
Los dos primeros mapas, el de normales y el diffuse, ya estamos acostumbrados a usarlos y no entrañan ninguna dificultad, pero de entrada el CubeMap se nos antoja como un tanto "peliagudo". No hay que preocuparse en absoluto, puesto que este mapa lo calcula el Simulador durante la sesión y lo proporciona de forma automática. Nosotros en el shader tan sólo debemos informar una textura cualquiera para que el slot esté ocupado, pero ésta puede ser cualquiera, en general una textura dummy al igual que la empleada para el shader de las sombras dinámicas. En algunos foros el personal de Kuju comentan que para estos efectos suelen repetir la textura diffuse que tenga ya asignado el shader, y de esta manera ahorran recursos. Vemos en detalle las texturas mencionadas.

Para el mapa de normales necesario en el slot 1 vamos a usar en esta ocasión éste:

ES_agu01_nm.ace

Se trata de un mapa de normales de lo más "normal" (que chiste más malo me ha salido).

Para el mapa diffuse necesario para el slot 2 usaremos este otro, con un marcado tono verdoso del agua, y que muestro con su canal Alpha:

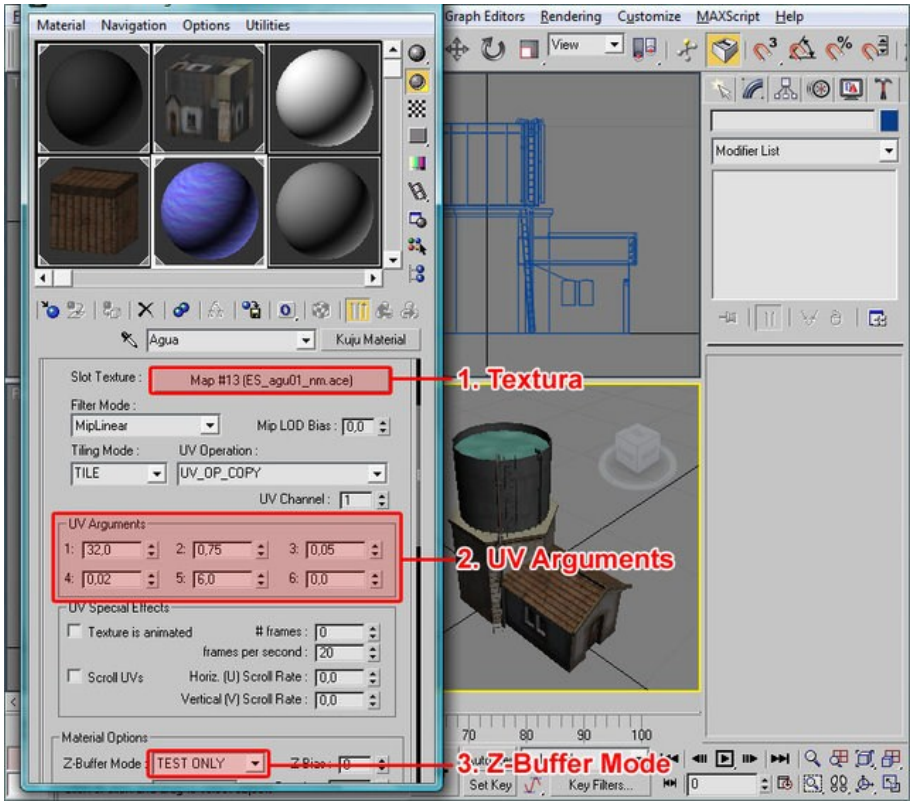


ES_agu02b_nm.ace

Le he dado un tono verdoso al agua para ver las posibilidades que tiene este mapa. Respecto al canal Alpha hay que decir que cuanto más oscuro sea el canal más transparente se mostrará el agua (negro dará una transparencia total) y por contra cuanto más claro más opaca será la superficie del agua, y más se visualizará la textura diffuse asociada.

Veamos ahora como definir este shader:

En el Slot 1 deberemos:



1. Asignar la textura que contiene el mapa de normales.
2. El control de la superficie del agua lo llevaremos a cabo mediante cinco de los seis argumentos UV de que disponemos. En particular:
    - **Argument 1:** Componente especular (valores entre 0 y 64), indica el grado de reflejo de la superficie del agua. Un valor medio de 32 es suficiente, aunque podemos variarlo a nuestro gusto para incrementar el reflejo o eliminarlo (en unas aguas encharcadas, con barro o vegetación).
    - **Argument 2:** Factor de oscilación (valores entre 0 y 1). Expresa el grado de movimiento u oscilación que se introducirá en la superficie del agua. Este shader modificará la superficie donde se aplica alterando la posición en altura de los vértices en ella, provocando una oscilación (ondulación) real de dicha superficie.
    - **Argument 3:** Velocidad del movimiento (0.02 es un valor correcto para aguas tranquilas).
    - **Argument 4:** Altura de la oscilación (0.02 es un valor correcto para aguas tranquilas).
    - **Argument 5:** Escala de la oscilación (6 es un valor suficiente para aguas tranquilas).
3. No olvidar poner el Z-Buffer Mode en TEST ONLY.

En el slot 2 deberemos:

1. Asignar la textura que contiene el mapa diffuse con el canal alpha indicando la transparencia del agua.

No es necesario cumplimentar nada más en este slot.

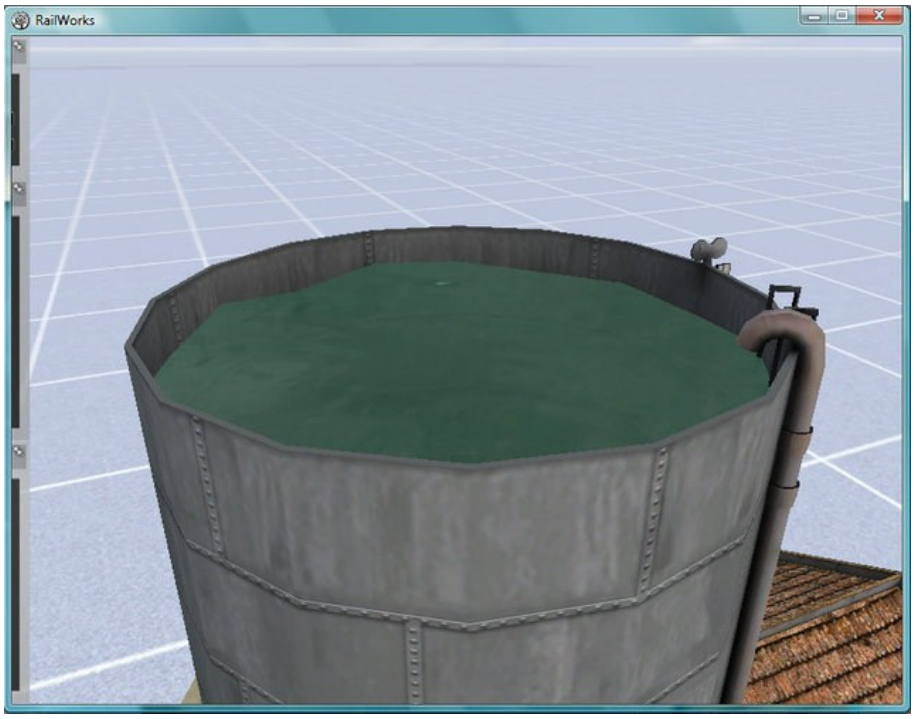
Y en el slot 3 deberemos:

1. Asignar alguna textura, no importa su contenido, para que el simulador utilice la memoria de ésta para calcular el mapa cúbico de reflexión. En mi caso me he limitado a repetir la textura diffuse usada en el slot 2.

Tampoco es necesario cumplimentar nada más en este slot.

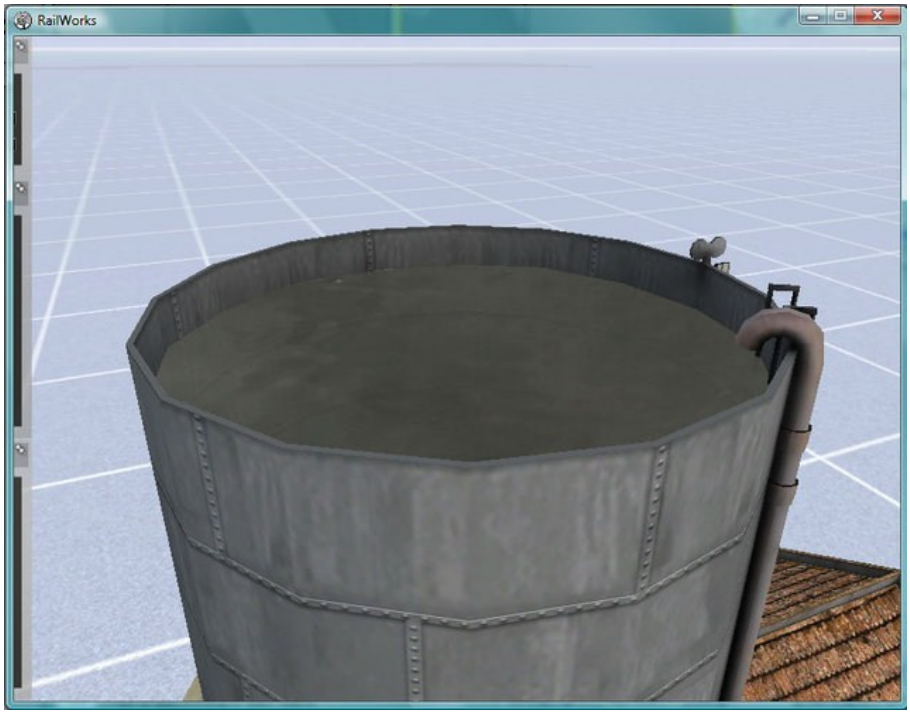
Con esto tenemos listo el material y podemos asignarlo al polígono que hará el papel de superficie de agua.

Atención:
Este es un vertex shader (es decir, las posiciones de los vértices de la superficie son físicamente alteradas) y es por tanto costoso de ejecutar, por ello se recomienda mantener su uso en un mínimo razonable. Observemos que la superficie del agua que se muestra no es plana, como ocurre con otros shaders de agua.



Ya tenemos nuestra superficie de agua con una animación de la misma producida por el viento.

El aspecto en general podemos variarlo como gustemos, particularmente, prefiero que el agua no tenga este aspecto verdoso (aunque en otros casos puede ser procedente), y por tanto he eliminado ese tono de la textura diffuse obteniendo este otro aspecto:



Nuestra cuba está ahora llena de agua muy clara, como corresponde. Nuestras locomotoras se merecen la mejor calidad de agua que les podamos dar.

(Véase vídeo: http://www.youtube.com/v/VLFMW5Ry-xY)

Para superficies mayores de agua, podemos usar el shader WaterCubeMap.fx, mucho más económico al no ser un "vertex shader", es decir no dinamiza los vértices del polígono, pero mantiene la reflexión en movimiento de la superficie del agua.

6. Trabajando sobre plano

Una vez finalizado el modelo del cubato, se tercia proveer al simulador de una grúa hidráulica ibérica. En esta ocasión el modelo escogido es la grúa de agua unificada de la compañía del Norte, que fue habitual en muchos parajes de nuestra geografía. Os muestro un par de ellas:

Una búsqueda de más material es imprescindible para documentar todo modelo que acometamos. En esta ocasión he encontrado fotografías, pero no planos o esquemas de la misma. Si bien en edificios el error en las medidas o proporciones puede ser tolerable hasta cierto punto, en otro tipo de elementos, y principalmente en material móvil, este error no debe existir, por lo que trabajar sobre plano o esquemas se hace muy necesario. ¿Y que hacer si no tenemos dicho material? Proceder a dibujarlo nosotros mismos.

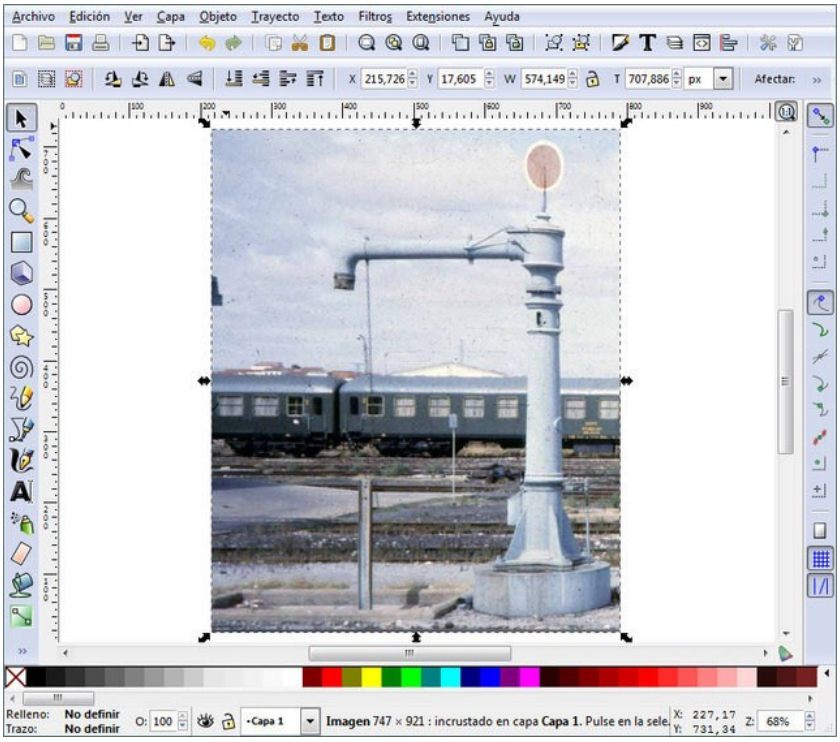
Por suerte hay herramientas que nos ayudarán en gran medida a este propósito. Vamos a ver en este capítulo como crear planos o esquemas de los modelos que vayamos a modelar, y como usarlos en 3ds para que nos ayuden en nuestra tarea de creación del modelo.

6.1 Construir los planos

Como herramienta para dibujo podemos usar alguna de las existentes, y en este sentido Corel Draw ha marcado tendencia en artes gráficas. No obstante, existe una aplicación open source, sencilla de usar y a la vez muy potente, que nos servirá perfectamente a nuestro objetivo: InkScape, y que podemos descargar desde su web www.inkscape.org.

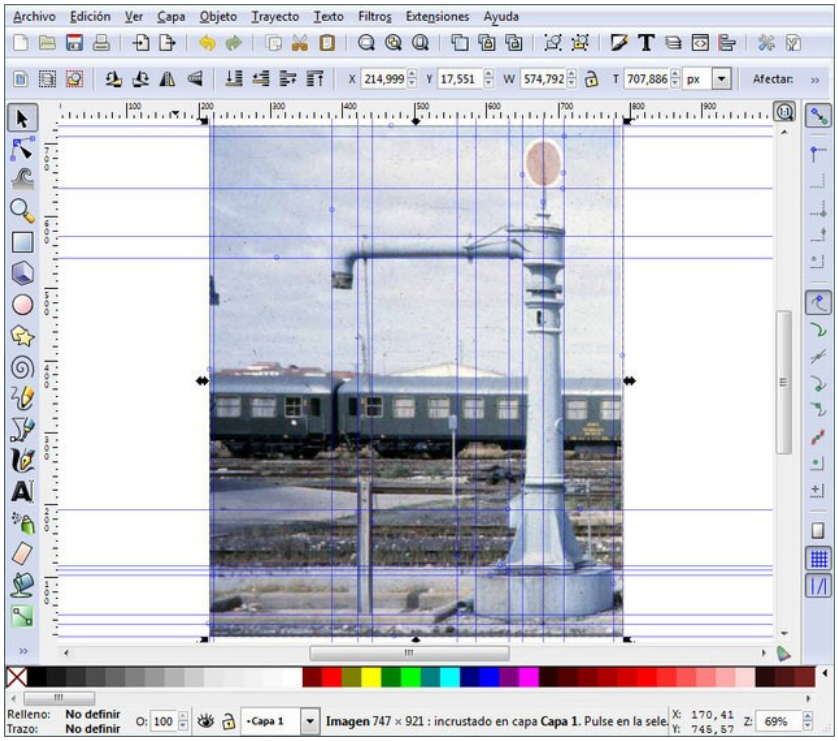
El programa nos permitirá ubicar en la zona de trabajo una fotografía del modelo y dibujar sobre ella (reseguirla con trazos) para obtener un esquema o incluso un plano. Os muestro de forma resumida los pasos básicos:

Con una interfase muy parecida a la usada por Corel, podemos crear un objeto con la fotografía del modelo:



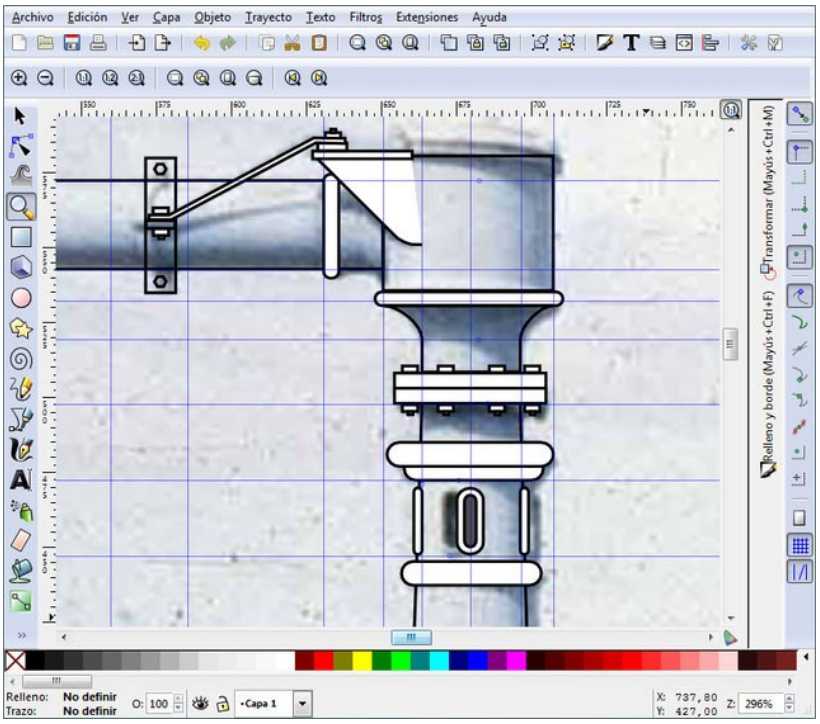
Por supuesto, la imagen debe estar correctamente nivelada y corregidas sus líneas de fuga y deformaciones, cómo ya se indicó para el caso de las texturas en su momento.

Sobre ésta imagen Inkscape nos permite ir arrastrando diferentes "líneas guía", tanto verticales como horizontales, que nos definan los puntos básicos de la figura a dibujar:

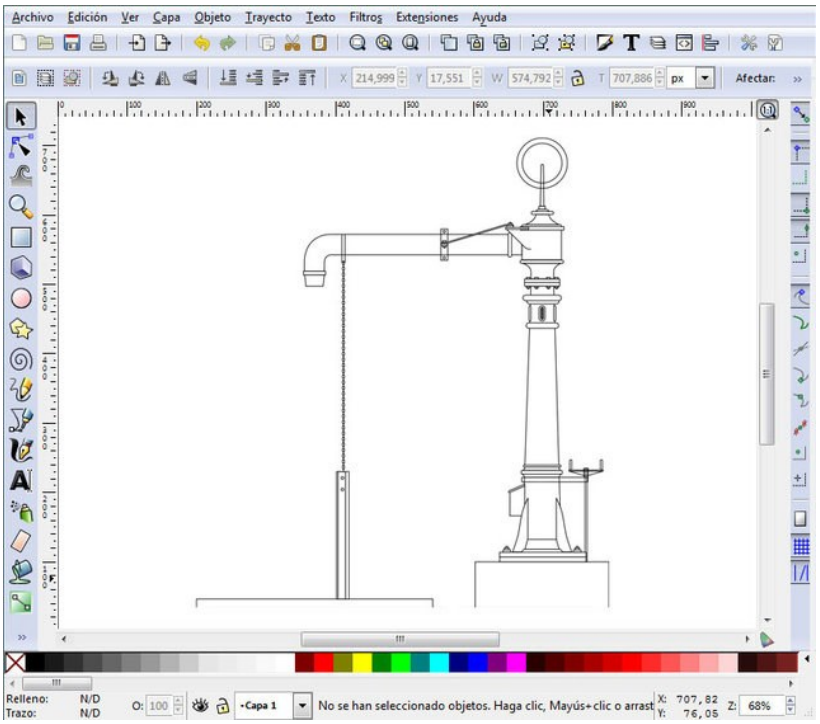


Las líneas guía, en azul en la imagen anterior, son marcadores que obtenemos pinchando y arrastrando en las reglas lateral o superior. Una vez ubicadas en el área de trabajo, y a modo de imanes, los elementos que dibujemos se adaptarán a ellas si están lo suficientemente cerca.

Una vez establecidas todas las que creamos convenientes podemos proceder a ir trazando rectángulos, círculos y líneas de dibujo que bordeen los contornos de la pieza. En algún caso podemos rellenar la figura creada en color blanco para ocultar líneas que estén situadas tras ésta. Esta operación no es complicada, pero hay que tener paciencia con ella. Este es un detalle durante el proceso de dibujado:



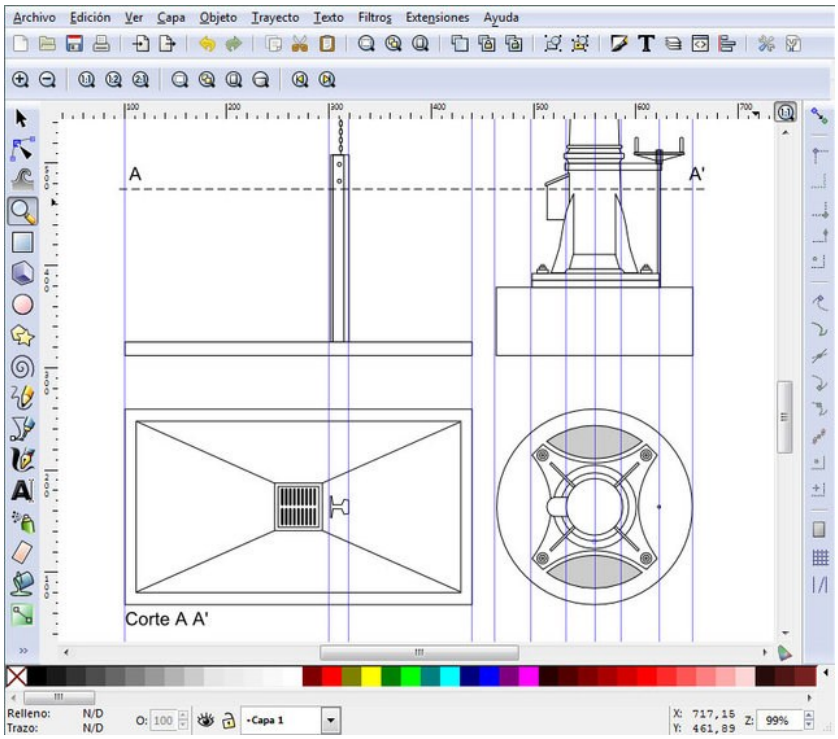
Cuando hayamos terminado de reseguir el modelo (en este caso representó apenas algo más de una hora de trabajo) podemos eliminar la fotografía de fondo y ocultar las líneas guía para ver el resultado:



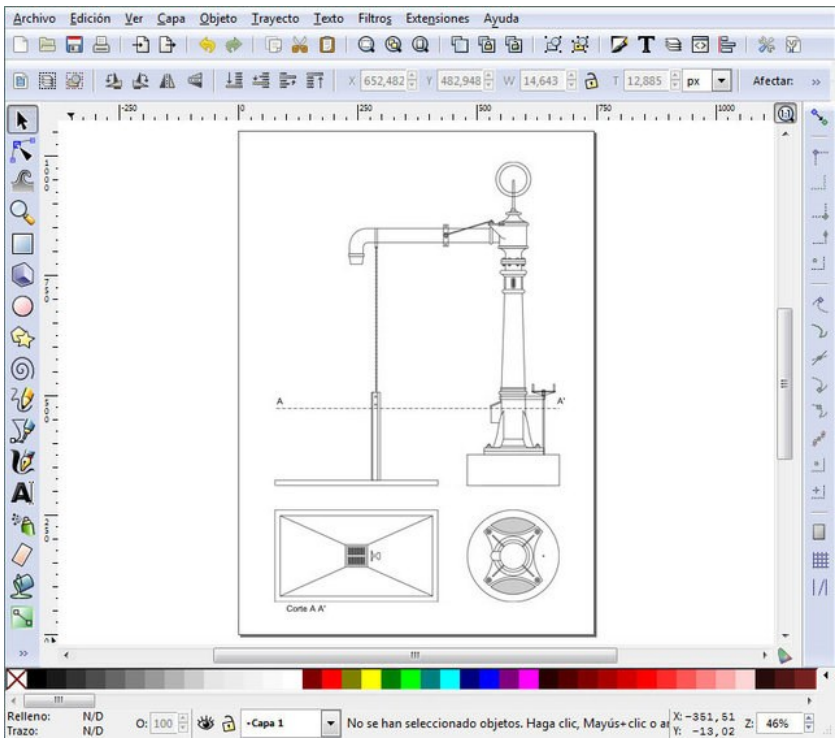
Si nos es preciso, podemos complementar el plano con una vista lateral o superior. En este caso la vista lateral no es muy necesaria al ser la columna de la grúa simétrica, pero por contra

si me interesaba la vista superior, con detalle de los elementos inferiores de la base y el pié de la grúa hidráulica.

Como, evidentemente, no disponía de fotografías cenitales del elemento, opté por realizar esta vista por proyección del lateral. Se ubican nuevas líneas guía en los puntos clave del lateral y se dibujan los elementos en la vista superior. En este caso, la base circular es fácil de crear, tomando cómo referencias el centro de la columna y sus dimensiones extremas. Esta base circular nos da a su vez la guía para el ancho de la cubeta (que en las fotografías se apreciaba era igual a dicha base). A partir de aquí basta seguir dibujando elementos:

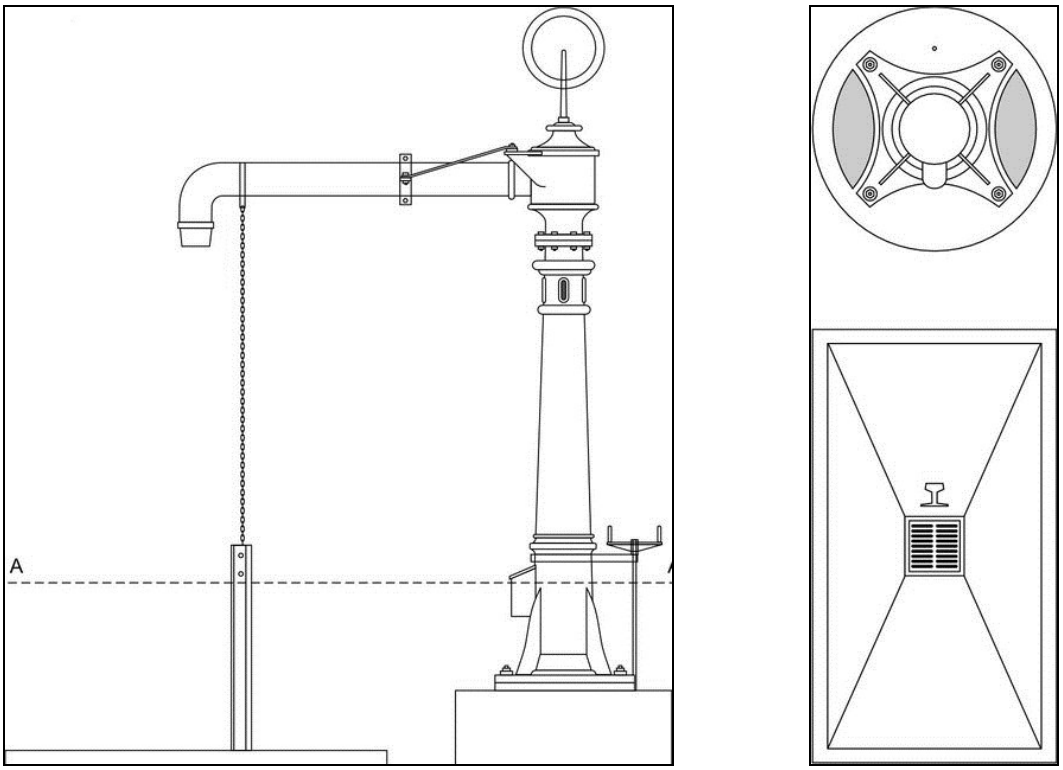


Y una vez terminado el trabajo observar el resultado final:



Tenemos un archivo con el dibujo vectorial de nuestra grúa que guardaremos y pondremos a buen recaudo en el formato que Inkscape nos sugiere ".svg". Éste es un formato para Gráficos Vectoriales Escalables (Scalable Vector Graphics) promovido por el organismo W3C y que actualmente se ha introducido como un estándar de facto para este tipo de archivos.

No obstante, este mismo plano lo exportaremos en un formato de imagen (jpg o png por ejemplo) para el 3ds Max. Del archivo resultante recortamos una imagen al tamaño, ajustado, de la vista lateral. En mi caso me ha quedado de un tamaño de 990 x 1126 píxeles:



Respecto a la vista superior, también la he recortado y girado 90º (para adecuarla a la vista top del 3ds), resultando de un tamaño de 324 x 990 píxeles:

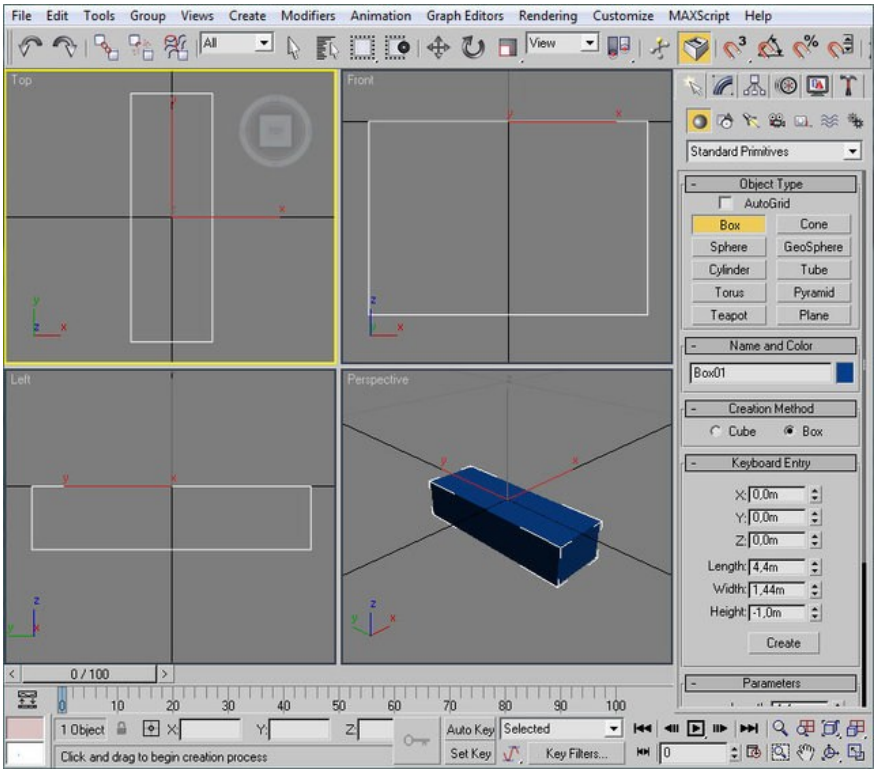
Estas dos imágenes las queremos para en 3ds usarlas como guías de modelado, por tanto debemos conocer sus dimensiones "reales" en metros.

El tamaño del material móvil no será un problema normalmente, puesto que éste es conocido (distancia entre topes, del bastidor, etc). En el caso de la grúa hidráulica se da por buena la altura de 5 metros, desde la base del terreno hasta el extremo superior del disco. Dividiendo estos 5 metros por los 1126 píxeles de altura de la imagen nos da una proporción de 0,00444 metros por píxel, que a su vez podemos multiplicar por cualquier dimensión en píxeles para obtener la respectiva medida real en metros. Así la vista lateral serán 5 metros de altura por 4,40 de longitud, y la base serán 4,40 de largo por 1,44 de ancho.

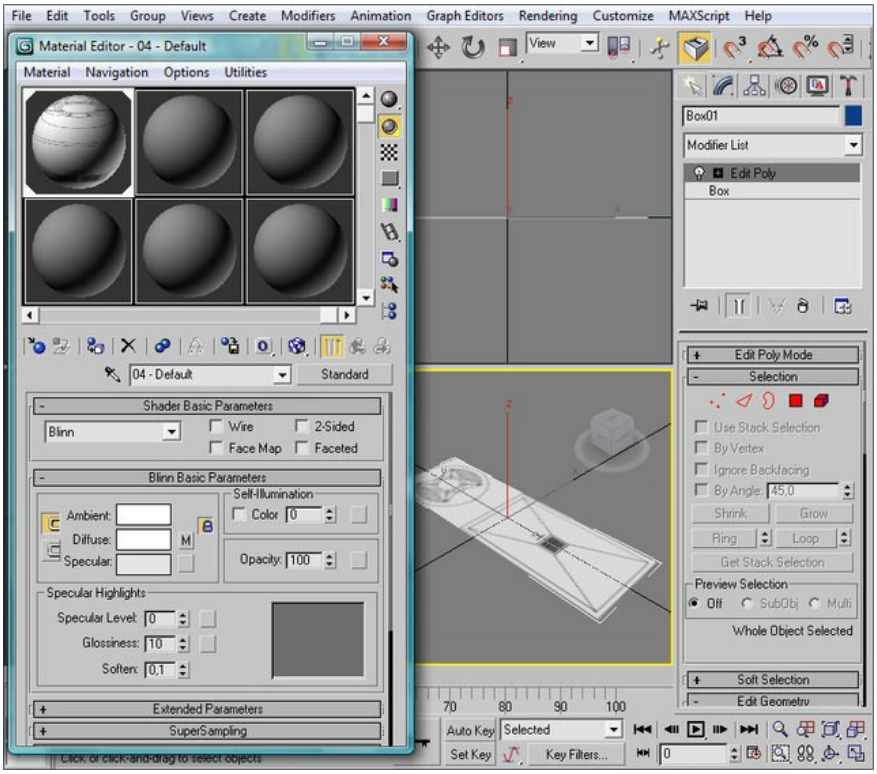
Estas dimensiones nos ayudarán en la creación de los planos en el editor del 3ds, cómo veremos a continuación.

6.2 Implementando planos en 3ds Max

Con estas medidas ya podemos ir al 3ds y crear en la vista Top un primer cubo para la base al que daremos las dimensiones: longitud 4,40 m, anchura 1,44 metros y altura -1,0 metros. Al hacer la altura negativa el cubo creado tendrá un metro de alto, pero será la cara superior la que quede alineada con el suelo (coordenada Z=0,0):

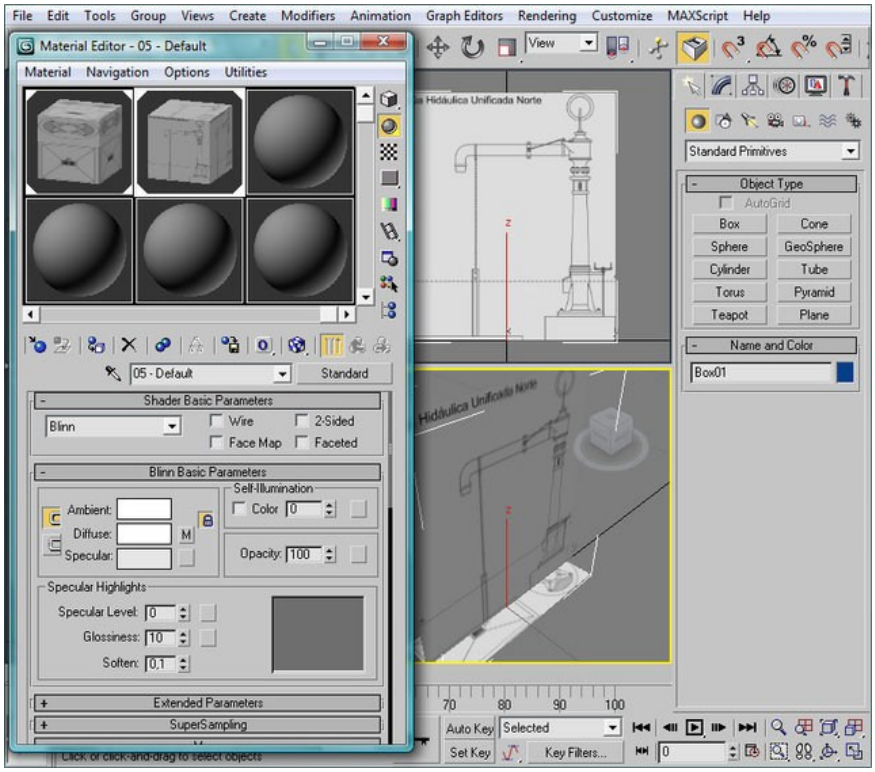


A este cubo le eliminaremos los vértices inferiores para quedarnos únicamente con dicha cara superior, a la cual asignaremos un material que tendrá asociada como textura la imagen de la vista superior antes obtenida:

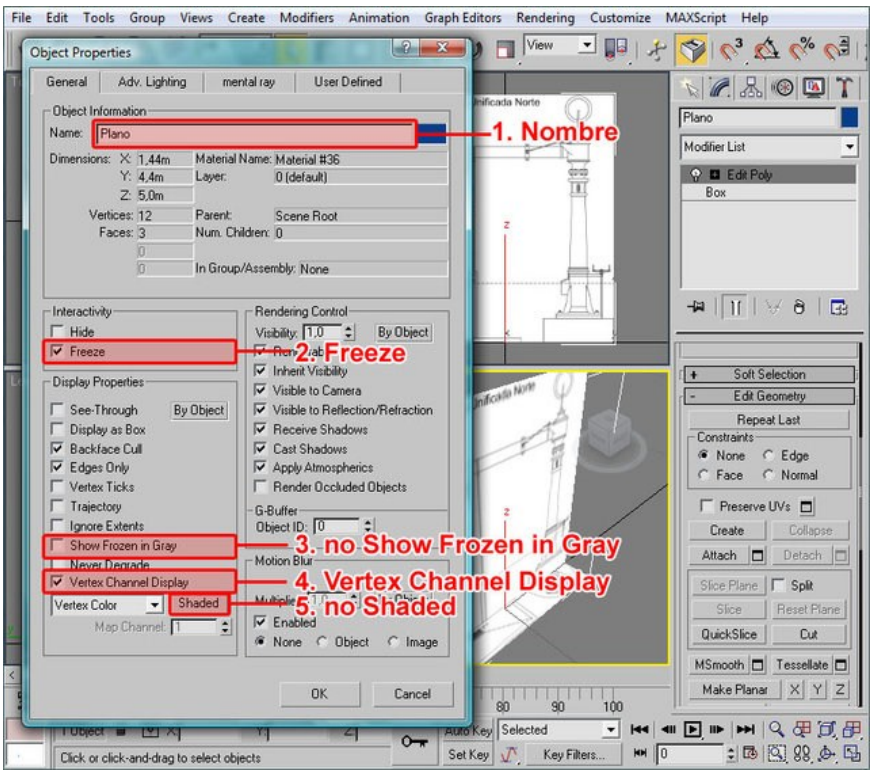


De esta forma hemos creado un plano patrón para la vista superior del objeto. Para el plano lateral procederemos de igual manera creando un nuevo cubo de 4,40 m de longitud, 2,0 m de anchura y 5,0 metros de altura. La anchura nos será intrascendente, pues una vez suprimidos los vértices posteriores del cubo desplazaremos éste en el eje x un metro para que quede alineado con la coordenada X=0.

Crearemos un nuevo material con la imagen lateral asociada y lo asignaremos al nuevo elemento:



Este plano lateral lo tendremos que repetir (clonar) para tener la vista por el otro lado, y tras su clonado procederemos a seleccionar el polígono y "fliparlo" (invertir su normal) para que sea visible por el lado contrario. Los tres elementos que tenemos hasta ahora (vista superior y dos vistas laterales opuestas entre sí) los "attacharemos" en un único elemento, del cual accederemos a sus propiedades para introducir alguna ligera modificación. Las propiedades del objeto las tenemos accesibles desde el menú contextual, con el objeto seleccionado:

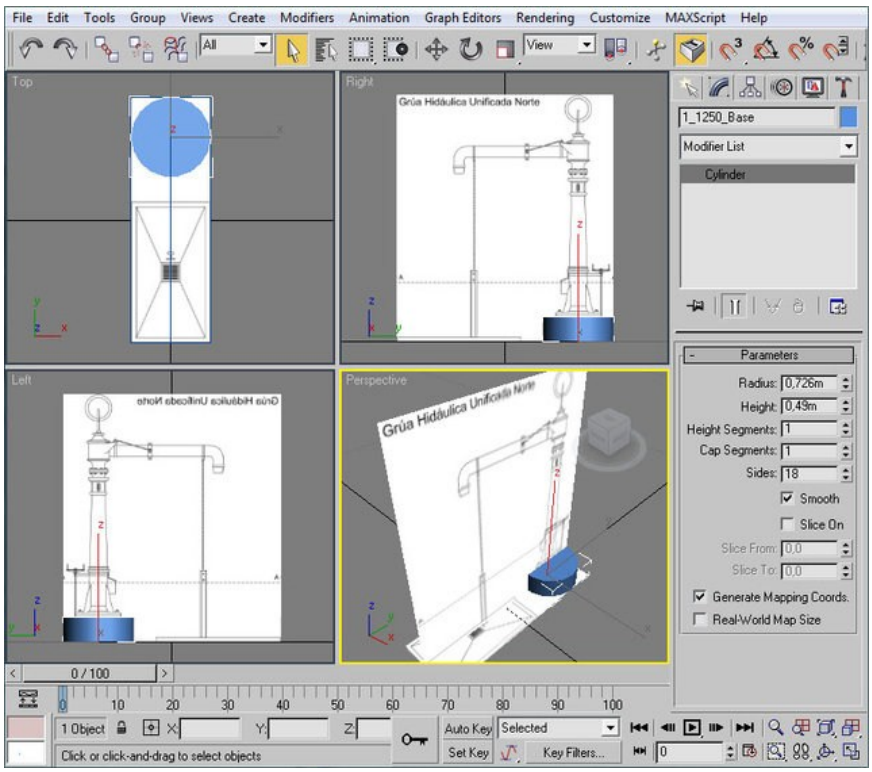


1. Podemos dar un nombre al objeto que nos lo identifique claramente en la escena.
2. Activaremos la casilla **Freeze** (Congelado) del apartado Interactivity. Un elemento "congelado" de la escena sigue apareciendo en la misma, pero no puede ser seleccionado ni interactúa con las herramientas de modelado del 3ds. Está... pero como si no estuviera y eso es precisamente lo que queremos con nuestro "plano" de la aguada: verlo pero que no moleste.
3. Desactivaremos la casilla Show Frozen in Grey, pues ésta lo que provoca es que los elementos "congelados" no rendericen sus texturas si no que por contra se representan en un color gris pálido, y en nuestro caso precisamente queremos "ver" el plano.
4. Marcaremos la casilla Vertex Channel Display. En realidad esta operación sería opcional, pero si la llevamos a cabo (en conjunción con la siguiente) el objeto no quedará afectado por las luces y sombras del Viewport donde se represente, con lo cual siempre se verá con su máxima luminosidad.
5. Verificaremos que no está presionado el botón Shaded del Color Vertex.

Con esto tenemos traspasados al editor del 3ds Max los planos que hemos creado de nuestro modelo, correctamente centrados y escalados. Hemos obviado las vistas frontales y traseras, pero de haberlo requerido bastaría con haberlas construido de la misma forma que las laterales.

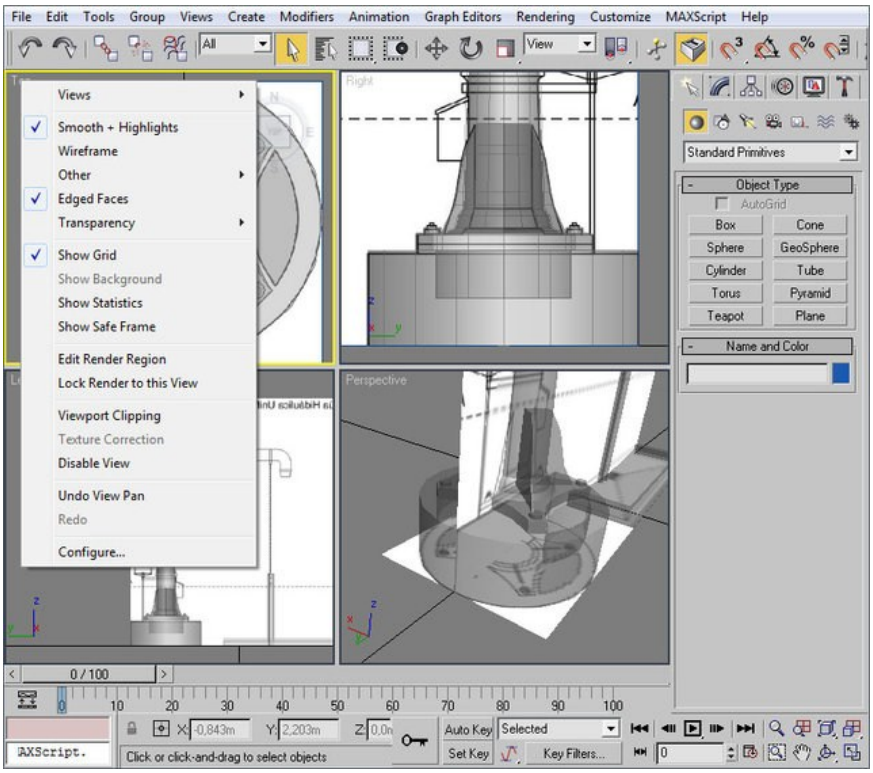
Atención
Para que las texturas de los planos se visualicen correctamente en los viewports, es importante haber realizado los ajustes en la configuración de las "Preference Settings" de 3ds, tal como se comentó en el capítulo "Personalización elemental" del Tutorial para iniciarse en 3ds Max, en lo relativo a los drivers de la tarjeta, accesible desde "Configure Drivers".

Ahora podemos empezar a crear elementos de nuestro modelo, y ajustarlos a su representación en el plano, con lo cual garantizaremos las medidas, proporciones y aspecto del modelo que realicemos:



Para dicho trabajo será necesario cambiar la visualización de los Viewports en los que vayamos a trabajar de "Wireframe" a "Smooth + Highlights", como se aprecia en la imagen anterior.

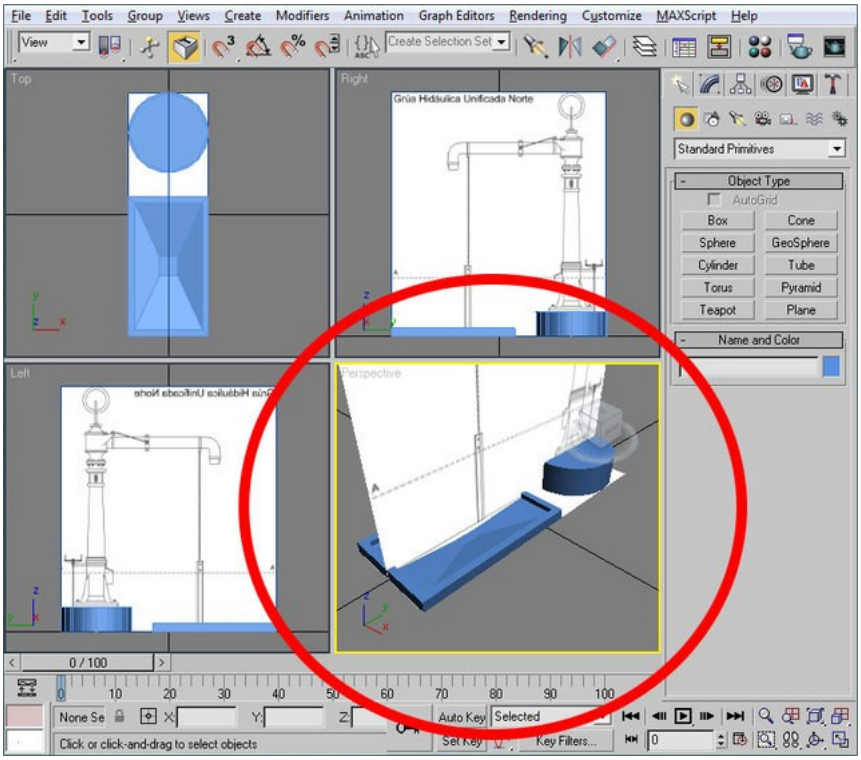
Mientras avanza nuestro modelo, podemos encontrarnos en la situación que una pieza ya realizada nos oculta parte del plano que necesitamos tener a la vista para seguir modelando otras piezas. En este caso bastará pulsar Alt + X, con la pieza molesta seleccionada, para convertirla temporalmente en transparente en el Viewport, y de esta forma poder apreciar el plano a la perfección. Evidentemente, una combinación de teclas: Ctrl + A y luego Alt + X provocará que todos los elementos de la escena se vuelvan transparentes .



Y en este último caso, si queremos visualizar las aristas del objeto transparente bastará con activar la opción "Edged Faces" de la visualización del Viewport.

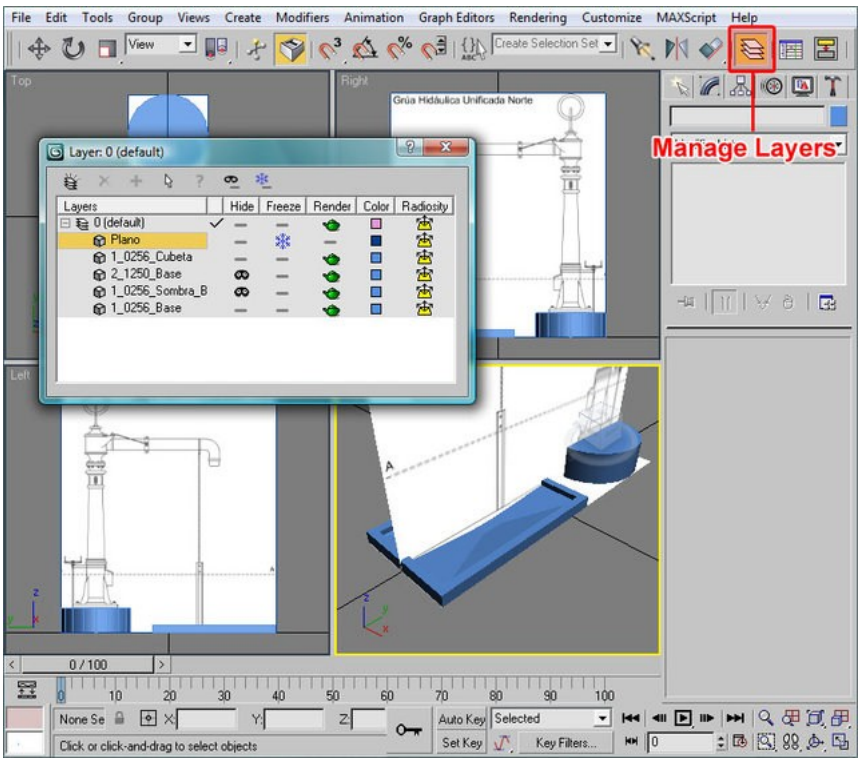
Cómo os podéis imaginar, una segunda pulsación de las teclas Alt + X restituirá la opacidad de los elementos que se encuentren seleccionados en ese momento.

Podemos trabajar en el modelo en las vistas ortogonales bastante bien, pero puede que los planos introducidos nos puedan molestar en algún momento en que queramos observar como va avanzando el modelo. En efecto, realizada la cubeta de nuestra grúa, no podemos apreciar bien su conjunto y cómo quedan las aristas en la vista en perspectiva:

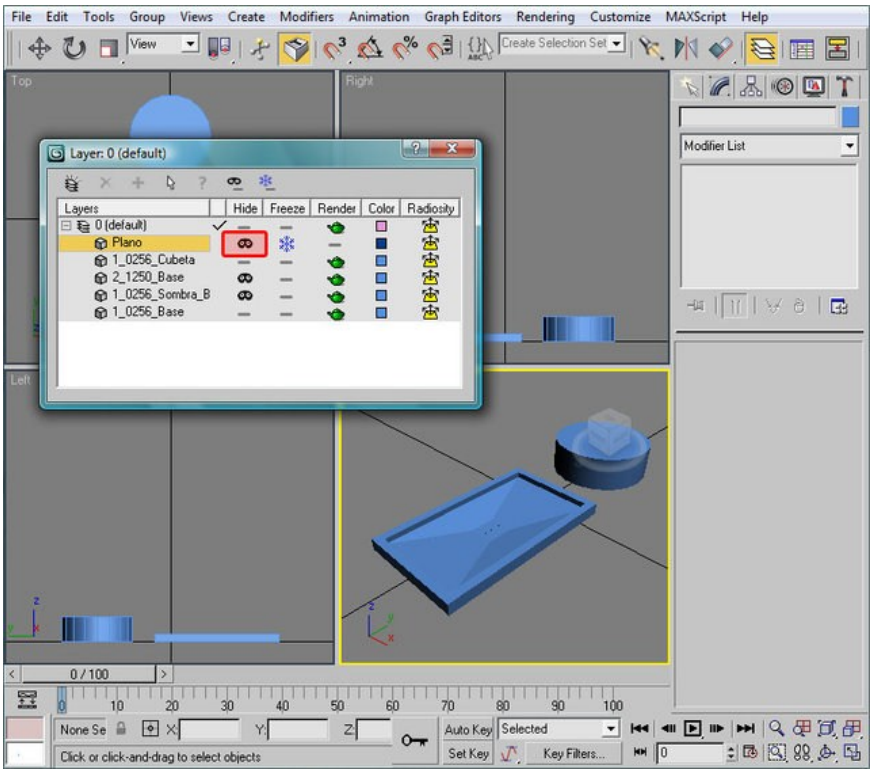


Para evitar esto nos interesaría poder esconder y mostrar el plano a nuestro antojo. Pero esta operación se nos muestra dificultosa dado que al estar este elemento congelado (Freeze) no se deja seleccionar. Podemos liberar todos los elementos congelados ("Unfreeze All" en el menú contextual del Viewport) para después seleccionarlo y ocultarlo, y una vez terminada la necesidad proceder al revés mostrando el elemento y volviéndolo a congelar, pero repetir todos estos pasos se nos puede antojar muy farragoso con el tiempo, por ello vamos a trabajar con las capas (Layers) de 3ds.

Podemos acceder a la gestión de capas (Manage Layer) en el menú Tools, o bien pulsando el botón "Manage Layers" de la "Main Toolbar":



Nos aparecerá una ventana con las capas de nuestra escena, en nuestro caso únicamente existirá una capa o layer, llamada "0 (default)". Podemos pulsar sobre el signo + al lado del nombre de la capa y se expandirá una lista con los objetos que contiene la capa, entre ellos se encuentra nuestro plano. Observamos que la ventana nos informa de una serie de atributos de estos objetos: Si son visibles o no (el antifaz), si están congelados o no (la estrella de hielo), si se renderizarán o no (la tetera), etc.. Estos atributos también están a nivel de la capa, y todos ellos los podemos modificar simplemente pulsando con el ratón sobre el icono. En nuestro caso, que queremos ocultar el plano, nos bastará con pulsar sobre la columna Hide a la altura del elemento:



El plano desaparecerá de las vistas a la vez que el icono del antifaz nos indica en la capa que ha pasado a estar oculto.

Desde esta ventana, que podemos si lo deseamos mantener abierta mientras trabajamos, podemos modificar estos atributos fácilmente de cualquier elemento de la escena: mostrarlos, ocultarlos, congelarlos, etc.

Por supuesto que podemos crear nuevas capas y repartir los elementos de la escena entre estas capas. Esto nos facilitará que, si trabajamos con los iconos a nivel de capa, podamos ocultar toda una capa a la vez si queremos, o congelarla, o hacer que no participe de la renderización, etc., de esta forma nos será más fácil trabajar en la escena sin problemas.

Aunque os he mostrado las posibilidades de las capas para el caso de la gestión de un elemento asociado a los planos de nuestro modelo, esta potente herramienta permite un manejo muy eficaz de todos los elementos de la escena, y no únicamente de un plano, pero sus posibilidades concretas a vuestros proyectos las decidiréis vosotros mismos.

Espero que estas indicaciones os sean útiles a la hora de modelar basándoos en planos.

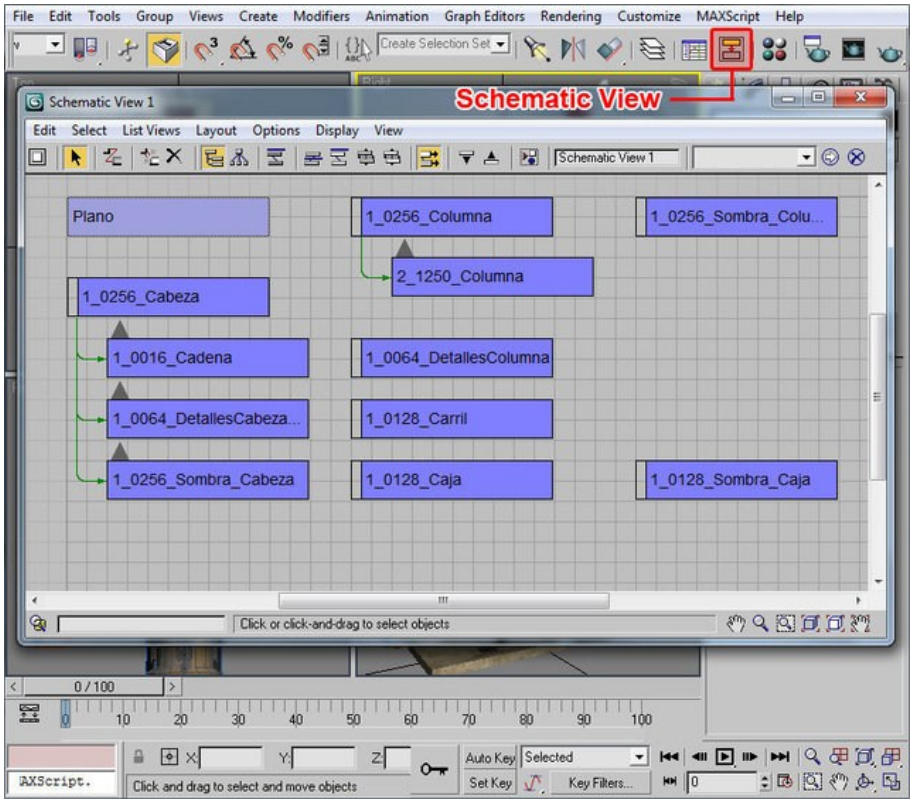
7. Animaciones

Aprovechando que he desarrollado el modelo de la grúa hidráulica, la haremos interactiva en RailWorks, para lo cual será imprescindible animar su brazo.

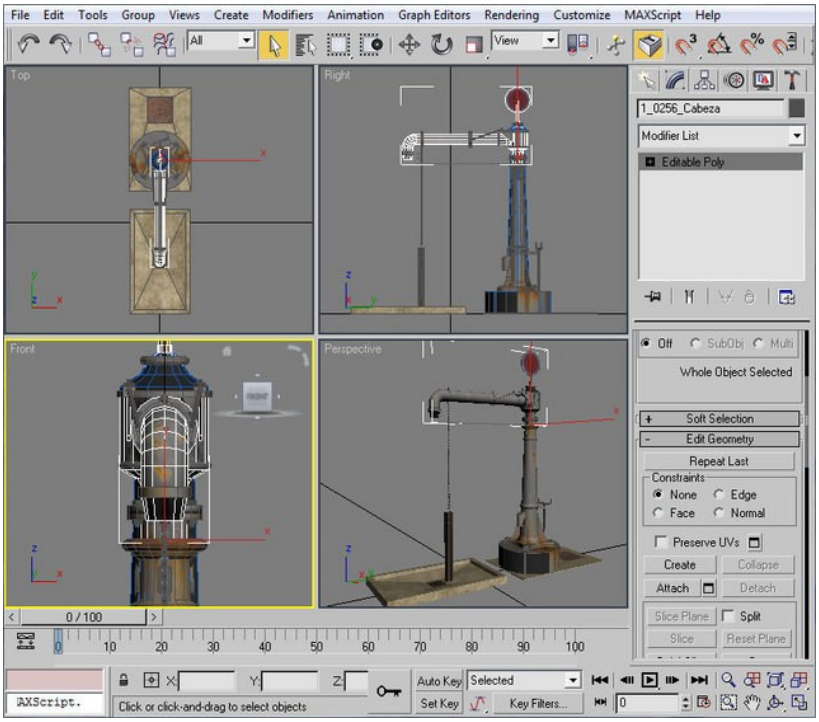
Cómo acciones previas al proceso de animación, el modelo deberemos tenerlo debidamente terminado, con todo el nivel de detalle deseado, diferentes LODs, y sombras dinámicas creadas.

Que estos elementos estén todos ellos creados no es estrictamente necesario pero si muy conveniente dado que la animación de algunos elementos deberá aplicarse, no únicamente al elemento principal, sino también a sus diferentes LODs que puedan existir y a los elementos de sombra dinámica, para que las sombras que proyecte dicho elemento estén también debidamente animadas.

Otro paso previo será establecer la jerarquía correcta de todos los elementos del modelo:

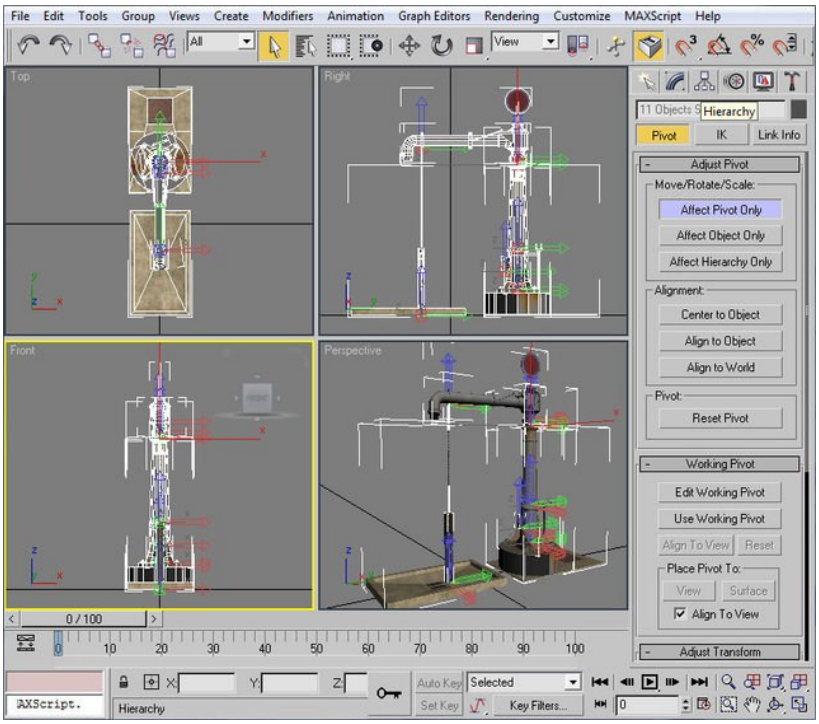


Abriremos la vista esquemática o "Schematic View", y nos aseguraremos que todos los elementos que deben participar en la animación dependan jerárquicamente de un único elemento "padre". En este caso, todos los elementos del brazo de la grúa, de la señal de disco, etc. , incluidas sombras o diferentes LODs, dependen de un elemento que he denominado "Cabeza", y que os muestro en esta imagen:

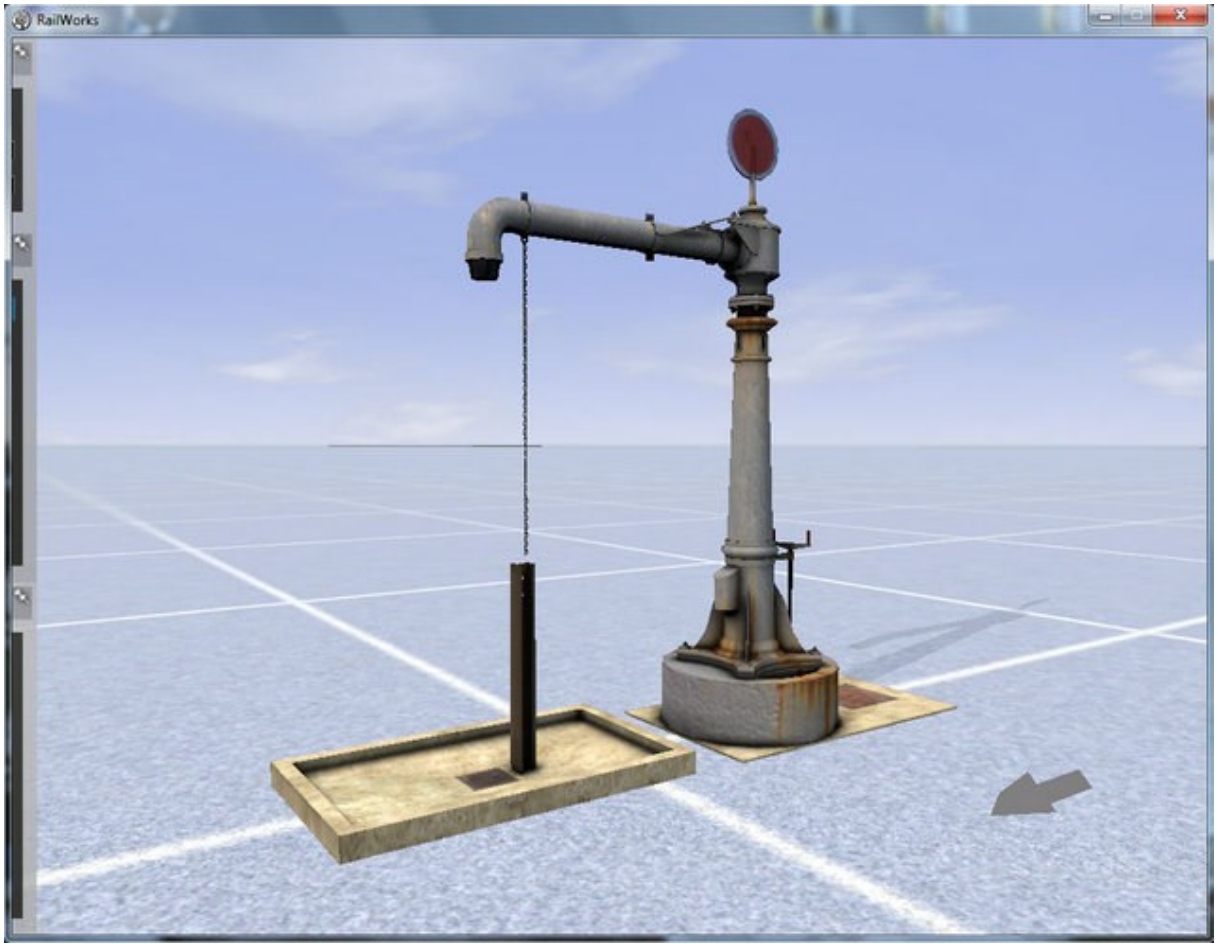


La "cabeza" de la grúa es el elemento sobre el que están unidos todos los elementos móviles y que tiene libertad de movimiento sobre la columna de la aguada.

Una última comprobación muy recomendable será la verificación de que todos los pivotes de los elementos de la escena estén correctamente alineados en el "mundo", es decir, las flechas rojas, que corresponden al eje X, apuntarán hacia la derecha visto el modelo de frente; las flechas verdes, que corresponden al eje Y, apuntarán hacia atrás; y las flechas azules, que corresponden al eje Z, apuntarán hacia arriba. Si hemos creado todos los elementos del modelo en el viewport "Top", esto será así y nos evitaremos problemas futuros. En efecto, si un elemento que vamos a animar tiene mal orientado su pivote, al aplicarle una transformación, por ejemplo una rotación en el eje Z, podemos encontrarnos que en la simulación dicha rotación se produzca en otro eje o en otro sentido al deseado.



Visto que todos estos requisitos los cumple nuestro modelo, deberemos efectuar una exportación y visualizar el modelo en el simulador "antes" de empezar a trabajar las animaciones, pues es importante comprobar que no existan errores o problemas previos y así eliminaremos complejidad a la operación.

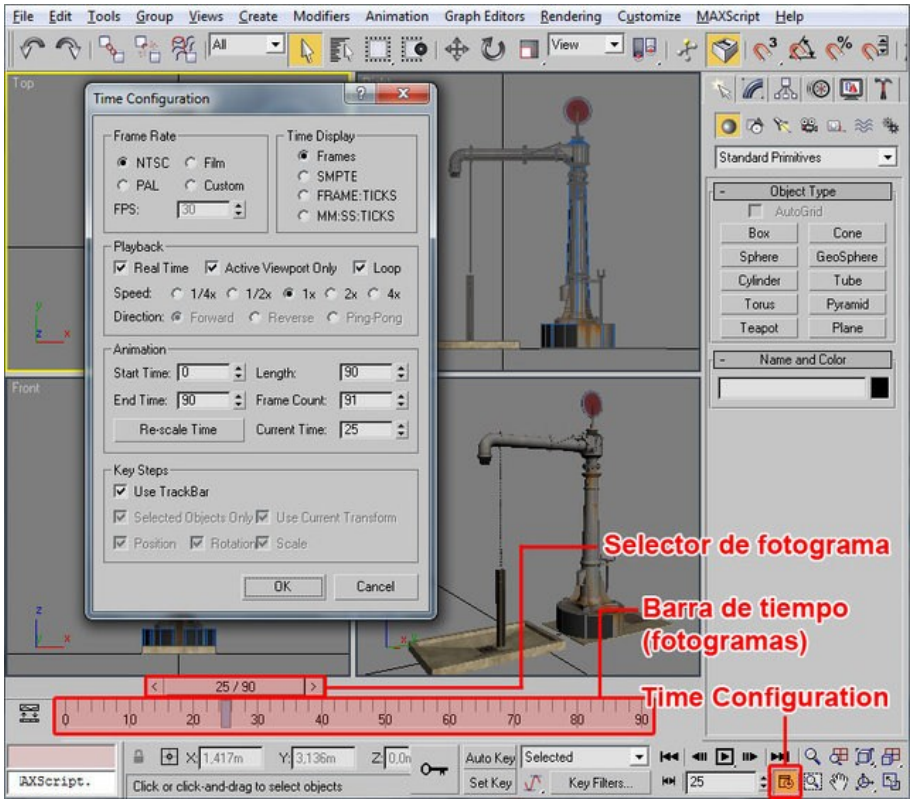


7.1. Control del tiempo de la animación

En primer lugar fijaremos el tiempo que durará la animación. Aunque más adelante podemos variar los parámetros de duración de la animación, nos será más simple si estos los determinamos a priori, y de esta forma nos evitaremos reescalados del tiempo.

Si bien en RailWorks hay algunos casos concretos de animaciones en que el simulador determina la duración total de las mismas (número de frames) o bien las claves que deben contener (luego veremos esto de las "claves"), en este caso, cómo en la mayoría de las animaciones en RW, tenemos libertad para establecer dichos parámetros. Por tanto vamos a definir éstos.

Abriremos la ventana "Time Configuration" pulsando el botón de la barra inferior de estado:



En esta ventana podemos determinar la duración de nuestra animación, y las unidades de tiempo que emplearemos. En ella observamos:

- **Frame Rate** - en esta sección debemos elegir la frecuencia de imágenes por segundo de nuestra animación. Hay tres ratios preestablecidos:
  - NTSC - Aplica a las normas de vídeo norteamericanas y determina un ratio de 30 frames (o imágenes) por segundo.
  - Film - Aplica al estándar en el cine y determina un ratio de 24 imágenes por segundo.
  - PAL - Hace referencia a las normas de vídeo europeas y establece 25 imágenes por segundo.

  Además de éstas, tenemos la opción Custom que nos permite establecer el ratio de imágenes por segundo que deseemos para nuestro caso particular. En principio cualquier ratio nos vendrá bien, teniendo presente siempre que el ojo humano aprecia como continua (sin "saltos") cualquier animación a partir de 24 imágenes por segundo. Dejaremos por tanto el valor NTCS (30 fps) que nos propone 3ds Max por defecto.

- **Time Display** - en este apartado podemos indicar las unidades en que se visualizará el tiempo:
  - **Frames**, por defecto se establece en Frames (fotogramas), es decir, si hemos elegido un ratio de 30 imágenes por segundo, cada segundo se representará en una escala de 30 frames, dos segundos serán 60 frames, etc..
  - **SMPTE**, este acrónimo (Society of Motion Picture and Television Engineers) es el formato estándar para la visualización en tiempo de trabajo de animación profesional. De izquierda a derecha, la norma SMPTE muestra: minutos, segundos y fotogramas, definida por dos puntos. Por ejemplo: 2:16:14 representa 2 minutos, 16 segundos, y 14 frames.

- FRAME:TICKS - Ticks es la unidad elemental de animación de 3ds. Un segundo se divide en 4800 ticks. Si el ratio de fotogramas por segundo se ha establecido en NTCS (30 fps) entonces cada fotograma se divide en 160 ticks, y en este caso el tiempo se muestra en Frames y en ciento sesentavos de frame. Personalmente no me aclaro con este formato.
- MM:SS:TICKS - En este formato el tiempo se muestra directamente en minutos, segundos y ticks (1/4800 segundos).

Los dos primeros formatos son los más claros, y para animaciones sencillas, cómo lo son casi todas en 3ds Max, el primero, Frames, es más que suficiente y es el que dejaré seleccionado.

- **Playback** - En esta sección controlaremos la forma en que se visualizará la animación cuando la ejecutemos.
- **Animation** - Y este es el apartado donde vamos a definir la duración de nuestra animación. En mi caso, pretendo que el giro del brazo de la grúa tarde 3 segundos en realizarse, por lo que a un ratio de 30 frames por segundo, la animación necesitará en total 90 frames para completarse. por lo que informaremos Start Time = 0 y End Time = 90.

Ya hemos establecido la duración total de la animación, y podemos comprobar que la barra de tiempo se ha ajustado a los 90 frames solicitados.

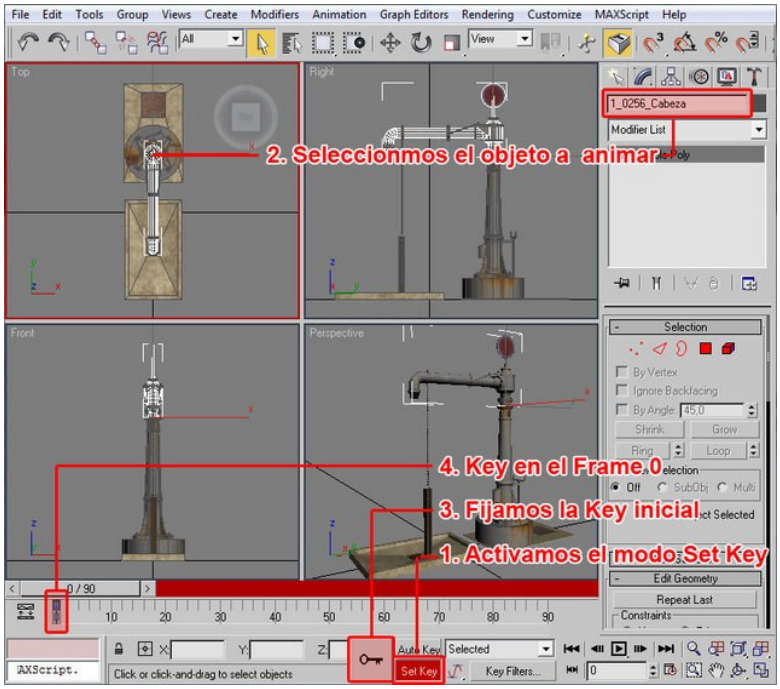
7.2. Estableciendo los puntos de animación (Keys)

Las transformaciones a un objeto o elemento de la escena se registran mediante un fotograma de inicio y de final de cada transformación. A dichos fotogramas que inician o terminan una transformación se les denominan **Keys**.

Por ejemplo, supongamos que tenemos un objeto que representa un ascensor que aun no se ha animado, y por tanto no tendremos fotogramas clave (o Keys). Si activamos el botón "Auto Key", desplazamos el selector de fotograma al frame 20, y desplazamos el ascensor en un viewport a lo largo del eje Z hasta una planta superior, se crearán Keys de posición en los frames 0 y 20. La Key en el fotograma 0 representa la posición del ascensor antes de que fuera trasladado, mientras que la Key en el fotograma 20 representa la posición del ascensor después de haber sido movido a lo largo del eje Z. Al reproducir la animación, el ascensor se moverá desde la planta baja al segundo piso a lo largo de 20 fotogramas.

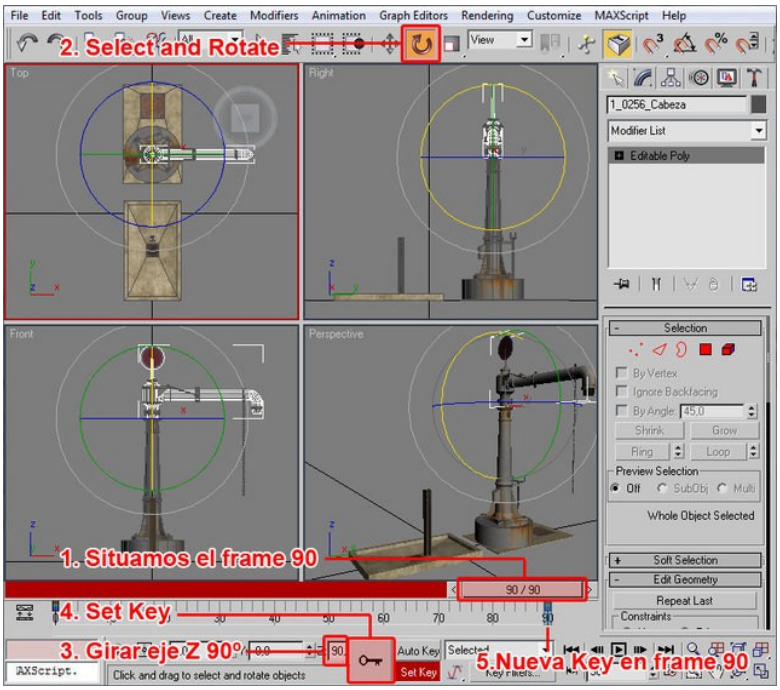
Con este ejemplo se ha pretendido aclarar el concepto de Key, y su diferencia con el fotograma o Frame, pero de pasada hemos introducido una forma muy sencilla y rápida de crear animaciones  : El botón "Auto Key".

A pesar de poder usar el mencionado método del "Auto Key", vamos a realizar la animación paso a paso sin él para entender mejor el proceso. En este caso:



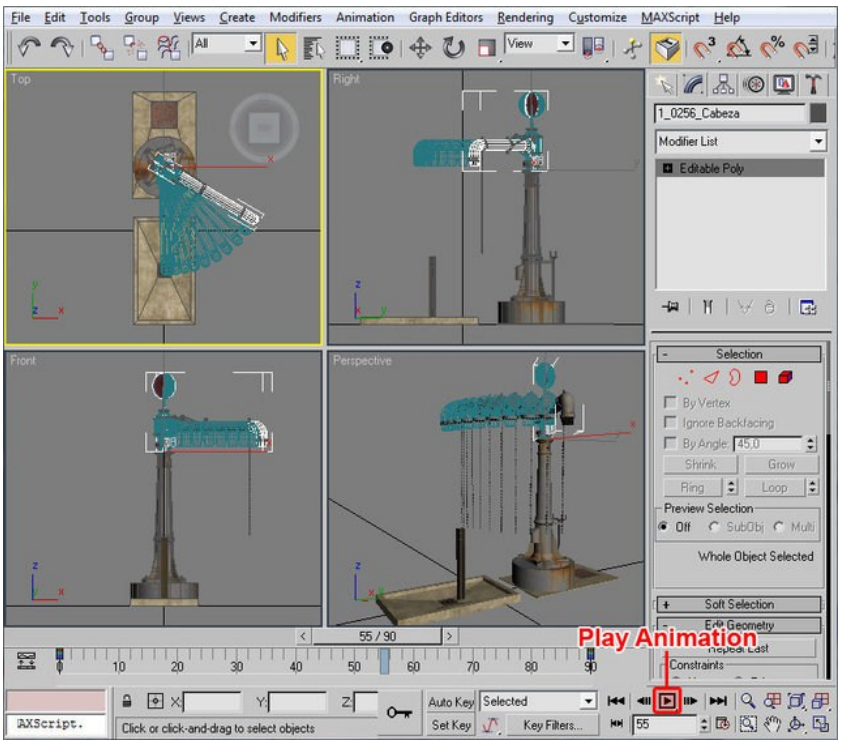
1. Activamos el **Set Key Mode**, pulsando el botón "Set Key". El botón se iluminará en color rojo, así cómo la Barra de tiempos y el marco del Viewport activo.
2. Con el modo activado, seleccionamos el objeto al que vamos a crearle sus Keys de transformación, en este caso la "cabeza" de la grúa, que incluye el brazo y de la cual dependen jerárquicamente la cadena y los demás detalles del conjunto móvil, así como la sombra de estos elementos.
3. Nos aseguramos de que el Selector de Fotogramas está en el Frame 0 y a continuación pulsamos la tecla grande "Set Key", para fijar la Key inicial de la cabeza de la grúa, o sea la posición de reposo o inicial de la animación.
4. Si observamos en la Barra de tiempos, en la frame 0 nos habrá aparecido una pequeña marca gris que indica la existencia en este frame de una Key de un objeto.

Una vez fijadas las condiciones iniciales vamos a determinar la situación final de la animación:



1. Desplazaremos el Selector de fotogramas hasta el frame 90, último de nuestra animación, para establecer los parámetros finales de ésta.
2. Seleccionamos la herramienta "Select and Rotate" para poder inducir la rotación deseada a la cabeza de la grúa
3. Asegurándonos que seguimos teniendo la cabeza de la grúa seleccionada, y el viewport "Top" activo, giramos este elemento 90º, bien pinchando en el círculo del viewport y arrastrando, o bien introduciendo el valor en el campo Z de la barra de Estado.
4. Una vez la posición final es la deseada pulsamos el botón grande "Set Key" para establecer una nueva Key con los valores de giro del elemento seleccionado.
5. Podemos verificar que aparece una nueva indicación de la existencia de una Key en el Frame 90. Terminando todo ello volveremos a pulsar el botón "Set Key Mode", para salir del modo de establecimiento de animación (los resaltes en rojo desaparecerán).

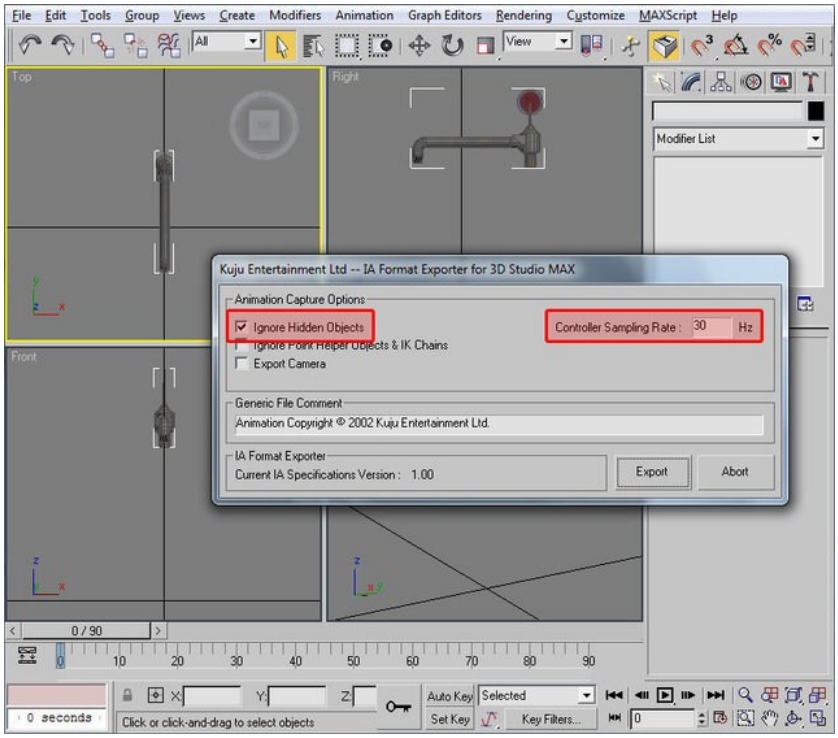
Ahora podemos comprobar el resultado de la animación pulsando en el botón Play Animation:



Una vez la animación sea de nuestro agrado procederemos a su exportación. Para ello, seleccionaremos el elemento que hemos animado, la cabeza de la grúa en nuestro caso, y esconderemos de la escena el resto de elementos, tanto los que no se animarán cómo los que si deberán animarse, pero a los que no hemos generado Keys (por ejemplo la cadena del brazo de la grúa) y cuya animación se produce por "animación directa" o FK (Forward Kinematics), es decir, se induce por las dependencias de los objetos: si la cadena tiene por padre a la cabeza de la grúa y ésta última se transforma, la cadena recibirá la transformación en su mismo grado.

Con el elemento "cabeza" visible (el único que ha recibido Keys de transformación) iremos al Menú **File -> Export...** y seleccionaremos el formato de exportación cómo "**Kuju Intermediate Animation (*.IA)**" y daremos un nombre al archivo que contendrá la animación.

Al aceptar nos aparecerá una ventana de diálogo donde:



- Nos aseguraremos que está marcada la casilla "Ignore Hidden Objects", dado que los elementos sin Keys no es necesario que participen de la animación. En objetos que tengan más de una animación, el hecho de que todos los objetos de un modelo participasen de todas las animaciones podría producir efectos indeseados en el conjunto (objetos que pierden la posición dentro de la animación).
- Indicaremos en el campo "Controller Sampling Rate" el valor de frames por segundo que definimos en su momento en la ventana "Time Configuration". Como entonces dejamos el Frame rate en NTSC, ahora indicaremos que el ratio es de 30 frames por segundo.

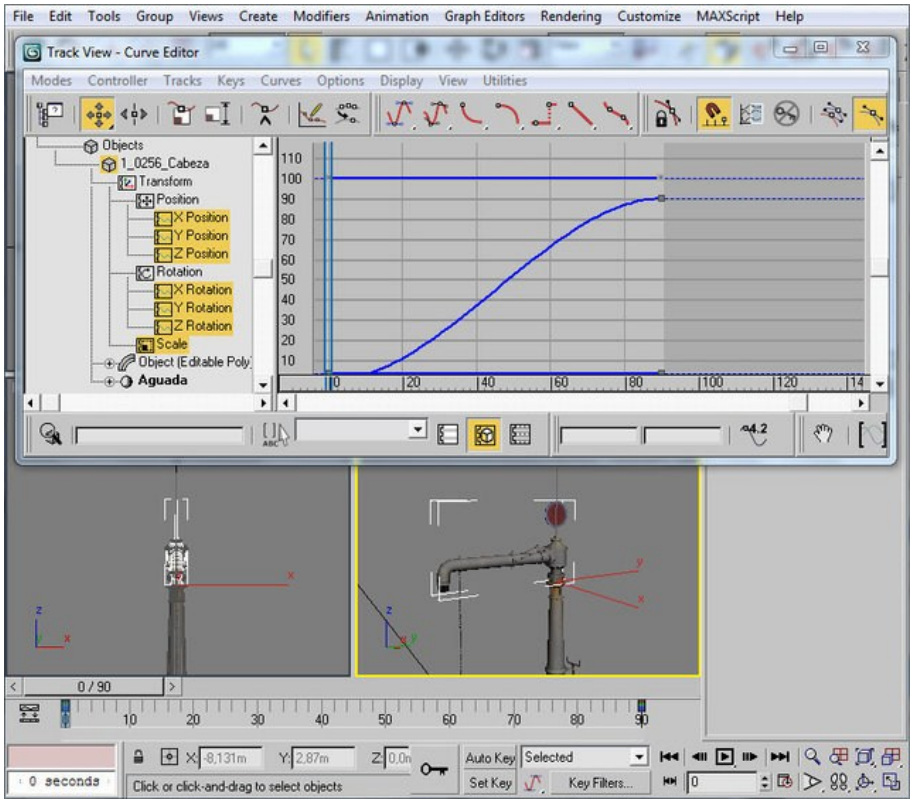
Una vez generado el archivo IA podremos crear un blueprint y ver nuestro trabajo en el simulador, pero eso será en un nuevo capítulo. Mientras tanto os dejo la primera visualización de la animación de la aguada:

(Véase vídeo: http://www.youtube.com/v/jtelcsVT9RU)

Próximamente mejoraremos la animación, incluiremos nuevos elementos y explicaremos cómo crear el blueprint para elementos de carga.

7.3. Trabajando con curvas de transformación - Track View.

Vamos a modificar el movimiento de la animación del brazo, en un intento de mejorarlo (claro). Y para ello abriremos el Track View mediante "**Menú -> Graph Editors -> Track View - Curve Editor**". Se nos abrirá una ventana con el contenido del Track View (Visor de Pistas):



El editor muestra un menú, una barra principal de botones, y una barra inferior de estado. En la zona central destacan dos áreas, a la izquierda el árbol de elementos y transformaciones de nuestra escena, y a la derecha el editor de curvas de transformación.

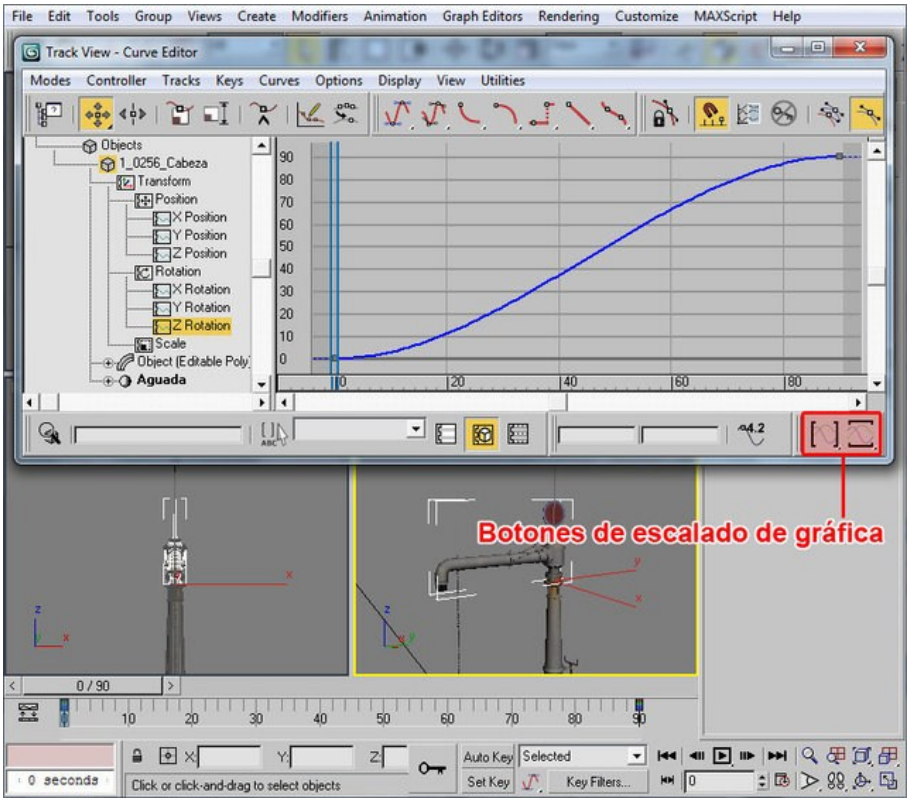
En el árbol de elementos tendremos que descender hasta llegar al nodo **Objects**, bajo el cual nos aparecerán aquellos objetos que tengamos seleccionados en los viewports. En este caso he seleccionado el elemento "1_0256_Cabeza".

El mencionado elemento nos muestra tres nodos: Transform, Object (Editable Poly) y Aguada:

- El primero representa las transformaciones básicas del elemento en sí: Posición (en cada uno de los tres ejes), Rotación (en cada uno de los tres ejes también) y Escalado. En este nodo nos encontraremos las transformaciones que podemos manipular para nuestras animaciones en RW.
- El segundo, que podemos desplegar, representa transformaciones que podemos realizar al tipo de objeto 3ds que represente. Cómo en este caso el elemento es una Editable Poly, nos permitirá un conjunto de modificaciones propias de este tipo de objeto.
- El último nodo hace referencia a los materiales que usa el objeto, en este caso un material que he denominado "aguada", y representa aquellas transformaciones que podemos realizar al material.

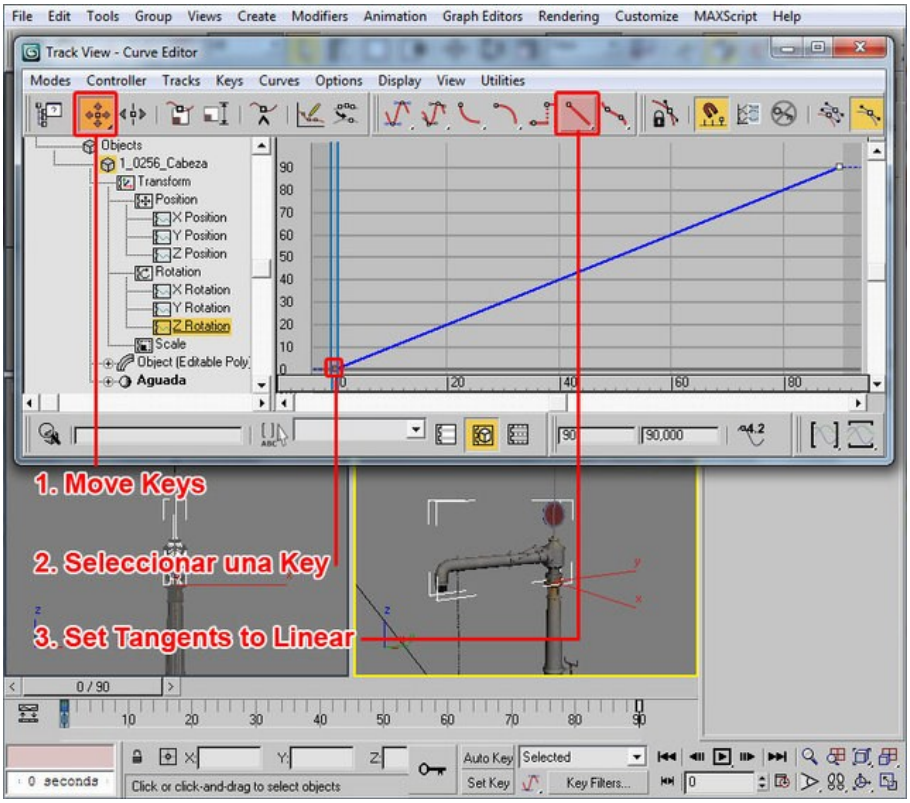
Railworks soporta animaciones que incluyan transformaciones del primer tipo, "Transform", de posición y de rotación. Por tanto nos centraremos en éstas.

Con la cabeza seleccionada, situaremos el árbol de forma que visualicemos cómodamente el nodo "1_0256_Cabeza" y marcaremos únicamente la transformación "Z Rotation", que es la que nos interesa seguir en su movimiento, ajustando la gráfica mediante los botones de escalado horizontal y vertical para su mejor visualización:



Observamos que nos ha quedado únicamente visible la curva que define la rotación de la cabeza en el eje Z, cómo era de esperar. Si nos fijamos en esta gráfica, vemos que el movimiento no se produce de forma lineal entre las dos posiciones, si no que comienza de forma más lenta para acelerarse después y terminar finalmente reduciendo progresivamente la velocidad. O sea, siguiendo la forma de la curva mostrada.

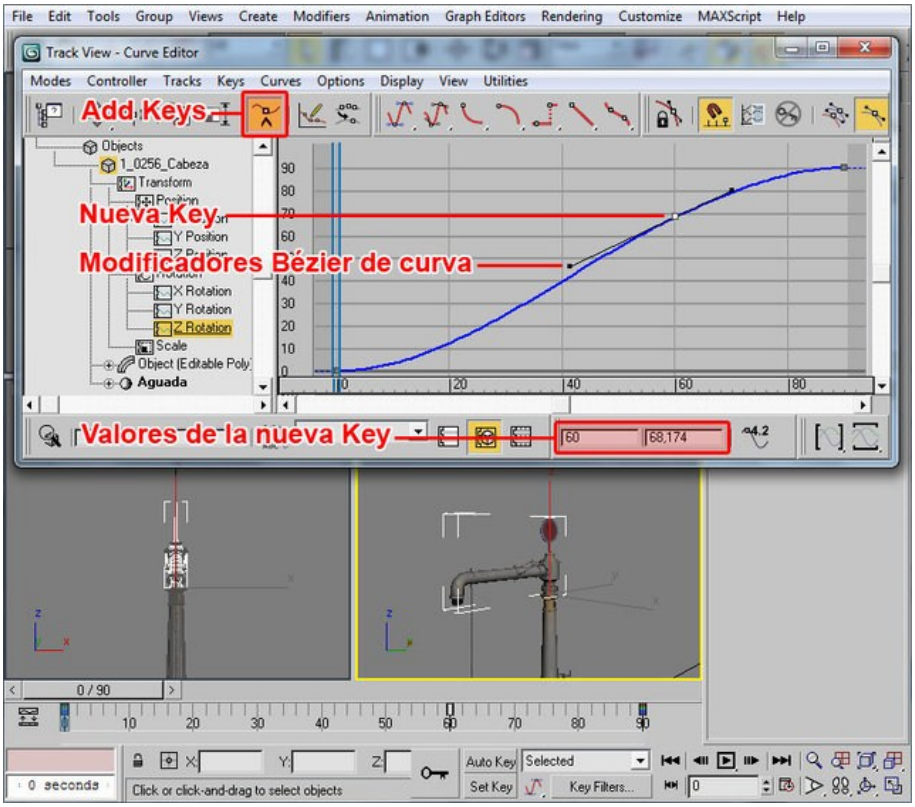
Podemos variar la forma de la curva de muchas formas, pero a modo de ejemplo haremos un movimiento uniforme del giro del brazo.



Con el botón "Move Keys" pulsado, seleccionaremos la Key de inicio del movimiento (pinchando con el ratón sobre el pequeño cuadrado al principio de la curva) para a continuación escoger entre uno de los diferentes modos de tangente, en particular "Set Tangents to Linear". Repetiremos la operación sobre la Key de final de movimiento, y con ello obtendremos una transformación representada por una línea totalmente recta, lo cual nos indica un movimiento uniforme a lo largo del tiempo que dure la animación.

El efecto original de aceleración y desaceleración del movimiento ya nos va bien en nuestro caso, por lo que desharemos estas últimas acciones, pero lo modificaremos para añadirle un movimiento de rebote al final del giro del brazo (cual pantógrafo que sube a buscar el hilo de contacto y rebota en él).

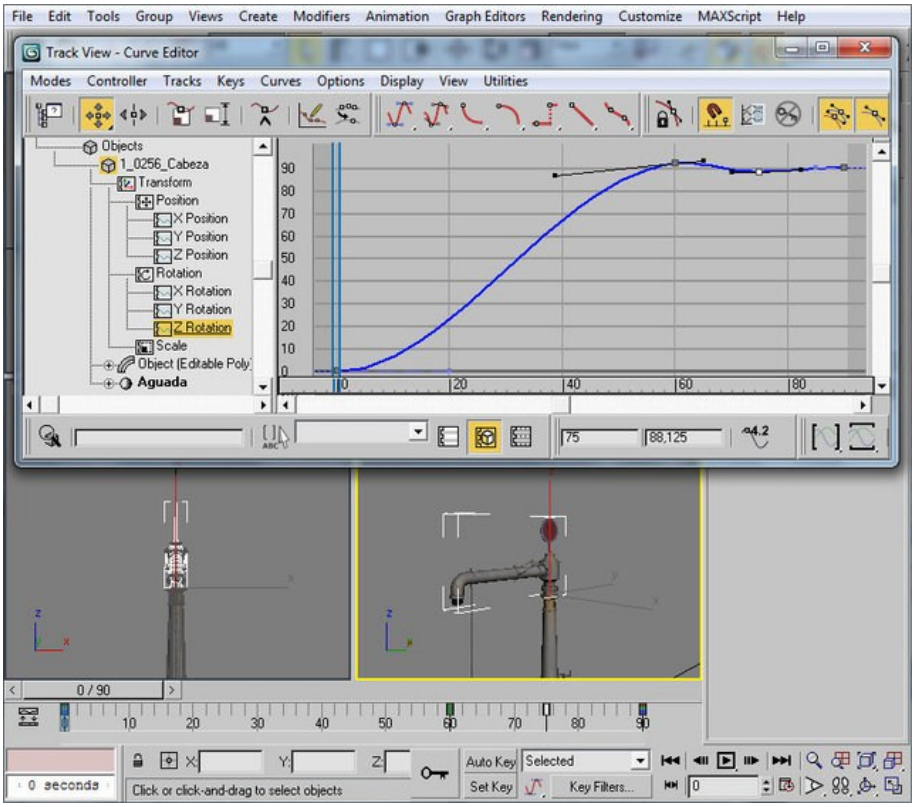
Empezaremos por crear una nueva Key sobre el fotograma 60. Para ello...



- Pulsamos el botón "**Add Keys**".
- Con el cursor pinchamos "sobre" algún punto de la curva cerca del valor de tiempo deseado, en este caso 60. Al pinchar aparecerá el pequeño cuadrado que indica la presencia de una nueva Key en la transformación. Podemos aprovechar para arrastrarla hasta una posición cercana a la deseada
- Podemos verificar en la Barra de Estado los parámetros de la Key: **frame** donde se encuentra y **valor**, en este caso grados dado que se trata de una rotación.
- A ambos extremos de la key aparecerán unas líneas rectas terminadas en un cuadrado negro. Son los modificadores del trazado por el método de Curva de Bézier. Pinchando en ellos y arrastrándolos modificaremos la tangente entre la Key y la curva, de manera que podemos darle al trazado de la curva la forma que deseemos.

Esta Key del frame 60 la elevaremos hasta un valor ligeramente superior a la posición 90, yo le he dado un giro total de 95 grados, un poco pasada respecto a la posición final.

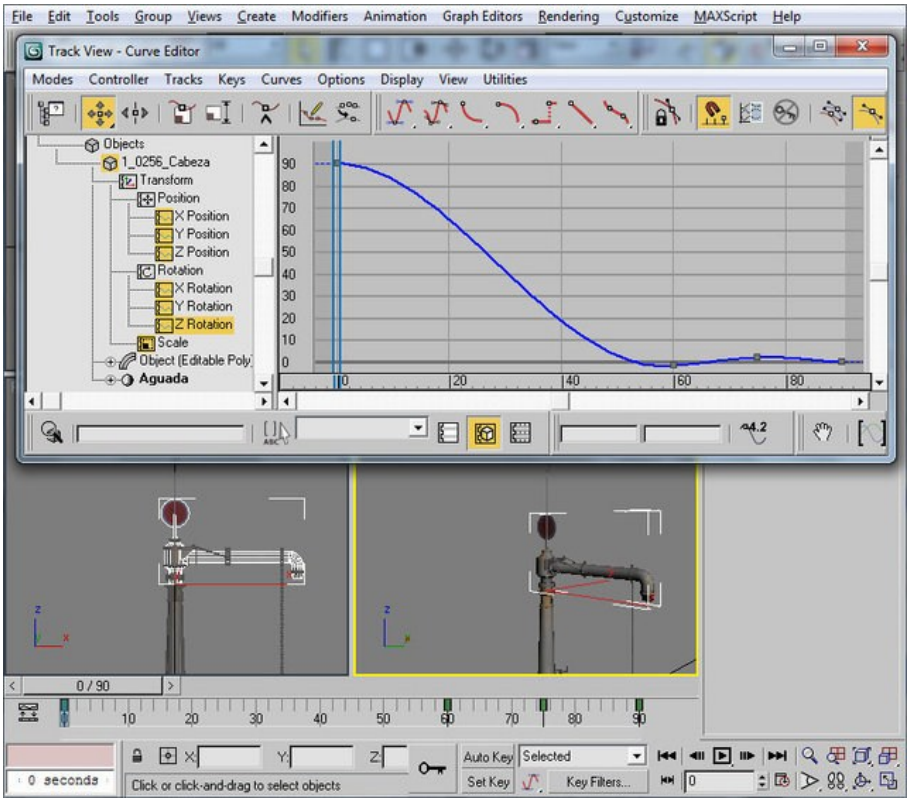
Ahora añadiremos una nueva Key sobre el frame 75 ligeramente por debajo de los 90 grados, por ejemplo 88º. La curva final nos ha quedado como muestro a continuación:



El movimiento de la grúa se realizará ahora en dos segundos, hasta el frame 60, algo más deprisa que antes, pero el brazo se pasará en su impulso y retrocederá ligeramente, volviendo a rebasar por poco el punto de detención final, al que se acercará ya mucho más despacio hasta detenerse.

Guardaremos la escena del modelo con la animación que acabamos de crear con un nombre diferente, y exportaremos la animación en el formato .IA.

Cómo en el caso de la grúa tendremos que realizar de igual forma el movimiento inverso para cuando tenga que regresar a su posición de origen, abriremos la escena original (sin la animación) y repetiremos estos pasos, pero desde un origen de la animación con el brazo girado 90º hasta la posición de reposo con un giro de 0º (posición original). La nueva curva será más o menos cómo ésta:



La curva sigue el mismo patrón que la anterior, pero en sentido contrario.

Guardamos también la escena con otro nombre para conservar esta segunda animación, y exportamos la nueva animación, igualmente con otro nombre, en formato .IA.

Por tanto, yo suelo terminar teniendo un archivo .max con el modelo sin animaciones, y otros tantos archivos .max con una animación diferente en cada uno de ellos, pero esto ya forma parte de la manera de trabajar de cada cual.

El resultado final es un poco más realista esta vez:

(Véase vídeo: http://www.youtube.com/v/zNZ1ZpD5vBU&hl)

Me diréis: si, vale, ya tenemos las animaciones exportadas, pero... ¿Cómo hacemos para ver el resultado en el simulador? Paciencia, esto lo veremos en la siguiente entrega.

8. Configurando un punto de carga o "Transfer Point"

Para terminar, vamos a ir efectuando la implementación de la grúa hidráulica en el simulador.

Los puntos de carga, o "Transfer Points", se definen mediante su respectivo blueprint, pero para comprender sus posibilidades hemos de entender su funcionamiento. Los Transfer Points pueden actuar en tres fases o etapas:

1. En primer lugar se desarrolla la **fase "previa a la carga"**. En esta fase el simulador activará una animación, si el modelo del punto de carga dispone de ella, bajo la clave "Activate", que podemos definir en el blueprint. La duración de esta fase es la de la animación asociada, si existe. En el caso de la aguada esta animación corresponderá al movimiento del brazo de la posición de reposo hasta situarse sobre el ténder de la locomotora.
2. A continuación se lleva a cabo la **fase de "carga"** propiamente dicha. Esta fase tiene una duración que puede ser variable dependiendo de parámetros de "velocidad de carga", que definimos en el blueprint, y de la capacidad del vehículo que recibe la carga. Durante esta fase se activan los elementos "Children" que tenga definidos el punto de carga en el blueprint. En nuestra aguada, la duración dependerá de cuanta agua por segundo hayamos definido como velocidad de carga, y de cuan vacío se encuentre el ténder, y como elemento "Children" de la aguada que deseamos que aparezca definiremos un gran chorro de agua.
3. Por último se desarrolla la **fase "posterior a la carga"**. En esta fase el simulador desactivará los elementos Childres que puedan haberse definido, y activará una animación, si el modelo del punto de carga dispone de ella, bajo la clave "Deactivate", que podemos definir en el blueprint. La duración de esta fase es la de la animación asociada, si existe. En el caso de la aguada esta animación corresponderá al movimiento del brazo de la posición sobre el ténder hasta regresar a la posición de reposo.

Cómo ya tenemos definidas las animaciones para las fases previa y posterior a la carga, tan sólo necesitaremos definir el violento chorro de agua espumosa que se mostrará durante la fase de carga saliendo de la boca de la aguada y cayendo sobre el ténder de la locomotora.

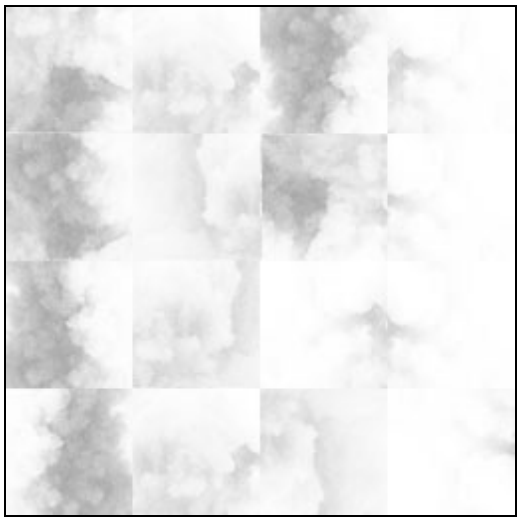
8.1. Un Emitter que represente agua.

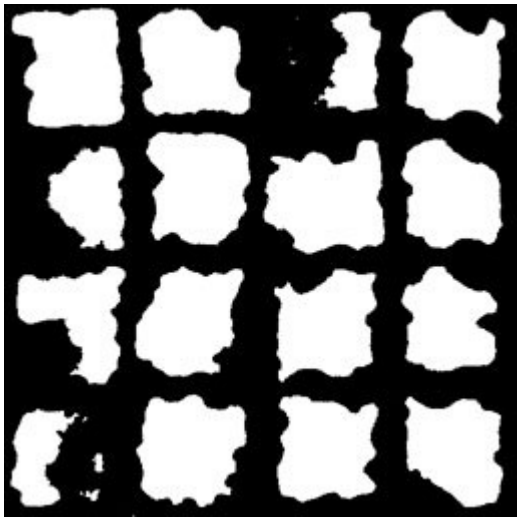
En su momento, en el capítulo "8. Efectos de humo" del tutorial "Avanzando en 3ds Max y RailWorks", usamos un generador de partículas, o Emitter, para representar un humo para la chimenea de nuestra casilla, pero este tipo de elementos puede ser muy versátil, y en este caso nos servirá para representar el chorro de agua que saliendo de la boca de la aguada caiga sobre el ténder.

Cómo en aquel caso, partiremos del blueprint denominado "Example_Smoke.xml" con un humo de ejemplo, que siempre guardo para empezar a definir generadores de partículas, pero en esta ocasión me limitaré a comentar únicamente los parámetros que se modificarán del XML de ejemplo citado.

Empezaremos duplicando el archivo mencionado y renombrando al nuevo como "Chorro_Agua.xml", por ejemplo. Lo abriremos con el Blueprint Editor y procederemos a su modificación.

Así mismo, he preparado un archivo de texturas de "humo" para el agua. He partido de una textura de humo real, aclarando mucho la imagen:





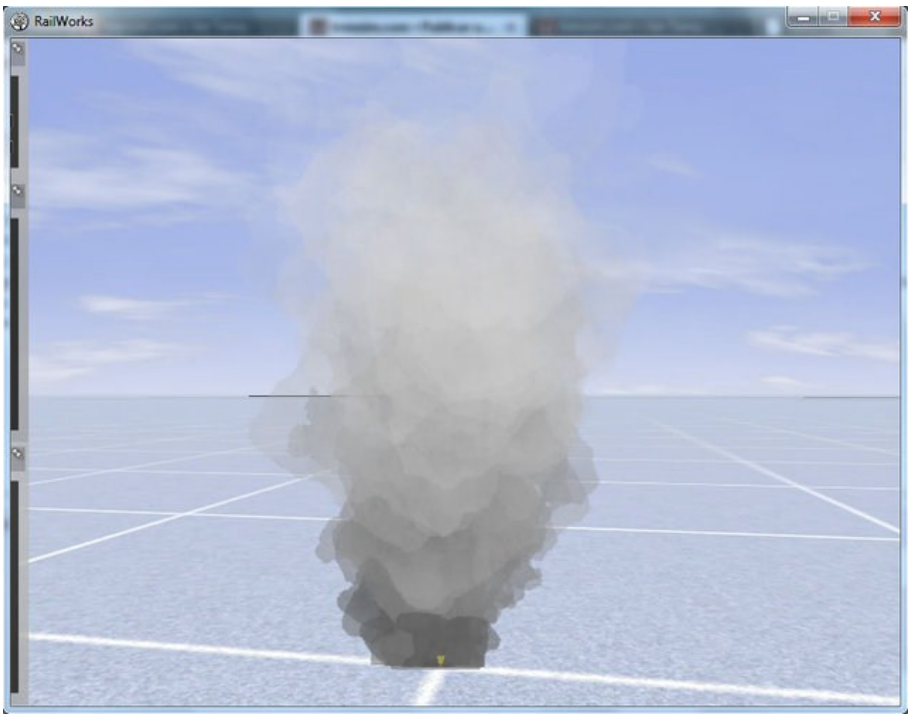
Y su canal Alpha lo he contrastado mucho para que las partículas tengan un aspecto más sólido que en el caso de un humo.

En primer lugar pulsaremos sobre el botón "Preview" para ver el humo en acción, y a partir de aquí poder comprobar visualmente cómo vamos adaptándolo a nuestras necesidades, e iremos alterando los parámetros:

- **Name** - Pondremos "Chorro agua"
- **Category** - Elegiremos "Exclude from browser list".
- Yo en la primera aproximación colapso los apartados "S emmiter properties" y "S particles properties" para facilitar el trabajo. Luego volveré a por ellos.
- **Tex ID** - Cambiaremos el nombre de la textura para el humo por "Particles\agua.ace", que es la textura que acabo de describir, y que da un aspecto más sólido a las partículas.

- **I row column size**- Deberemos comprobar que tiene el valor 4, pues la textura que vamos a usar tiene una matriz de 4x4. Este es un tamaño de matriz habitual.
- **F frame switch time** - Aquí expresaremos el tiempo (en segundos, creo) que indicará cada vez que se cambiará aleatoriamente la textura usada dentro de la matriz definida. Le he dado 0,1.
- **Shadow Type** - Deberá ser "None", pues no queremos sombra para el humo.
- **Detail level generation range** - Estos valores, que ya vimos para la casilla, determinan cuando será visible el humo según el nivel de detalle escogido por el usuario. Unos valores de 2 y 5 son suficientes en nuestro caso.

Ahora podemos volver a previsualizar el emitter:

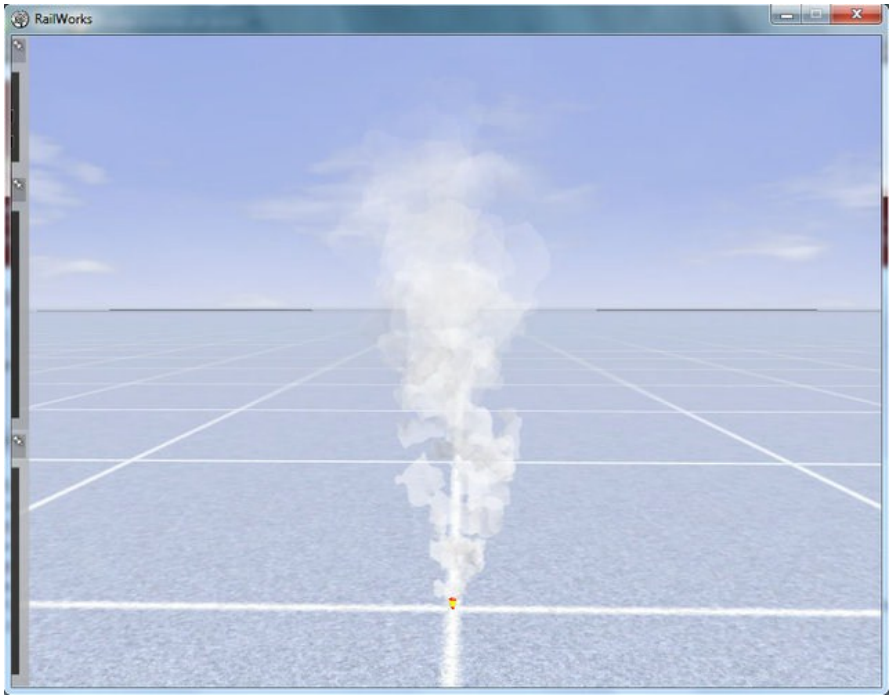


Como primera acción correctora, cambiaremos el color de las partículas modificando el valor:

- **Hc mid color**. Elegiremos el color blanco, informando R = 1, G = 1 y B = 1.

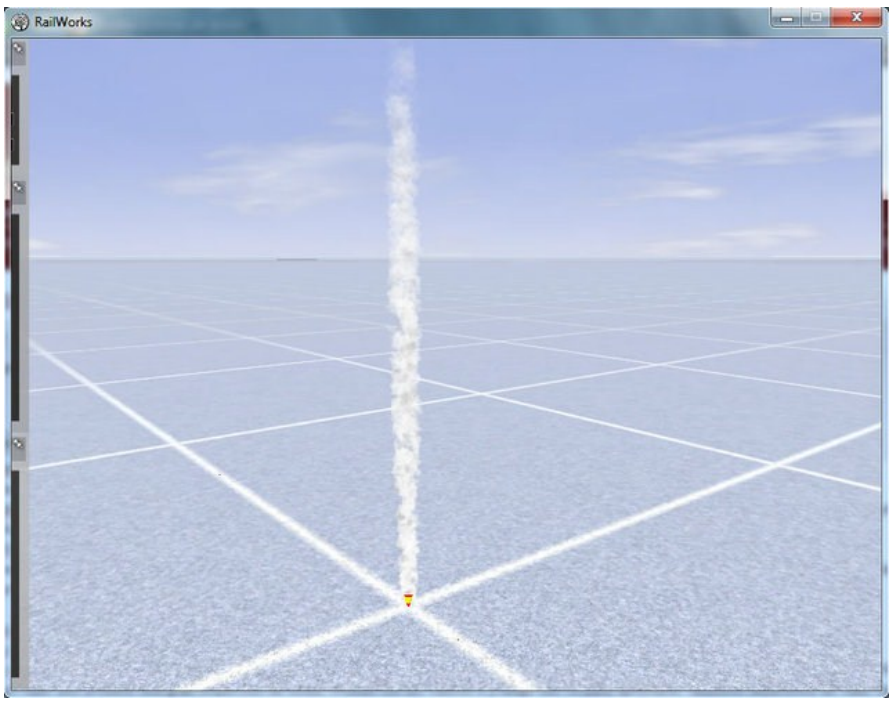
Así mismo reduciremos el tamaño de las partículas, pues estas son muy grandes, y dejaremos:

- **F size**. Tamaño de las partículas en el momento de su generación. Como queremos un humo discreto, dado el escaso tamaño de nuestra chimenea, le daremos un valor de .2.
- **F final size**. Tamaño que llegarán a alcanzar las partículas en el momento final de su vida. Dado el reducido tamaño inicial, le daremos una dispersión hasta un tamaño de .8.



Tal cómo lo tenemos ahora, creo que nos hará falta aumentar la cantidad de partículas, para dar consistencia al chorro de agua, y reducir la dispersión de éstas. Para lo primero modificaremos "I max particles" y sus parámetros asociados, y para lo segundo ajustaremos "F velocity variance":

- **I max particles**. Este es el número máximo de partículas vivas que mantendrá el emisor. Lo elevaremos a 500.
- **I particles to release**. cantidad de partículas emitidas simultáneamente. Lo elevaremos también a 10.
- **F velocity variance**. Determina la variabilidad en la velocidad y direcciones iniciales determinadas. Un valor muy bajo produce una columna uniforme de partículas, un valor alto produce una gran dispersión de éstas. Probad con los valores 0 y luego con 10 para apreciarlo. Nosotros dejaremos un valor de 0.1.



Nuestra columna de partículas se parece más a lo que deseamos: un chorro denso, uniforme y compacto de partículas de color blanco, imitando la espuma del agua a presión que sale por la boca de la aguada.

Tan sólo observamos un "pequeño" inconveniente, el chorro sale hacia el cielo y se eleva limpiamente hasta desaparecer, y creo recordar que ese no es precisamente el comportamiento del agua líquida. Veamos como corregirlo.

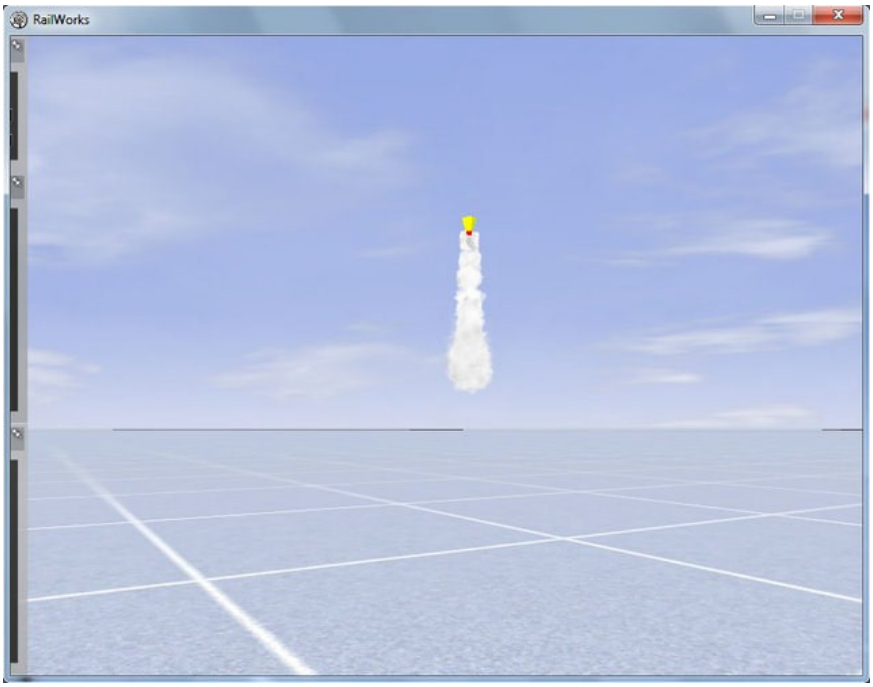
Para ajustar la dirección de las partículas deberemos tener presentes los parámetros "V init velocity" y "V gravity". El primero determina tanto la velocidad cómo la dirección iniciales de las partículas, mientras el segundo determina la aceleración y dirección que sufrirán éstas una vez libres del efecto de chorro que genera el primer parámetro. Por tanto, deberemos jugar tanto con los valores cómo con el signo de dichos parámetros, pues una velocidad en el eje Y (hacia arriba) si es negativa producirá que las partículas se generen hacia abajo. Veamoslo:

- **V init velocity**. Velocidad inicial de las partículas, definida en cada uno de los tres ejes X, Y y Z, y que por tanto también determina la dirección inicial. Como el eje Y es el que apunta hacia arriba (en el simulador) los valores de 0, 1 y 0 determinan que las partículas se emiten directamente hacia el cielo. Para nuestro agua daremos un valor de 0, -1.8 y 0 (valor negativo en el eje Y), con lo que ésta saldrá en dirección hacia abajo y con fuerza.
- **V Gravity**. Similar al anterior, determina la gravedad a la que estarán sometidas las partículas una vez abandonan el emisor. En realidad representa una aceleración (en cada uno de los ejes) que modifica la velocidad inicial de la partícula. Un valor de 0, -1 y 0 (valor negativo en el eje Y) haría que las partículas una vez han salido del emitter siguiesen cayendo hacia el suelo, pero en este caso se vería el chorro de agua por debajo de los ténderes durante el proceso de carga (cómo si estuviesen agujereados) y eso no es lo que queremos. Para nuestras partículas daremos un valor de 0, 0.6 y 0, con lo que dejaremos que éstas vuelvan a ascender suavemente hasta desaparecer y no sigan cayendo.
- **V 2S pread** (En realidad se denomina V2 Spread). Determina un valor de propagación en el plano horizontal del emisor. Afecta tan sólo a los dos ejes X e Y. Para nuestro chorro incrementaremos ligeramente los valores hasta 1.2, 1.2 y 1.

(He subido el punto del emitter, pues en caso contrario las partículas surgían bajo el suelo y no se apreciaban).

El efecto ya se parece mucho al deseado, si no fuera porque tras caer inicialmente, las partículas ascienden y tapan completamente el chorro inicial 😟. Acortaremos la vida de las partículas para evitar esto, y que éstas desaparezcan antes de que vuelvan a ascender:

- **F life cycle**. Duración de la vida de las partículas. Un valor de 2.5, la mitad del valor por defecto, hará que estas se comporten como deseamos.

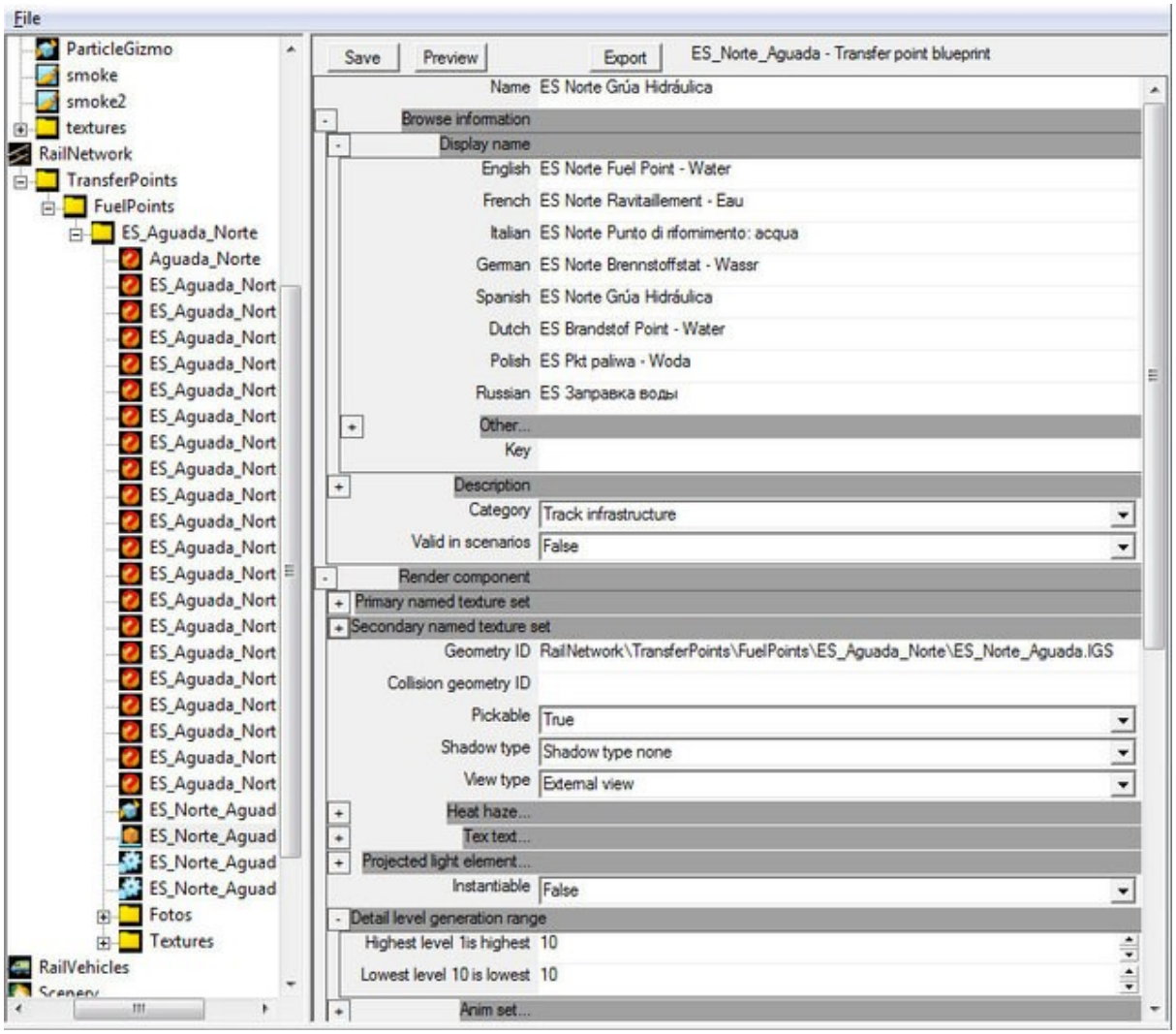


(Véase vídeo: http://www.youtube.com/v/uAG_kITJYLk&hl)

Satisfechos con nuestro emisor de partículas, lo guardaremos y exportaremos al simulador.

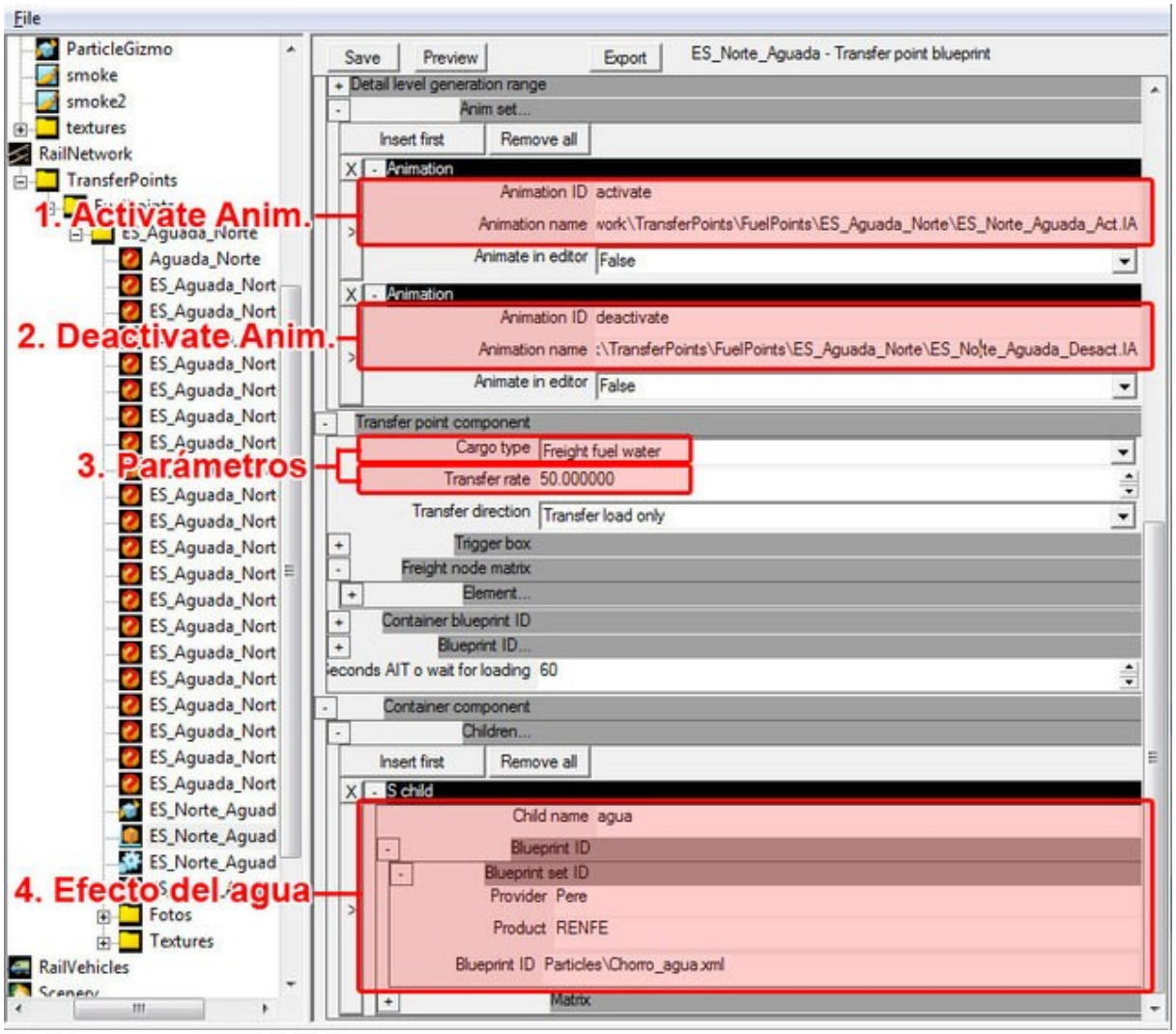
8.2. El Blueprint del punto de carga.

Crearemos un blueprint del tipo "Transfer point blueprint". La primera parte del mismo la rellenaremos como de costumbre en un objeto escénico:



Para los nombres del objeto en los diferentes idiomas encontraréis traductores en la red a gusto de todos.

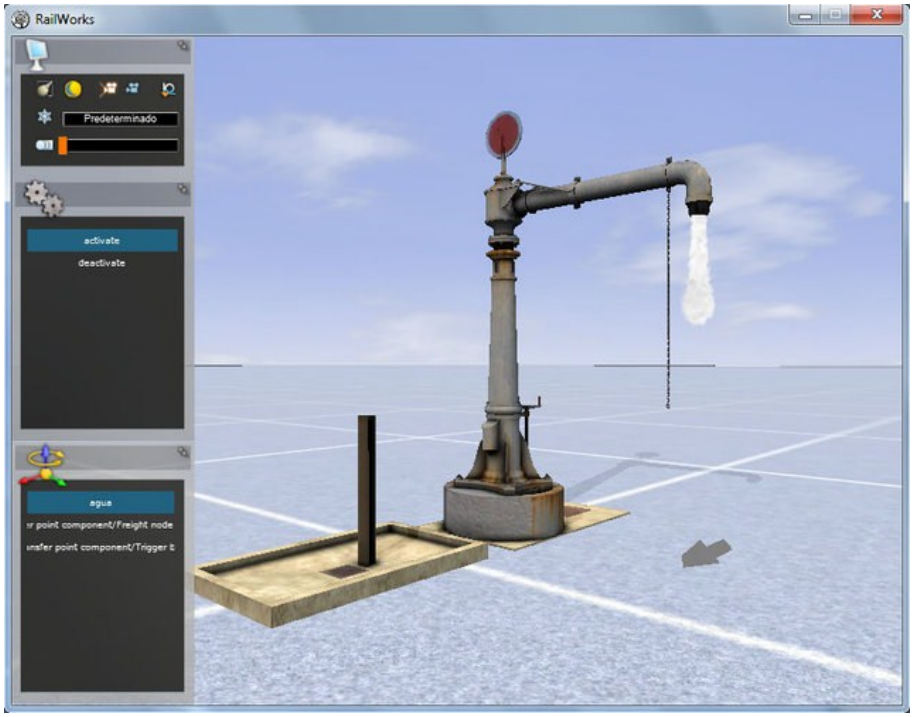
En la segunda parte del blueprint encontraremos los parámetros más interesantes para configurar un punto de carga en el simulador:



1. Bajo el encabezado **Anim Set...** deberemos insertar dos entradas para animaciones: La primera la definiremos bajo el **Animation ID = activate**, y en ella referenciaremos la animación de activación de la aguada, o sea el movimiento del brazo de la posición de reposo a la posición de carga.
2. La segunda la definiremos bajo el **Animation ID = deactivate**, y en ella referenciaremos la animación de desactivación de la aguada, o sea el movimiento del brazo de la posición de carga a la posición de reposo.
3. Luego, en **Transfer point components**, daremos los valores de comportamiento de la aguada:
    - **Cargo Type** – escogeremos "Freight fuel water".
    - **Transfer rate** - Expresados en galones por segundo, para un punto de repostaje de agua, indica la velocidad de carga de este elemento. Daremos un valor de 50 g/s, lo cual equivale a unos 190 litros por segundo. Un ténder unificado de Renfe, con capacidad para unos 33.000 litros, se llenará en unos 3 minutos.
    - **Transfer direction** - Existen puntos que sirven tanto para la carga cómo para la descarga (p.e. grúas de contenedores). Este no es el caso, por lo que elegiremos la opción "Transfer load only".
    - **Seconds AIT o wait for loading** - Tiempo que esperarán los trenes IA para efectuar la carga. Hemos especificado 60 segundos.
4. Finalmente en **Container component** insertaremos un elemento **Children** para el efecto de carga, en este caso el chorro de agua, y al que he denominado **Child name = agua**, aunque el nombre puede ser cualesquiera, y en **Blueprint ID** he especificado la

ruta de acceso al blueprint de partículas anterior, que en este caso he denominado "Chorro_agua.xml"

Con esto podemos previsualizar correctamente la aguada:



En la ventana central podemos activar las dos animaciones incluidas "activate" y "deactivate", y comprobar visualmente su correcto comportamiento.

Así mismo en la ventana inferior podemos seleccionar el elemento Child "agua", y al igual que en su día hicimos con el humo de la casilla, desplazarlo para ajustarlo a la boca de la grúa hidráulica cuando su brazo se halla totalmente perpendicular a la aguada.

Comprobado todo ello, podemos exportar al simulador y verificar el resultado final de nuestro objeto:

(Véase Vídeo: http://www.youtube.com/v/V14rlyv5GTw&hl)

Una aguada totalmente funcional y con un elemento, el agua, que en pocas ocasiones se ha reproducido en este tipo de objetos.

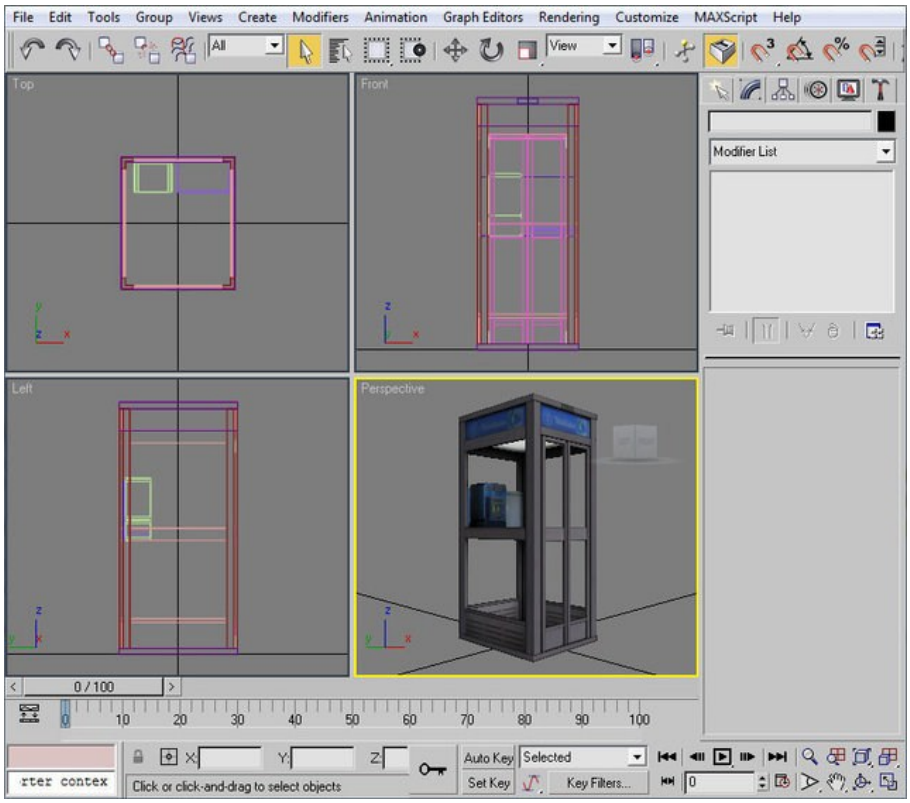
9. Metal - aluminio anodizado. Shader TrainBumpSpec.fx.

Este capítulo me gustaría dedicarlo a "jjlor", incansable autor de modelos 3D y de tutoriales sobre la materia, que me ha proporcionado información a menudo y a quien sin querer le he "pisado" el modelo que mostraré.

Vamos a representar un objeto que esencialmente estará formado por metal, en concreto por aluminio, y cristal. Es una combinación de materiales de muchas estructuras modernas, y que tiene sus peculiaridades para que sean creíbles en el simulador: los brillos del metal anodizado y los reflejos del cristal.

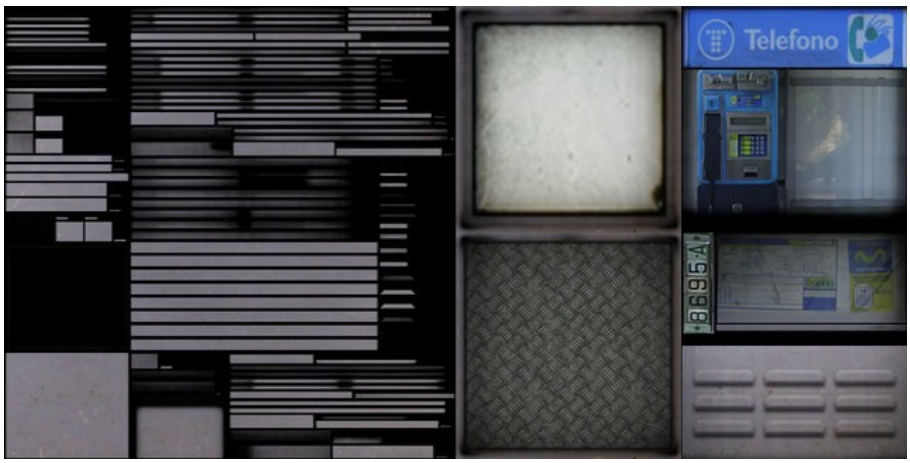
Y cómo elemento simple, pero suficiente, que incluya estos materiales podemos representar una cabina telefónica.

Empezaremos modelando la cabina, cómo no podía ser de otra manera:



Para texturar las partes metálicas he partido de la imagen de una superficie de aluminio cómo la que muestro:

Una vez debidamente complementada con otros elementos, he mapeado el modelo y he aplicado un Render to Texture, obteniendo un mapa diffuse semejante a este:



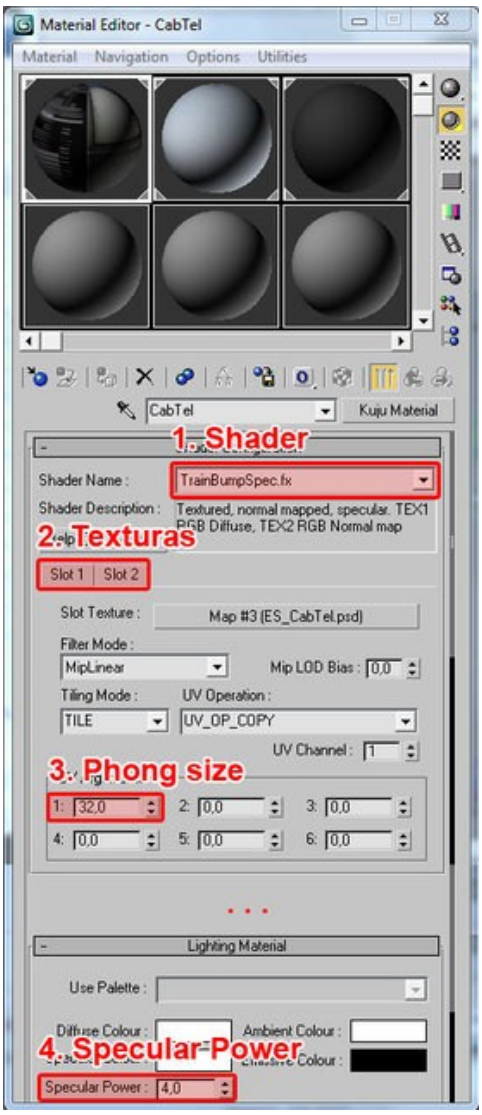
Con este acabado, y mediante el uso del universal shader "TrainBasicObjectDiffuse.fx", podemos proceder a exportar el modelo, con lo que obtendremos una vista en el simulador semejante a esta:

Hasta aquí hemos procedido cómo de costumbre. El resultado no está mal, pero rápidamente se observa que el modelo adolece de una falta total de brillos sobre las superficies metálicas, tal como cabría esperar de la pulida superficie del aluminio anodizado. ¿Qué hacer? Un nuevo shader vendrá en nuestro auxilio.

"**TrainBumpSpec.fx**" es un shader que implementa mapas de normales, para definir irregularidades de la superficie, y brillos especulares propios de superficies metálicas. Este shader demanda el uso de dos slots: en el primero debemos informar de una textura diffuse del material a emplear, y en el segundo podemos suministrar un mapa de normales para indicar irregularidades en la superficie de dicho material.

Nuestra cabina tiene unas superficies totalmente lisas, razón por la cual este mapa de normales no nos es necesario para el modelo, pero cómo el shader "debe" tener informado un mapa en su segundo slot, he preparado un mapa de normales totalmente "plano" para estos casos especiales. Os lo muestro:

Y ahora podemos proceder a modificar el material de nuestra cabina para trabajar el tema de los brillos en sus superficies. Abriremos el editor de materiales y modificaremos:



1. Seleccionaremos el shader "TrainBumpSpec.fx", que proporciona la capacidad de informar un mapa de normales al material, y también ajusta las propiedades del brillo especular del material.
2. Informaremos las texturas en ambos slots, en el slot 1 la textura diffuse de nuestro modelo, y en el slot 2 el mapa de normales, en este caso el mostrado más arriba que representa una superficie totalmente plana.
3. **Phong size** - Este parámetro determina la dispersión del brillo especular, y se informa en el Argumento UV número 1. Puede tomar valores entre 0 y 64, siendo 0 el valor que determina una mayor dispersión del brillo, cómo correspondería a una superficie "bruñida", y siendo 64 el valor que determina una menor dispersión del brillo, cómo correspondería por contra a una superficie "pulida". Para nuestro aluminio anodizado un valor de 32 será suficiente.
4. **Specular power** - Este parámetro determina la intensidad del brillo y se informa en el campo del mismo nombre que encontraremos en la sección "Lighting Material". Puede tomar valores entre 0 y 8, siendo 8 el valor de máximo brillo. En nuestro caso nuevamente tomaremos un valor intermedio y lo dejaremos en 4.

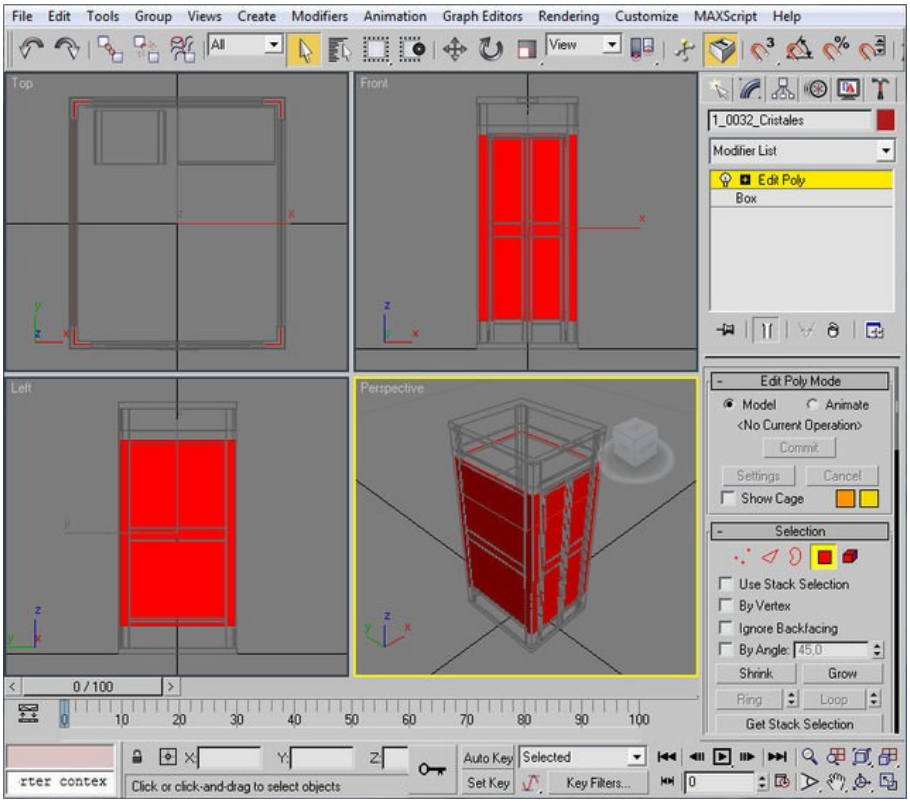
Con estos simples ajustes en el material, podemos volver a exportar el modelo y ver el resultado del cambio en el simulador:

La captura está tomada exactamente con la misma orientación y a la misma hora del día que la anterior, y claramente se observa que las superficies de aluminio orientadas hacia el sol despiden el brillo de la luz recibida, dando una mayor sensación de superficie metálica. También se ha aplicado el efecto de brillo especular en el plástico superior con el texto "Teléfono", por lo que éste también refleja el brillo de la luz que recibe.

Con todo ello la cabina ha mejorado, pero aun nos faltará simular de forma convincente los cristales que la envuelven, puesto que hasta ahora tan sólo aparenta no tener montado ni uno de ellos dada la total transparencia que presenta, así como la ausencia de reflejos en los mismos...

10. Cristales. Shader TrainGlass.fx.

Para representar los cristales de la cabina de teléfonos, primero los deberemos modelar mediante sus respectivos polígonos. En la siguiente imagen os muestro los cuatro polígonos que conformarán los cristales en las cuatro fachadas laterales de la cabina:



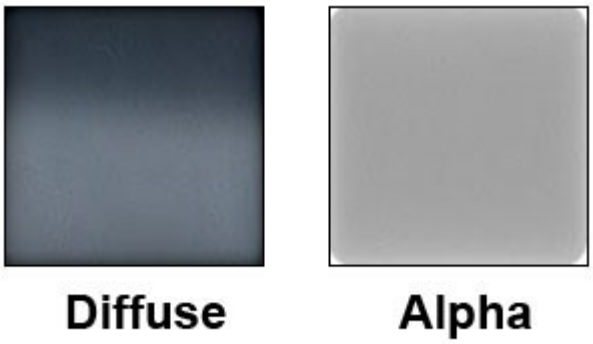
Están situados en el montante de cada ventana, cómo corresponde.

Ahora es cuando nos plantearemos cual debe ser la forma de definir el material adecuado para unos cristales. Nuevo shader:

"**TrainGlass.fx**" es un shader específico para cristales, con efecto de brillo sobre su superficie (al igual que en el caso de los metales) reflejos del entorno sobre ellos y transparencia y coloración graduables. Su uso es recomendable para aquellos cristales que estarán cerca de la cámara del simulador, puesto que para un edificio que se verá siempre a lo lejos no es preciso malgastar los recursos de nuestro ordenador en su representación, y en particular ventanas de vehículos: locomotoras, coches y vagones, edificaciones ferroviarias cercanas al trazado: marquesinas, estaciones, torres de enclavamientos, etc., y en general edificios modernos llenos de cristal .

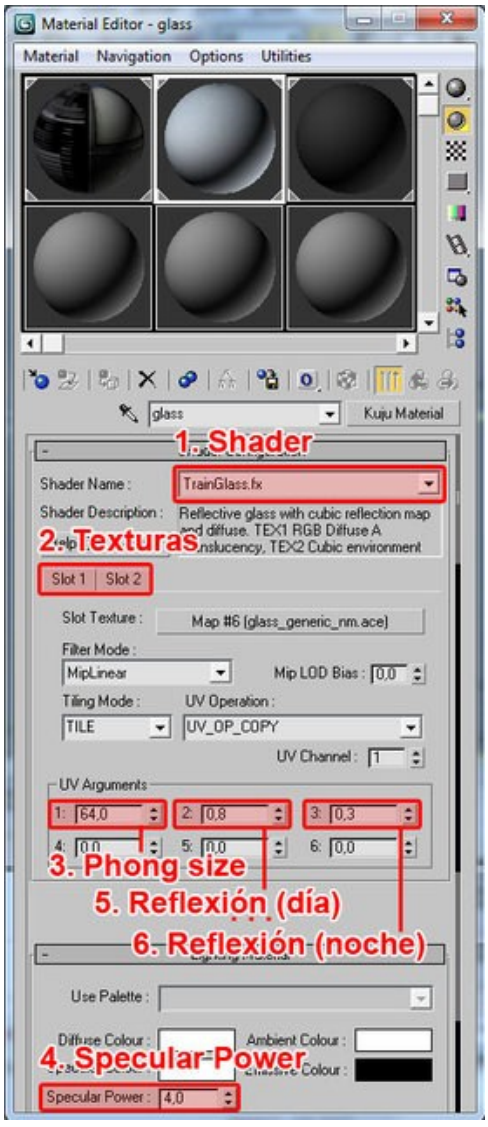
Este shader tiene dos slots, pero vamos a necesitar tan sólo una textura para el slot 1, dado que el slot 2 está destinado a contener una textura cualquiera para darle al shader una ubicación para un mapa de entorno que calculará él dinámicamente durante la simulación.

Vamos a analizar la textura a usar en el slot 1:



Se trata de una textura diffuse que representará el "tinte" que recibirá el cristal, ese degradado azulado en mi caso, con un canal "alpha" que determina la cantidad de transparencia u opacidad de dicho tinte sobre el cristal, teniendo presente que el negro (valor 0) representa una transparencia absoluta, semejante a haber practicado un agujero en el cristal, y el color blanco (valor 255) será un opacidad intensa, no total, de aproximadamente el 50%. En mi caso el canal alpha tiene un gris claro (valor 160) que producirá una tenue tinción del cristal.

Y ahora podemos proceder a modificar el material de nuestra cabina para trabajar el tema de los brillos en sus superficies. Abriremos el editor de materiales y modificaremos:



1. Seleccionaremos el shader "TrainGlass.fx", que proporciona la capacidad de informar un mapa diffuse con transparencia en el canal "alpha", un mapa "dummy" para calcular el mapa de entorno (de reflexión), y también ajusta las propiedades del brillo especular del material y de la reflexión.
2. Informaremos las texturas en ambos slots, en el slot 1 la textura diffuse de nuestro cristal tal cómo la he mostrado antes, y en el slot 2 repetiremos esta misma textura dado que tan sólo debemos proporcionar una ubicación para que el simulador calcule el mapa de entorno que se reflejará en nuestro cristal.
3. Phong size - Este parámetro, al igual que en el caso del metal, determina la dispersión del brillo especular, y se informa en el Argumento UV número 1. Puede tomar valores entre 0 y 64, siendo 0 el valor que determina una mayor dispersión del brillo, cómo correspondería a una superficie algo mate del cristal, y siendo 64 el valor que determina una menor dispersión del brillo, cómo correspondería a un cristal perfectamente pulido. Para nuestro caso un valor de 64, cómo corresponde a la pulida superficie de un cristal, será apropiado.
4. Specular power - Este parámetro determina la intensidad del brillo y se informa en el campo del mismo nombre que encontraremos en la sección "Lighting Material". Puede tomar valores entre 0 y 8, siendo 8 el valor de máximo

brillo. En nuestro caso nuevamente tomaremos un valor intermedio y lo dejaremos en 4.

5. **Intensidad de la reflexión durante el día** - Este parámetro determina el grado en que el mapa de entorno aparecerá reflejado en el cristal, y se informa en el Argumento UV número 2. Puede tomar valores entre 0 y 1, siendo 0 el valor que determina una ausencia de reflejo sobre el cristal, y siendo 1 el valor que determina un reflejo máximo sobre el cristal. Para nuestro caso un valor de entre 0,5 y 0,8 será el apropiado (dependiendo del grado de limpieza que queramos dar al cristal).
6. **Intensidad de la reflexión durante la noche** - Este parámetro determina el grado en que el mapa de entorno aparecerá reflejado en el cristal durante la noche, y se informa en el Argumento UV número 3. Actúa de la misma manera que el parámetro anterior tomando valores entre 0 y 1, siendo 0 el valor que determina una ausencia de reflejo sobre el cristal, y siendo 1 el valor que determina un reflejo máximo sobre el cristal. La existencia de este parámetro, que parece redundante respecto a la anterior, es debida a que el reflejo durante la noche, en condiciones de poca luz, es mucho más acusado que durante el día, y conviene tener un valor inferior al que se usará durante el día para evitar efectos desagradables. Para nuestro caso un valor de entre 0,2 y 0,3 será el apropiado.
7. Un último punto que hay que tener presente y no olvidar, dar a **Z-Bufer Mode** el valor "TEST ONLY", de igual forma que para todos los materiales que incluyan transparencias.

Con estos simples ajustes en el material, podemos volver a exportar el modelo y ver el resultado del cambio en el simulador:

La total transparencia anterior de los cristales ha desaparecido, lo cual ya ha mejorado el aspecto, y el sutil reflejo del entorno en los mismos le añade un grado de verosimilitud a la escena.

En el presente caso no sería estrictamente necesario, pero podemos duplicar los polígonos de las ventanas y voltearlos ("fliparlos") para reproducir también los reflejos en el interior de la cabina. El material soporta perfectamente diferentes niveles de transparencias, unas tras otras, sin provocar interferencias entre objetos que las incluyan. Los que hayan modelado objetos

con semitransparencias (canal alpha) en otras plataformas sabrán agradecer el no tener que preocuparse por la ordenación de los canales alpha para evitar que otros objetos desaparezcan tras las ventanas del nuestro.

Un re-exportación del modelo con "dobles" cristales nos muestra que ahora también se producen brillos y reflejos en el lado interior de los cristales de la cabina:

A modo de resumen, muestro juntas las cuatro fases de desarrollo de la cabina con la adición de los diferentes efectos de brillos en metal, cristales exteriores y cristales interiores, siempre desde una misma ubicación y ángulo de cámara, y a una misma hora del día:

Para finalizar un par de vistas in-game del objeto relacionándose con otros objetos de su entorno. Vista exterior:

Y vista a través de los cristales desde el interior:

Espero que os sea útil y tan sólo cabe recordar que todo esto es de aplicación en material móvil.

Con este capítulo doy por finalizada esta tercera entrega de tutoriales. En ellos hay vertida mucha ilusión en el sentido de que estos pasos sirvan para ver algún día creaciones de nuevos autores.

No se trata ni mucho menos de todo lo que se puede llegar a realizar en RailWorks con estas herramientas, pero eso sería una tarea casi imposible. De momento creo que hay material más que suficiente para divertirse practicando. Y sobre todo recordemos que en las pequeñas realizaciones también hay mucha satisfacción... que no todo han de ser Big Boys  .

Ánimo y suerte.