Mod Tools –
Object Editor Manual

 Version 2.1

Table of Contents

Introduction

The object editor is a proprietary tool designed by NovaLogic to modify 3ds max® (3D Studio Max) objects for use in our action games. Using this tool, you can assign a number of individual characteristics to an art object such as material, luminescence, texture animations, part animations, and transparency values.

This manual assumes that the user has a strong working knowledge of both 3D Studio Max and Adobe® Photoshop®.

File Standards

Before you can begin to work with files in the Object Editor, you need to make sure that any objects exported from 3D Studio Max (3.1 or higher) are properly standardized and are in the correct format. Usable file types are .ASE, .TGA, and .PCX. All files must be named with 8.3 file names

Within 3D Studio Max, all geometry must be named according to the following lists. For objects with more complicated geometry, adhere to these names.

Generic objects

- Upg01 ground… Center object, defines ground line
- 01 Name… All geometry that comprises the main object
- CB01 Name… All geometry that comprises the collision of the main object

Attachments

- 02 DoorName… Door02 (or any other object dependent upon the Main Object
- ~01a HingeName… Attachment point for Door02 to Main Object (01)
- _02 CenterName… Center for Door02 *Note that this object and the attachment point should share the exact same pivot location.
- CB02 DoorName… Collision box for Door02
- CD02 DoorName… Door Activation Space

* Continue in numerical order for additional objects (such as doors or windows) that pivot; 03, 04, etc.

All geometry pivots with respect to main object 01 shall use ~01a for attachment points. For all objects that pivot with respect to objects other than 01 use respective numbers. Use attachment points that correspond to the object to which the geometry attaches, ~02a for attaching to object 02, etc.

Overall Naming Conventions

^	Ground Center point	^01
_	Center point	_01
~	Attachment point	~01
~	Attachment point for guns	~*xx* Treat as normal object
0-99	Part Number	1
CA	Collision Box for Armory	CA01
CB	Generic Collision Box	CB01
CD	Collision for Doors	CD01
CE	Collision for Emplaced Weapons	CE01
CL	Collision for Ladder	CL01
LP	Light, Point	LP01
UPL	User Point Light	UPL01

Orientation

When creating your object in Max, you must make sure that the three axes are facing the same way as all the other objects in the game. If they are facing the wrong way, your sub-objects will move in the wrong directions when animated.

To see the true orientation of your object's axis in Max, set the **Reference Coordinate System** to **Local**. When properly oriented, each axis will represent the following movement:

- X = pitch
- Y = roll
- Z = Yaw

When using the **Top** view, the Z-axis should be pointing up in 3D space. Also, the Y-axis should always point towards the back of your object when in **Local**.

If everything has been set up properly, you should be able to set a range of movement in OED where a positive number will create a clockwise movement and a negative number will generate a counter-clockwise one.

Dimensions, File Types, Size Restrictions, and Required Fields

Standard Dimensions

The following list gives a guide to generic sizes for standard objects and orientation.

- **Doors:** 1.5 meters wide by 2.4 meters in high.
- **Windows:** 1.5 meters wide by 1.5 meters high. 1.0 meter from the floor to the sill.
- **Ceilings:** 3.75 meters in height.
- **Actionable Items:** 1.2 meters in height from floor (levers, doorknobs, switches, etc.)

File Types

All files known to be compatible with OED are listed below:

- **.TGA** - Targa File with 24-bit true color, or 32-bit true color with Alpha channel
- **.ASE** - Scene Export. ASCII from 3DStudioMax
- **.3DA** - This is the format in which OED saves files that it can edit.
- **.3DI -** This is the file that OED exports for use in the game. It does not include the textures.

File Size Restrictions

The total amount of texture memory allowed in the game is 60 megabytes.

Getting Started

Importing Files

When 3D Studio Max (3.1 or higher) creates an object, it exports it as an .ASE file. These files will have to be imported and saved in the .3DA format before they can be used by OED. Follow these steps to import and save your files so you may work on them:

1. Open OED from your `ModTools` directory.
2. Make sure the **1 Unit = 1 meter** box is checked. If you don't, your units will be in feet, not meters, and everything will be 1/3 its regular size
3. Select the resolution that you are going to import: High, Medium, Low or Tiny.
4. Open the **File** menu and click **Import 3DS ASCII**.
5. Find the filename that matches your selected resolution (**H, M, L, or T**) and click **Open**.
6. Repeat those steps for each resolution. Be sure to change the OED resolution setting for each.
7. Once you have all the different resolutions imported, go back to the **File** menu and choose **Save All**. This will save all of the resolutions into the proper .3DA format. Be sure that you save these files into the same folder as the original documents.

NOTE: All texture files used in the model must be in the same folder as the .3DA to open.

Once you import files for this object, you won't have to do it again, even if you alter the original .ASE file. When you re-open the .3DA files in OED, it will check them against the saved .ASE files. This is why they all need to be saved together in the same directory.

Opening Files

Once you've imported your files, or received files that are already imported, you can open them directly. This can be done individually or as a group of files.

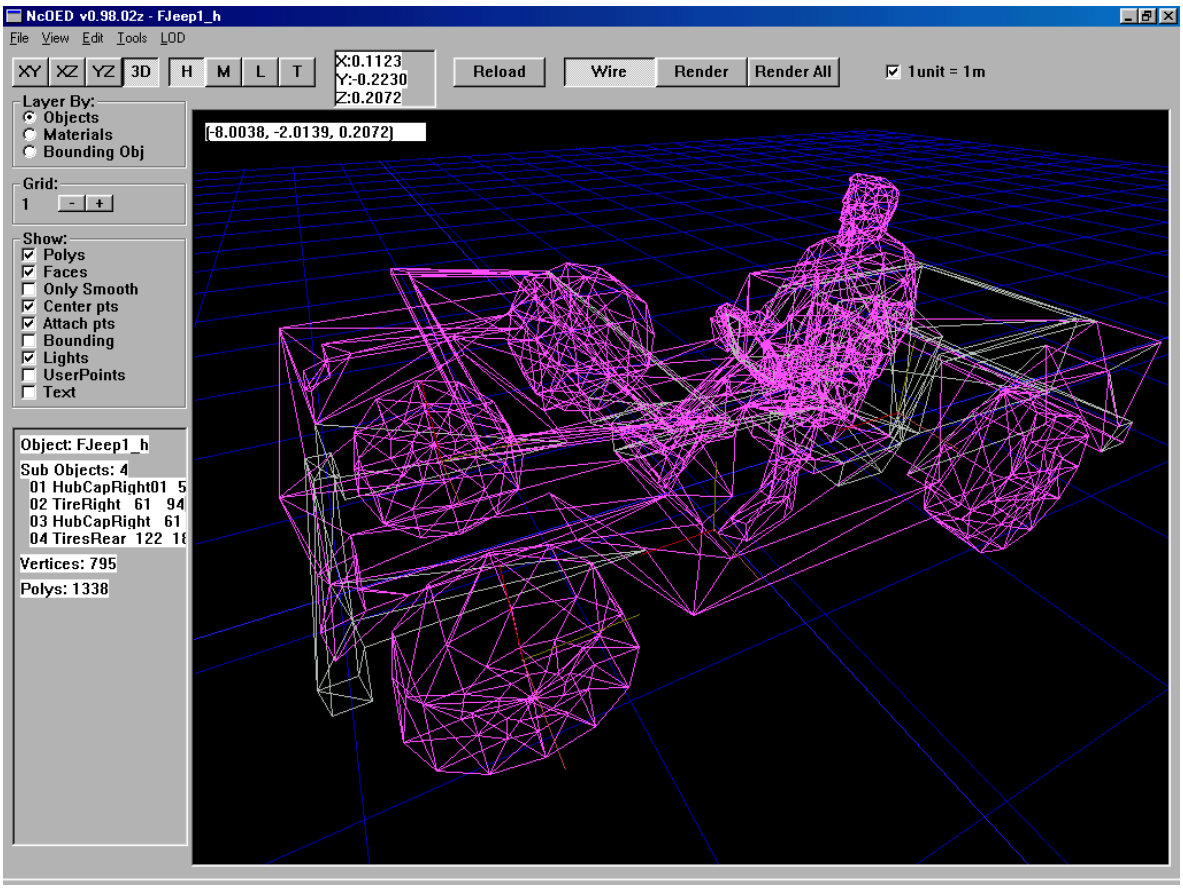
Choose **Open** or **Open All** to get started. If you click **Open All**, be sure that all of your resolution versions are in the same folder. Open one file. As long as the file names end with H, M, L, or T, the program will automatically open the High, Medium, Low and Tiny resolution files (not the husk, that's a separate entity) and you are ready to work.

If you open a new file, you may have to click **Reload** or another render button, to update the screen. Usually it is best to click the **Clear All** button before loading a new object.

Navigating the Tool

In any of the views, you are able to move your relative position around your object, move the object, or move the lighting around it by using key and mouse combinations.

- **SHIFT + left mouse** – Drags the object up/down, left/right relative to your current view. In XY view, it moves the object only in the X and Y directions. In XZ, it moves it only in those directions. This change does not transfer to the game and is used just to help orient the object into the proper view.
- **SHIFT + right mouse** – Zooms your view in and out.
- **CTRL + right mouse** – (3D View Only) Moves your view around the object in 3D space.



The Main Screen

Buttons and options

XY, XZ, YZ, 3D – These buttons choose your viewpoint in the editor. XY looks straight down on the object, XZ views the side, YZ faces the front of the object and 3D gives you an isometric view that you can move around using the SHIFT key and your right mouse button.

H, M, L, and T – These Buttons choose which resolution model you are currently editing: High, Medium, Low, and Tiny. Some objects may not have a tiny version. Each resolution gets its own settings.

Coordinates box - The X, Y, and Z refer to the object's location in 3D space from the grid's center point. You can click on this box to open up a new window where you may enter new values. Since these numbers are not exported, this function is most often used to adjust your view of the object.

Reload – Clicking this button will revert back to the most recently saved version of the object. It won't ask for a confirmation, so be sure you want to do this. Any unsaved changes will be lost.

Wire – Brings up a wire frame view of your current object.

Render – Shows you a fully rendered version of your currently viewed resolution of your model. When you zoom in or out, you will only see this model.

Render All – Renders all resolution versions of your object. If you have set a threshold on polygon size (described later), the objects will switch out when zoomed to the preset limits. This means that your High resolution version will become a Medium resolution version when zoomed out enough.

1 unit = 1m – Keep this box checked, especially before opening a new item. It standardizes your values for this object to 1 unit equaling 1 meter.

Layer By – These selections are used in conjunction with the **Layer** selection found in the **Edit** menu or by pressing CTRL + L. Opening the **Layer** window will show a list of all the elements that match the type chosen in the **Layer By** window. Check or uncheck to hide or show specific elements.

Grid – Changes the grid size in the viewer bigger or smaller.

Show – Use the following buttons to show specific types of objects while in the wire frame mode:

> **Polys** – Displays the flat polygons that make up an object.
> **Faces –** Draws a diagonal line across each polygon to give a better representation of where the polygons are. Without them, it gets easy to confuse objects like a closed crate with an open one.
> **Only Smooth –** When an object is displayed in the wire frame view, this setting causes only triangles that have smoothing set to be visible.
> **Center pts –** Shows the artist specified center point of each object. The center point is the point from which the X, Y, Z coordinates are measured.
> **Attach pts –** Shows the pixel where two objects are joined, such as the body of a jeep and a tire.
> **Bounding –** Displays the geometric collision box that surrounds the object.

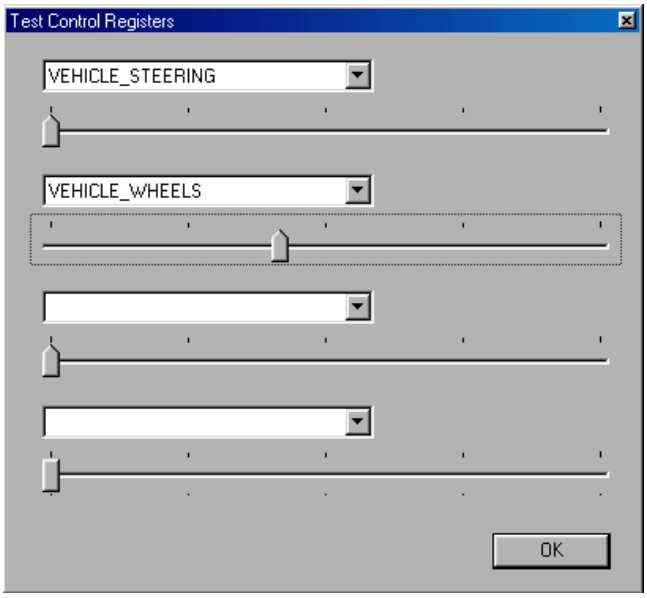
Object Info Window - This window shows you the current object's name as well as its sub-objects, vertices and polygon count.

Screen Functions

There are a number of functions that are accessible by right clicking anywhere on the main screen and selecting the appropriate selection from the window that opens. Most of the functions are accessible only when you have entered **Render Mode** and are not seen when you are in **Wire Frame** mode. Most of these functions occupy full sections of this manual. The rest are described here:

Layers - Use the **Layers** function to hide parts of the object so you can concentrate on a single part.

Control Camera / Light / Object - When you enter render mode by clicking on the **Render** or **Render All** buttons, you are able to choose to move the camera or object by right clicking anywhere on the screen and selecting the appropriate choice in the pop up window. Then by pressing CTRL + right mouse button, you can move the selected object. Currently the ability to move the light is not usable.



Limit Render to 1 pass / 2 pass – If an object has two effects (such as bump mapping and a chromed effect) they are rendered in separate passes. By selecting one or the other of these options, you can view the first or second effect on its own.

Test Object Destroy – Used to test buildings that have a husk and an animated destruction sequence.

Test Control Registers – Use the pull down menus to select your pre-configured control registers. Use the sliders to watch the animations attached to those registers. Since the sliders only track from 0 to 1, you may only be able to move in one direction even if the object is properly configured to move forwards and back.

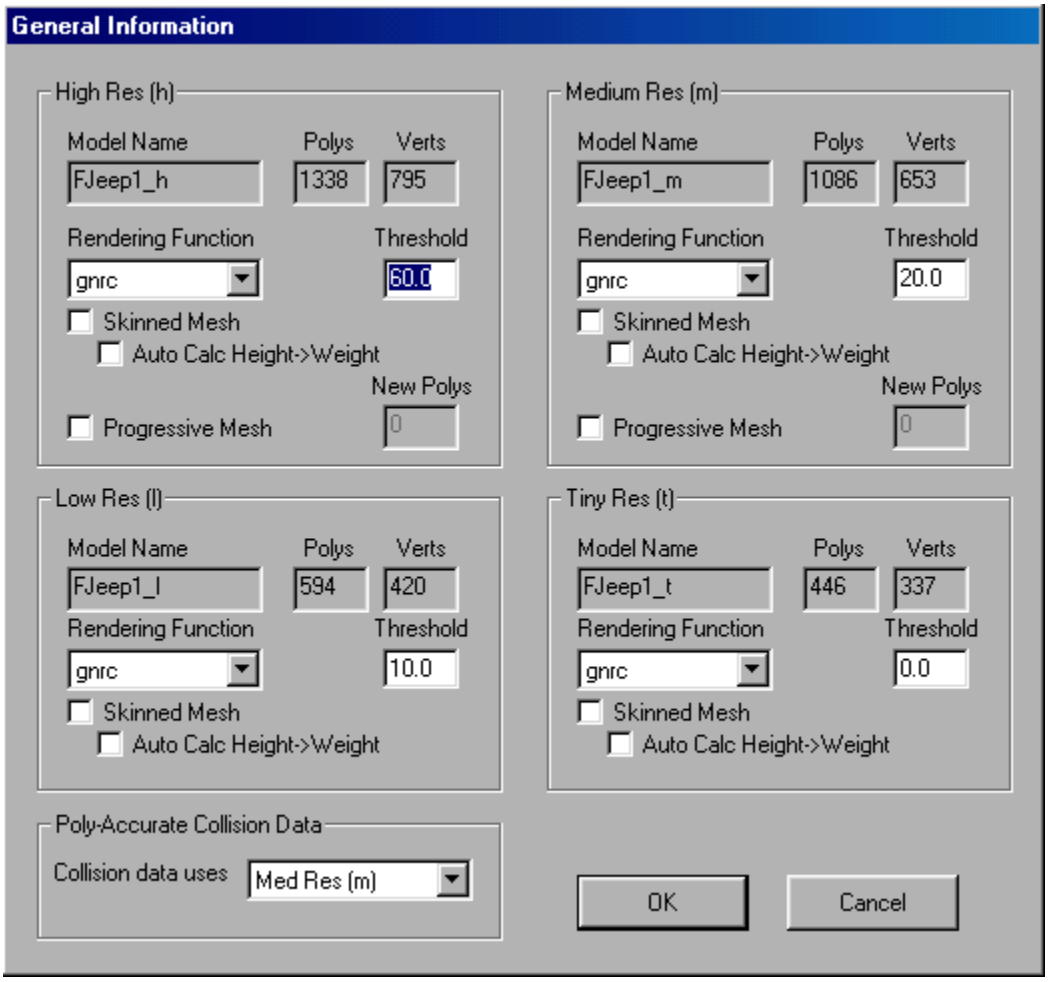
Jungle / Desert / Snow – This feature is no longer used.

Save PCX – This feature is no longer used.

General Information Screen

An object is made up of various resolution models (High, Medium, Low, and Tiny) that are viewed when the object reaches different distances from the viewer. By opening up the **General Information** screen you can edit information for each resolution.

In the general information screen, you can give specific characteristics to each resolution and determine the distance at which it switches to the next res. Not every model has every resolution. Some may have only one and others may have all but the tiny resolution.



Model Name – This box contains the name of the file for the current version of the model (High-Tiny).

Polys – Shows the number of polygons that make up this resolution of the object.

Verts – Shows the number of Vertices (a point defined by the intersection of two lines of a polygon) that make up this object.

Rendering Function – Render functions identify specific objects to the game engine so they act in a specific, predetermined, manner. These settings are have been almost entirely replaced by Part Animations and additional settings found in the **Edit Materials** screen. Certain objects still use these functions, however. They are listed below:

- **bldg –** Buildings.
- **cmch –** Comanche. Assigned to the player and A.I. Comanche helicopters.
- **gnrc –** Generic. Used for almost everything in the game.
- **org0 –** Organics. Used for any person or animated being.
- **tgrp –** Tree Groups. Multiple trees in one object.
- **tree –** Tree. Check this to specify a single tree or foliage item.

Threshold – The number entered here will determine when the game switches from one resolution to another. As the object moves away from the player, it shrinks in size. The number of polygons being displayed also shrinks. When the number of polygons reaches the threshold number, it switches to the next resolution on the list. The last resolution should always have a threshold of 0.

Skinned Mesh – This box gets checked whenever you have a model designed to receive animation information. This includes trees that sway in the wind and people. If this box is checked for one model, you must check it for all the models. Failure to do so can result in a game crash when the engine attempts to change to a lower resolution.

> **Auto Calc Height _weight –** Used for single trees only, this function give a lighter weight to the upper parts of trees to allow them to sway in the wind. The weight is based off the distance from a specified vertex. This function should only be checked for the high resolution model (the others are too small to see swaying anyway).

Progressive Mesh – This feature is no longer used.

New Polys – This feature is no longer used

Poly Accurate Collision Data – When calculating the impact of bullets and missiles, the computer uses a much more accurate collision box based on the true outline of the model. You can determine which resolution outline is used by selecting it in this box. Typically Medium is used since its outline is almost identical to the High resolution, but takes up less memory.

Whichever resolution you set for the collision check is the resolution that gets flagged as being a specific material such as stone or cloth. This is explained in more detail in the Materials section.

Materials and Textures

An understanding of Materials and Textures is vital in the use of The Object Editor and the creation of visually compelling 3D objects. There is some confusion as to what constitutes a Material and what constitutes a texture. The following should help to dispel some of the related myths.

Material

A material completely describes how a set of polygons is rendered. Material attributes include:

- Color information (either solid-color or derived from 1 or 2 texture maps)
- Alpha information (either solid-alpha or derived from 1 or 2 texture maps)
- Lighting characteristics
- Animated shading
- Animated texture coordinates
- Physical attributes
- Advanced features (including reflective surfaces and light projection)

Texture Stage

A texture stage (i.e. texture) is a single component of a material, comprising of RGBA bitmap data. A material may use no texturing at all, or it may use one or more texture stages, each created from 1 or 2 bitmap files.

A texture stage contains the following information:

- 1 or 2 bitmap files that are combined to create an RGBA image 'texture'
- U & V Animation data (for sliding, skewing, scaling, & rotating the texture coordinates)
- UV Clamp flag (to prevent tiling of this texture stage)
- Note that the texture stages are a subset of the material.

Each texture stage contains a full RGBA bitmap derived from one or two bitmap files, as follows:

Color Filename	***Alpha Filename***	***Texture stage contains***	***RGB***	***Alpha***
Colorfile.pcx	-	RGB data, from single PCX file	From PCX	255
Colorfile.pcx	Alphafile.pcx	RGBA data, merged from 2 PCX files	From PCX	From PCX
-	Alphafile.pcx	Alpha data only, from single PCX file	255,255,255	From PCX
Image24.tga (24-bit)	-	RGB data, from single TGA file	From TGA	255
Image32.tga (32-bit)	-	RGBA data, from single TGA file	From TGA	From TGA

As you can see, when there is no alpha component specified from a file, the alpha is set to 255. When there is no color component specified, the color is set to 255,255,255.

Because each Texture Stage contains only one final RGBA image, if you specify a separate color and alpha PCX file, the two files must be of the exact same dimensions. They are merged together into one RGBA bitmap that cannot be done with differing dimensions. No exceptions!

Image files (.PCX, .TGA) must be sized according to any of the following dimensions (measured in pixels and x, y coordinates).

8 x 8	8 x 16	8 x 32	8 x 64	-	-
16 x 8	16 x 16	16 x 32	16 x 64	16 x 128	-
32 x 8	32 x 16	32 x 32	32 x 64	32 x 128	32 x 256
64 x 8	64 x 16	64 x 32	64 x 64	64 x 128	64 x 256
-	128 x 16	128 x 32	128 x 64	128 x 128	128 x 256
-	-	256 x 32	256 x 64	256 x 128	256 x 256

* All dimensions are in pixels and x, y coordinates.

Texture creation

Textures should be created to look normal as if it were being viewed at noon on a nice sunny day. The game engine initially lights everything as if it were a sunny day. Where appropriate, the game engine will darken the textures. Don't make dark textures to compensate for the bright game engine lighting.

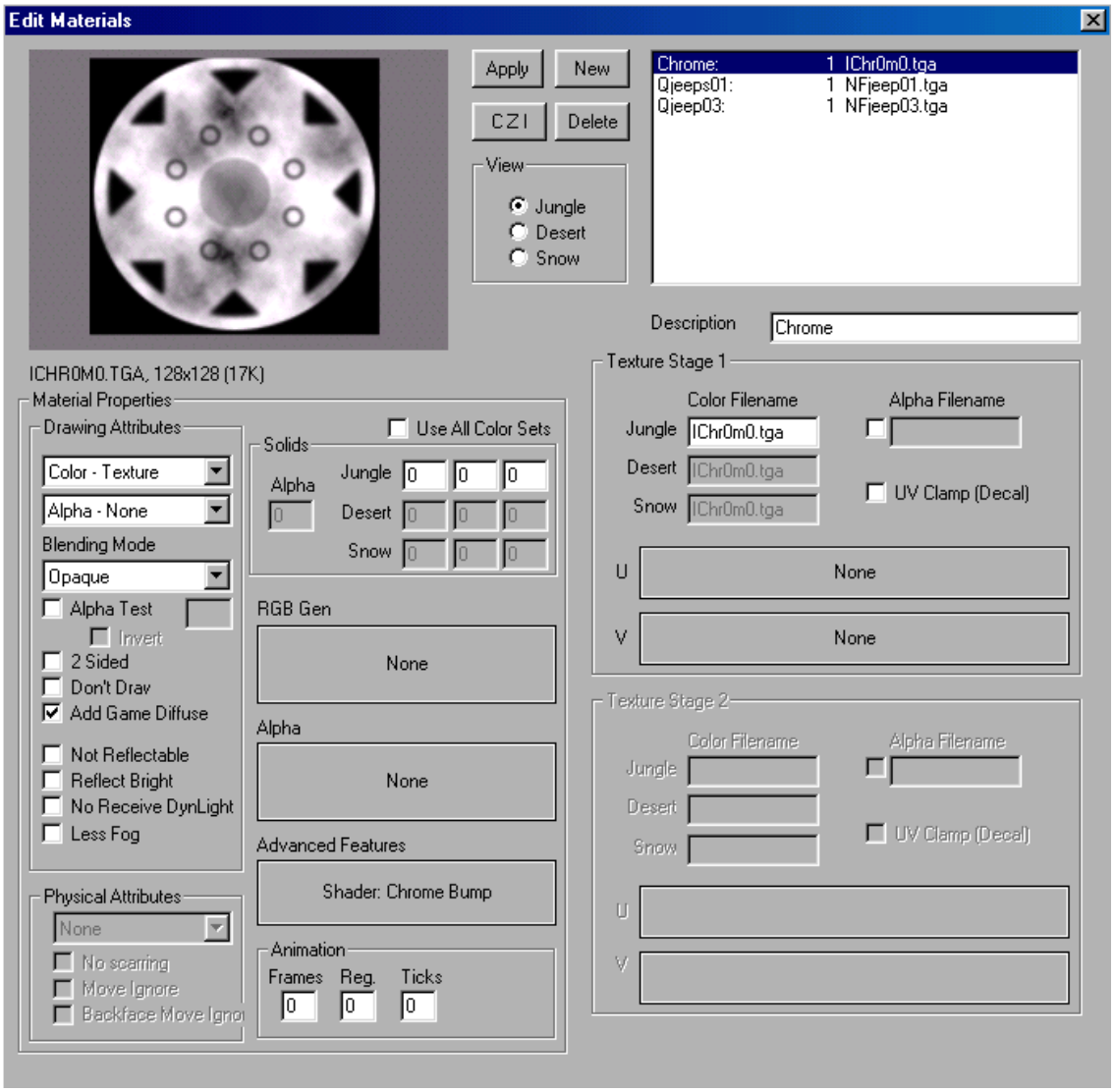
Mission designers are able to darken the lighting to produce dusk, dawn, midnight, or other lower lighting effects. If the textures are darkened to fit the mood, then everything will be too dark.

Remember, even though a mission may occur at night, the player may be carrying a torch or flashlight, and if the torch is brought close up to the wall of a building, the engine needs to render a very bright patch of the wall. This is only possible if the wall texture is bright. The game engine can infinitely darken geometry, but it can't make it brighter, unless the textures are bright to begin with.

Edit Materials Screen

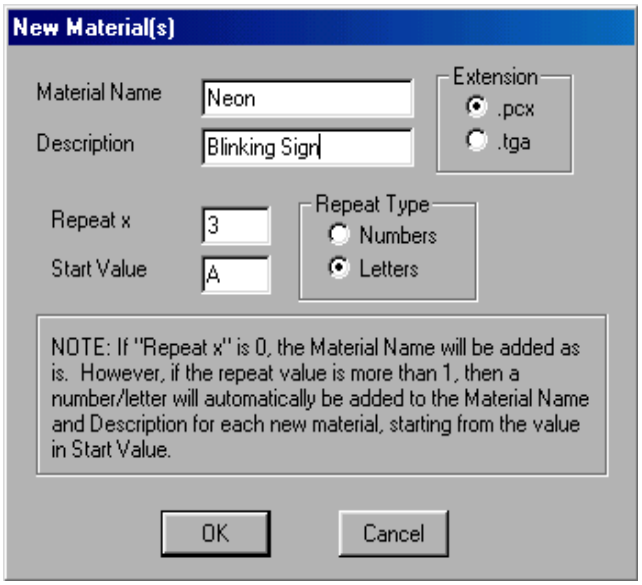
Each texture is independently adjusted. If two textures are to share attributes, you will have to do each one separately. To apply your changes just close the window.

Save your work before experimenting. There is no way to make changes and not have them applied when you close the window.



Each of the materials functions is listed below.

Apply – Press this to apply your changes to the selected object. If possible, the changes will appear in the preview window.



New - Press this to add another texture map to the object. This is most often used to add additional files for an animated texture. Once the button is pressed, you will see this window open. Simply enter the name of the file you want to use and click **OK.** If you want to cycle through textures for animation, they must be numbered (or lettered) in order.

If you have created a number of files, you can automatically add multiple names by putting the number of files into the **Repeat** field. Every file will retain the same material settings (a time saver!). You can test this feature by entering any name and using the repeat setting. Just delete the files, or don't save the object when you close.

CZI – Not Used.

Delete – Click this button to delete the selected texture map

Description – You may add a comment into this line. It does not affect the model.

Material Properties - Drawing Attributes

The **Drawing Attributes** section describes how the polygons are to be drawn by the hardware.

Materials contain color and alpha information for the rendering of polygons. The color component can come from a solid color, a texture map, or two texture maps modulated together. The alpha component can come from solid alpha value, a texture map or two texture maps modulated.

Color –

> **None** – The texture is not used and nothing will show up.
> **Solid –** This setting gives the texture one solid color. The color is set by altering the Texture File RGB settings. (Note: You may have to click and unclick the **Use All Color Sets** to get the color to load.
> **Texture –** Select this to use the texture map in its full form.

Alpha – Affects the Alpha (transparent) sections of the object.

> **None -** If there are no Alpha sections, check this box.
> **Solid -**
> **Texture –** Select this setting to use the object's alpha textures as intended. When selected, you will use the settings in the **Blending Mode** to adjust it.

Blending Mode – This mode is only used when the **Alpha** is set to **Texture**.

Blending Mode

The **Blending Mode** setting decides how to blend the color fragment from the material with the pixels in the frame buffer. The following blend modes are supported:

	Function	Comments
Opaque	$DEST_{RGB} = SRC_{RGB}$	Copies pixels to the frame buffer and appears the same as in the OED. The color is drawn over the background and is used for all solid materials. Does not use Alpha and is drawn first with the Z buffer.
Alpha Blend	$DEST_{RGB} = SRC_{RGB} * SRC_{A} + DEST_{RGB} * (1.0-SRC_{A})$	Blends the pixel with the frame buffer, using the alpha of the pixel to determine the blending ratio. The color is mixed with the background (Smoke, anti-aliased cutouts, etc.) Alpha Blended objects sometimes have issues with sorting that sometimes allow objects in the background show through foreground ones. Uses Alpha, drawn last and sorted.
Premultiplied AB	$DEST_{RGB} = SRC_{RGB} + DEST_{RGB} * SRC_{A}$	Multiplies the frame buffer contents with the alpha of the pixel, and add the pixel color without modulation. The background is darkened with Alpha and color is added. This is like Alpha Blend and Additive in one texture. Uses Alpha, drawn last and sorted.
Additive	$DEST_{RGB} = SRC_{RGB} + DEST_{RGB}$	Adds the pixel color to the frame buffer contents. An additive object will lighten another object if laid on top. One example is muzzle flashes that lighten the gun texture when triggered. As it fades out it becomes transparent. Does not use Alpha, drawn last, sorted.
Multiplicative	$DEST_{RGB} = SRC_{RGB} * DEST_{RGB} * 2$	Multiply the contents of the frame buffer with the pixel color. Color filters the background (Stained Glass) Does not use Alpha, drawn last, sorted

* $DEST_{RGB}$ denotes the destination of the pixel – i.e. the frame buffer
* SRC_{RGB} denotes the source of the pixel color – derived from the material (texture and/or solid color)
* SRC_{A} denotes the source of the pixel alpha – derived from the material (texture and/or solid alpha)
** Note that the above formulas assume that an **R**, **G**, **B**, or **A** value of 1-255 is actually a value of **0.0** – **1.0**

Alpha Test – Used to cut out texture based on interpolated ALPHA for smooth edged textures. It is best used with Opaque Blend Mode, but can be used with any mode. Alpha Test drops pixels and does not anti-alias. This may look bad at a distance and probably should be avoided in medium and low-res geometry. Alpha Test does not affect draw order, sorting, or Z buffer use.

2 Sided – Check this box if the front and back sides of the object will be seen. If you leave this unchecked, one side will have texture and the other will be transparent. Checking this box will increase the memory requirements of the object.

Don't Draw – Don't use. This function does not work properly.

Add Game Diffuse – Check this box to use the in-game lighting. If switched off, the object will use only its own vertex shading.

Not Reflectable – Makes the object not reflect in mirrored surfaces.

Reflect Bright – Makes the object reflect only bright light sources such as the sun or an explosion.

No Receive DynLight – Check this to disable the effects of dynamic lighting on the object. Usually used on Alpha Blend materials.

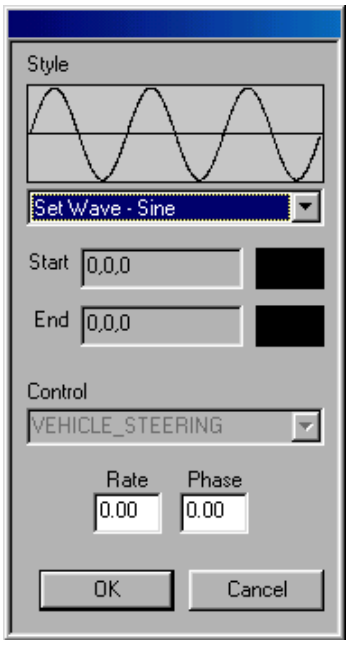
Less Fog – Reduces the amount of fogging that will cover an object in the game.

Physical Attributes – These settings determine the material of your object and, therefore, its protective qualities. The setting will also determine if sparks or other effects are displayed when hit by weapon fire.

- **No Scarring –** Check this to prevent any markings from appearing when the object is hit with rocket or gunfire.
- **Move Ignore –** Bullets will pass through this object without making a collision check.
- **Backface Move Ignore –** Bullets will pass through in one direction, but not the other.

Use All Color Sets - Not used.

Solids – When your **Drawing Attribute** is set to **Color - texture**, it will draw its values from this area.

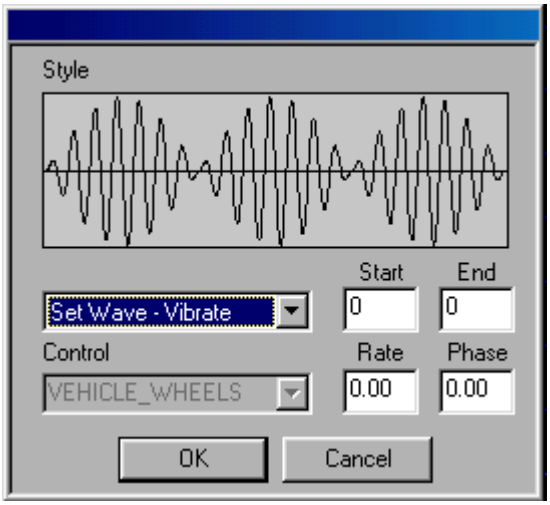


- **Alpha –** Input a number from 1 to 255 to set the transparency of your solid texture.
- **Jungle/ Desert/ Snow –** Only **Jungle** is valid.
- **RGB Gen –** Clicking in this square will open up a new window. Using the RGB settings you can create a self-luminous object. This self-luminance setting now allows you to set a full RGB color, and also set a wave function between 2 colors.

  Use the function sets to generate a constant self-luminous color. Note that RGB 128,128,128 is effectively non-lit, and RGB 255,255,255 is double-lit (because of 2X lighting).

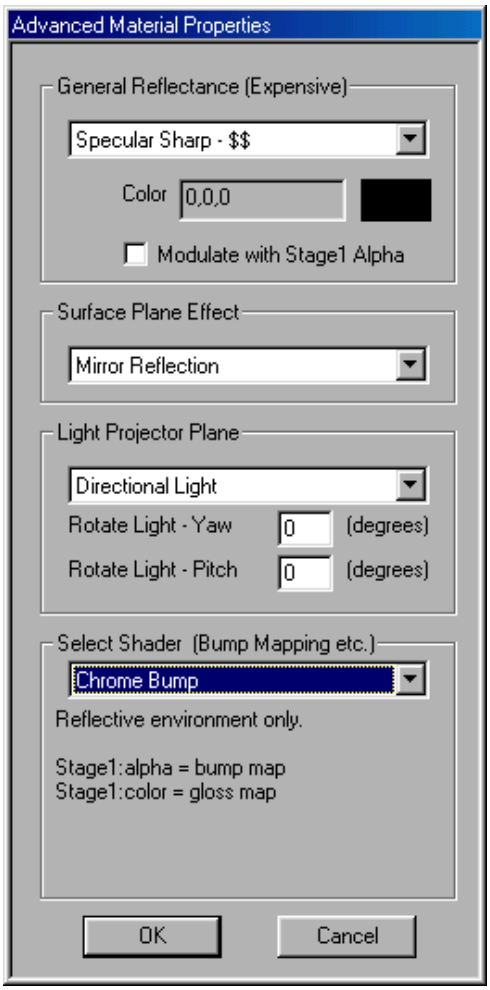
  Make sure that you turn off **add game diffuse** otherwise you will get additional luminance from the game lighting.

Alpha – Use these settings to make special transparency effects. Use the **Style** window settings to make objects fade in and out at various intervals or create a number of different fade patterns.



Advanced Features –

- **General Reflectance –** This determines the amount of light reflect by an object and the color. The settings are **Specular Sharp $$**, **Set Environment Sharp $$$**, and **Add Environment Sharp $$$$**. The dollar signs indicate how much their use will affect memory and framerate.
- **Modulate with Stage 1 Alpha –** When selected, the reflectance will be applied only to the Stage 1 Alpha texture. White areas will have full reflectance and black areas will have none.
- **Surface Plane Effect –** Gives the object a mirror surface.
- **Light Projector Plane –** Not often used, but when it is, you can choose special lighting effects from this pull down menu. The object must be previously set up in Max to be enabled.
- **Select Shader –** Choose your shader here.



Animation

Use these windows to adjust your frame rate when animating an object by switching between texture files.

- **Frames -** This will be the same as the number of files that make up the total animation.
- **Ticks –** The amount of time between frames. One tick is equal to 1/60 second.
- **Reg** – This feature is no longer used.

Texture Stage 1 & 2

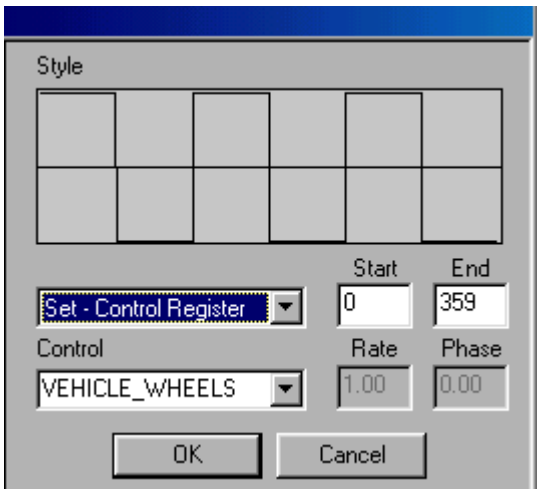
The Texture box allows you to define which bitmaps are used in each texture stage. Enter the appropriate file into the **Jungle** field. The **Desert** and **Snow** fields are not used. Separate objects are created for those terrains instead of swapping out textures.

When you have a RGB multiplied texture created in 3D Studio Max, the multiplied layer is shown in the stage 2 area.

- **Color Filename –** The Color Filename is imported from the 'diffuse' texture field in the Materials Editor of 3D Studio Max. (RGB - Multiply). The **Desert** and **Snow** options are not used.
- **Alpha Filename** -The Alpha Filename is imported from the 'opacity' texture field in the Materials Editor of 3D Studio Max.
- **UV Clamp (Decal) –** Prevents the texture from tiling.

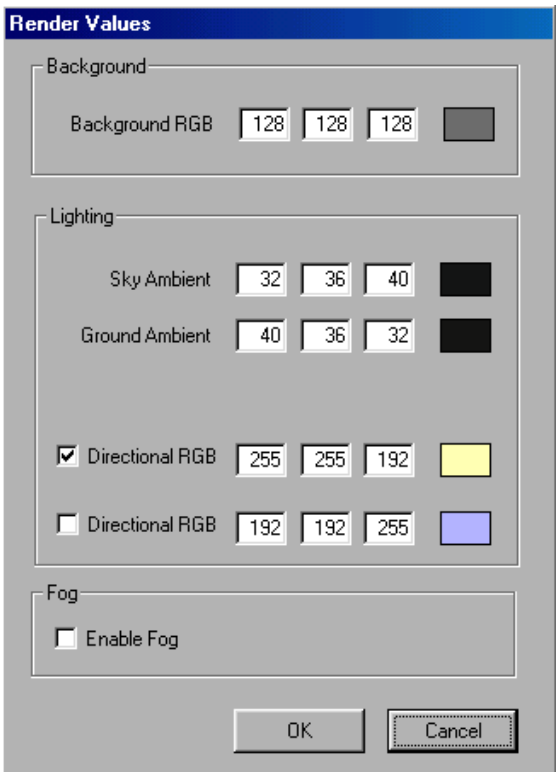
U
Enables texture animation along the U, or left to right, axis. Utilize various styles and values from 0 to 359 to get different results.

V
Enables texture animation along the V, or top to bottom axis. Utilize various styles and values from 0 to 359 to get different results.



Render Values

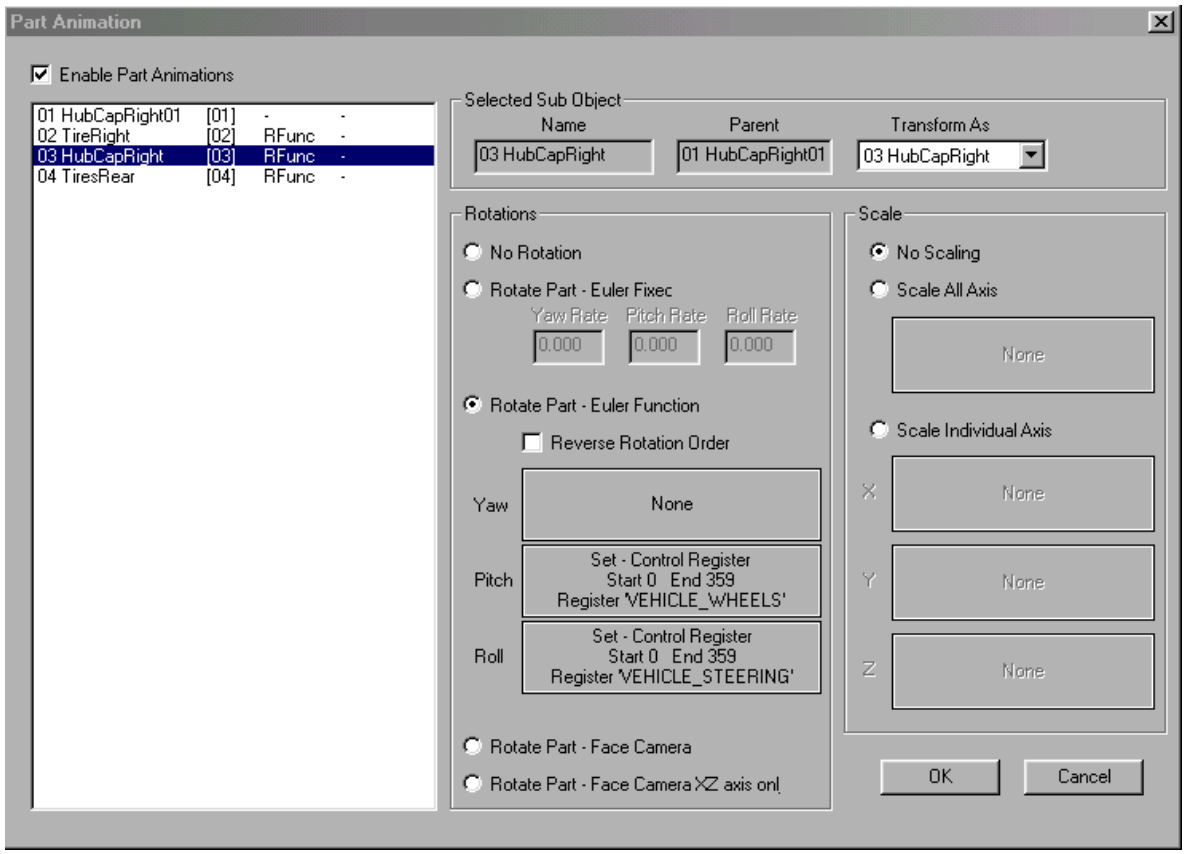
Render Values are temporary adjustments to lighting effects. Using this feature will allow you to view the object in different lighting situations and against different background colors. These settings do not carry over into the game.



Part Animations and Control Registers

There are two ways to make a part move. The first is through control registers. The other is by simply assigning it a **Set wave _** on a certain axis. Control registers are set if you want a part to move in regards to in game occurrences. A good example of this is the wheels of a vehicle. When the game tells that vehicle to stop, it will also stop the rotation of the tires.

If you want constantly animated objects, such as a flashing texture on a computer screen, you can assign the appropriate “set_wave”. This movement will be continuous.



Control Registers

A control register is a text-based token that identifies a dynamically changing value set by the game engine. In easier terms, it's a way to make sub-objects move, rotate, scale, or slide. The current value of the register can be interpreted by a 3DI (during rendering) to affect texture UV animation, color cycling, and sub-object animation. The value of any control register is specified as a position anywhere between the **Start** and **End** values.

In order to use a control register in a 3DI model, you must first determine how you want the 3DI to respond to the values in the control register. Do you want to animate a sub-object? Or do you want to animate the UV mapping of a texture? Or do you want to cycle a solid color RGB?

Depending on the above answers, you will need to adjust settings in either the **Edit Materials** dialog or the **Part Animation** dialog in OED.

Note that all **Start** and **End** positions can be set from 0 to 359.

Rotating a sub-object based on a control register:

- Ensure that the **Enable Part Animations** checkbox is set
- Select the appropriate sub-object to apply the animation
- In the **Rotations** box, choose **Rotate Part - Euler Function**
- Click on either **Yaw**, **Pitch**, or **Roll**
- Set the function type (first combo box) to **Set - Control Register**
- The **Control** combo box will become enabled, and you can type a control register name into it (or select from existing default names)
- Set the values in the **Start** and **End** edit boxes to reasonable start and end angular positions. Note that as the control register dynamically changes, the current rotation will move between your specified **Start** and **End** values
- Click **OK**
- Choose another rotation to set control registers (optional)
- Set a scale function on the sub-object (optional)

Scaling a sub-object based on a control register:
You use the **Part Animation** dialog in OED. Here is the process

- Ensure that the **Enable Part Animation** checkbox is set
- Select the appropriate sub-object to apply the animation
- In the **Scale** box, choose either **Scale Individual Axis** or **Scale All Axis**
- If you selected individual axis, click on either **X**, **Y**, or **Z**. Otherwise click on the single box below

- Set the function type (first combo box) to **Set - Control Register**
- The **Control** combo box will become enabled, and you can type a control register name into it (or select from existing default names)
- Set the values in the **Start** and **End** edit boxes to reasonable start and end scale multipliers. Note that as the control register dynamically changes, the current scale will move between your specified **Start** and **End** values. A scale of 1.0 means no change, a scale of 0.5 means half-size, and a scale of 2.0 means double size. A scale of 0 is also valid and means scale down to (effectively) nothing
- Click **OK**
- Choose another axis to set control registers (optional)
- Set a rotation function on the sub-object (optional)

Moving or sliding a sub-object based on a control register:

You need to place an additional blank sub-object into your model in between the parent of the moving sub-object and the sub-object itself. You can then use the **Part Animation** dialog in OED.

Here is the process:

- Ensure that the **Enable Part Animation** checkbox is set
- Select the blank sub-object to apply the animation
- In the **Scale** box, choose either **Scale Individual Axis** or **Scale All Axis**
- If you selected individual axis, click on either **X, Y**, or **Z**. Otherwise click on the single box below
- Set the function type (first combo box) to **Set - Control Register**
- The **Control** combo box will become enabled, and you can type a control register name into it (or select from existing default names)
- Set the values in the **Start** and **End** edit boxes to reasonable start and end scale multipliers. Note that as the control register dynamically changes, the current scale will move between your specified **Start** and **End** values. A scale of 1.0 means no change, a scale of 0.5 means half-size, and a scale of 2.0 means double size. A scale of 0 is also valid and means scale down to (effectively) nothing
- Click **OK**

Control Register FAQ's

How do I know whether a particular control register is to be used for scaling or for rotation?
That is completely up to you. Control registers do not have a specific purpose other than whatever you wish to assign them to. If you want a control register to rotate something, then go ahead and use it for that. And then if you want to use the same register to scale a different sub-object (or the same sub-object) then go ahead and use it for that too. And if you want that very same control register to animate the UV mapping on a texture, then go ahead and use it there as well. There is no limit other than your imagination.

How do I know whether a specific control register is to be used for yawing a sub-object, pitching it, or rolling it?
Again, that is up to you. If a control register has a name such as "vehicle_steering" then you apply common sense and make sure that it turns the wheels left and right as if the vehicle is steering left or right. There is no specific technical requirement in the game engine that predetermines the use of a control register, other than for AI purposes, where we want to visually match what the AI is doing.

What range should I use (Start & End) when I set up my control-register rotations?
In the case of AI controlled turrets, wheels etc. see below. Otherwise, just use a range that looks good. There are no hard and fast rules.

How do I set up control-register rotations for an AI object that needs to aim at something or point in certain directions?
This is the single exception where you have to use specific values when interpreting a control register. Generally speaking, the AI needs to control the precise angle at which an object is oriented when it comes to targeting or pointing. The game engine will set the appropriate control register to a value from "start" to "end" that relates to an angle between 0-359 degrees. You have to set the angle range in the part animation to go from 0-359 in order to match this.

How do control registers actually work?
Okay, let's get technical … Internally, the game engine treats control registers as numeric variables that range from 0.0 - 1.0.

A value of 0.0 represents "start" in OED.

A value of 1.0 represents "end" in OED.

A value of 0.5 represents halfway between "start" and "end".

All values between 0 and 1 are linearly interpolated between "start" and "end".

There is a predetermined master-list of control register names built into the game engine.

When a 3DI model is loaded, the control registers it uses are matched (by name) to the control registers in the master list. If the name does not match any of the predetermined registers, then the control register is ignored.

When an object is about to be rendered, the game engine calls a rendering function (which knows what type of object it is), and will set relevant control registers to values in the above range (from 0.0 - 1.0).

During rendering, the control register value is converted into the range specified in OED (start to end) and the resulting value is used as the rotation/scale component.

For example with these settings:

Part animation - rotation on Yaw axis

control register = HELO_GEAR

Start = 100

End = 200

when HELO_GEAR = 0.0, yaw will be 100 degrees

when HELO_GEAR = 0.5, yaw will be 150 degrees

when HELO_GEAR = 1.0, yaw will be 200 degrees

The game AI it has no knowledge of how you interpret the control register values in the 3DI image. It only knows that when the gear is up, the control register needs to be set to 0.0, and when the gear is down, the control register needs to be set to 1.0. When we are moving the gear from up to down, we just linearly interpolate between 0 and 1.

When the game engine sets the control register to a value between 0 and 1, and the 3DI rendering will take care of the rest.

The combination of the control registers with your 3DI image gives you a powerful tool. The game engine can set the values in the control registers and the 3DI has very flexible animation options. All of this can be performed without changing any game code. If you wanted to make the gear open more quickly, you would have to change the control registers values, but the range of movement would be completely specified in the art, making that element completely data driven.

Turret and Barrel combination

When you have a turret & barrel set up on a 3DI model, the rotations you require are yaw for the turret, and yaw + pitch for the barrel.

OED normally rotates a sub-object in the order (1) yaw (2) pitch (3) roll, but in the case of the barrel, you need to rotate the pitch first, and the yaw second. Make sure to click the **Reverse Rotation Order** checkbox in the **Part Animation** dialog of OED to get the correct rotation order. This can also be used in other situations where the rotations don't seem 'right'.

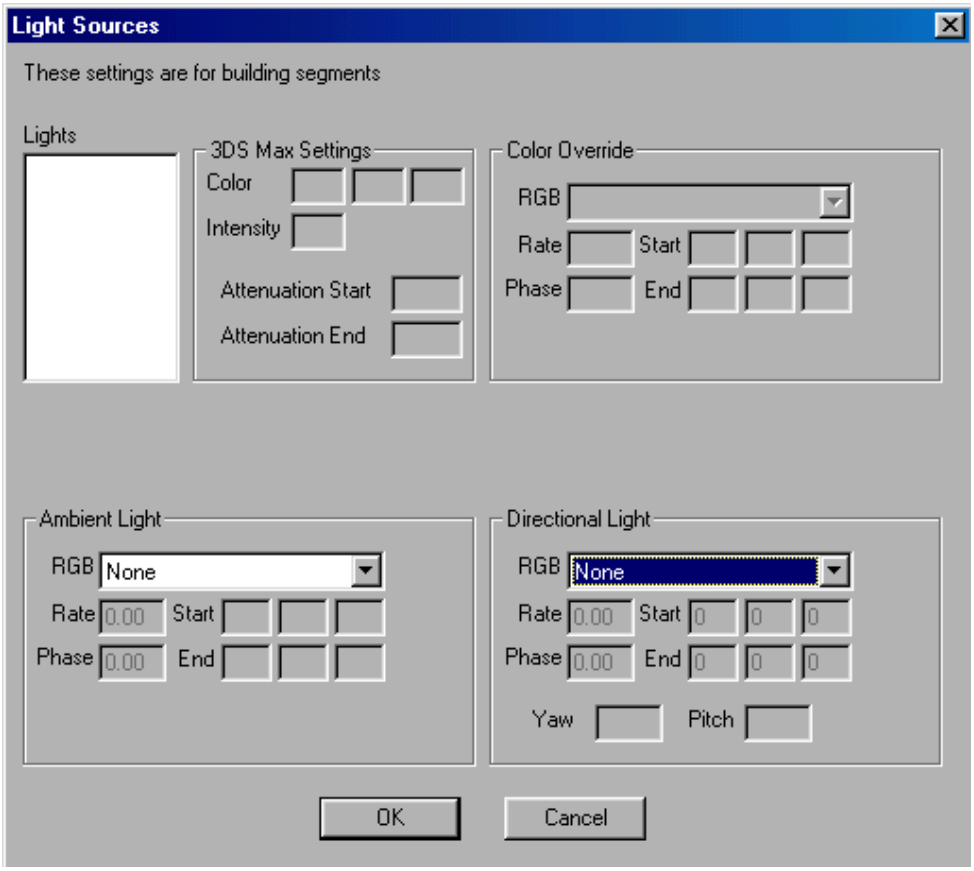
Light Sources

When lighting an interior area, you must use your vertex lighting to illuminate the room itself. To light objects inside the room such as players, enemies, etc, you will need to use 3D Studio Max Omnilight in conjunction with the OED's Light Sources.

To use a light source, you must first have placed one to four Omnilights inside the object when building it in 3D Studio Max. The files must be specified as LP01, LP02, etc. up to a (recommended) total of four. Only the intensity, color and far attenuation will translate to the OED, however.

Ambient Light – If an object is not being lit by an Omnilight, the ambient light will light it. You will want to make sure that these settings match your vertex variables.

Directional Light – This setting is not used in the OED.



Optimizations

Lights

Limiting the total number of light sources affecting any one object in the game will improve the speed of the rendering. When the player passes through an interior, using a minimal number of lights for all visible objects will increase the efficiency of the rendering. This is why each building piece can only own up to four light sources. Also, the falloff radii of the lights in each room segment should ideally not intersect one another.

The game engine is set up by default to use either vertex colors or game diffuse lighting. Using both together will cause slower rendering.

For building interiors, use vertex colors only, and no game diffuse (where possible).

For exteriors, and all other objects, avoid using vertex colors, and use game diffuse only.

General Streamlining

There are four main factors contributing to the overall speed of the rendering engine. These are in order of importance:

Total number of strips per object

Total number of vertices per object

Total number of sub-objects per object

Number of object s rendered per frame (i.e. per scene)

Note that these factors become increasingly significant in a fully laid-out mission; you may see less difference in OED where only one object is being rendered per frame.

Strips

To determine how many strips are needed to render an object, first divide your object based on sub-objects, then divide each sub-object based on materials used. You now have the number of strips needed to render the object.

If a tree has 1 sub-object that uses 2 materials, then it needs 2 strips.
If the tree has 2 sub-objects, where one sub-object (01) uses one material and the other sub-object (02) uses another material, then it also uses 2 strips.
If the tree has 2 sub-objects and one sub-object uses one material, while the other sub-object uses 2 materials, then it uses 3 strips.
If the tree has 2 sub-objects and both sub-objects use both materials, then the Object uses 4 strips.

Polygons and Vertex Count

Polygons that share the same material and mapping coordinates also share vertices. If an object is mapped with different mapping coordinates on alternate polygons, all vertices that occupy a point where the mapping is separated, will be counted once for every change in mapping.

An example of this on a cube would be an instance where the cube is mapped planar, or using UVW Unwrap, then the total number of vertices for the cube is 8. If the same cube were mapped with Box mapping, or each side mapped planar, then the number of vertices for that same cube is 24, counting each vertex 3 times since there are 3 mapping coordinates assigned at every vertex. By mapping the cube with UVW Unwrap, and moving the vertices without breaking them apart in the editor, the polygons retain their shared vertices and material, keeping the object in one strip.

Combining textures that decal map onto one texture per object will increase rendering efficiency by sharing the same texture file even though the mapping coordinates may increase the vertex count.

An example of an object that may occupy one or two strips based on different mapping is a tree. If a tree is a single 3DI object that has two parts with different mapping coordinates that use different materials, the tree will occupy two strips (one 3DI object x two materials). If the same tree were mapped with different mapping coordinates but shared the same texture for both parts, then it will only occupy one strip. Note that both of these methods increase the number of vertices compared to an object that used continuous mapping coordinates, but that does not contribute to the number of strips needed.