D3D9Client documentation

Requirements

D3D9Client uses external libraries that need support for the SSE2 Instruction set (GPSVC.DLL e.g.) and requires XNA Math support, so that the processor must support those Streaming SIMD Extensions.

Installation

You need a DirectX February 2010 or newer to run D3D9Client. To install the client itself you need to extract the zip package in the root folder of the Orbiter. In order to use a graphics client you need to run "Orbiter_ng.exe" instead of "Orbiter.exe". Also the client must be activated from the Modules tab.

DirectX Runtimes

If the redistributable package isn't installed in your computer you will receive an error message "The program can't start because d3dx9_42.dll is missing from your computer". Or you may see a pop-up window in Orbiter LaunchPad telling about a missing runtimes. If that happens then download and extract the content of the package in any empty directory you want and then find a Setup.exe and run it. You can delete the contents of the directory after the setup is completed. The directory is just a temporary storage for the installation files.

Here is a link: http://www.microsoft.com/en-us/download/details.aspx?id=9033

Orbiter Sound 3.5, Spacecraft3.dll

In order to use Spacecraft3.dll and Orbiter Sound (Version 3.5) with an external graphics client –like D3D9Client is one–, symbolic links in `/Modules/Server/` folder must be created.

Note, that for the current version of Orbiter Sound (Version 4.0) the "Sound" link is not necessary anymore. It will not be created via the according Button in D3D9Clients "Advanced Setup" dialog, if D3DClient detects that the loaded Orbiter Sound Module if of Version 4.0. Anyhow, it does not harm if the "Sound" link is present for a Orbiter Sound 4.0 setup.

The two symbolic links in `/Modules/Server/` folder for `Config` and `Sound` folders point (or link) to their "originals" located in the root folder of the Orbiter installation.
If you are using Windows XP or newer and your installation is on a NTFS filesystem, these links can easily be created from Video Tab -> Advanced Setup. If you have problems with creating the links you can also use a software called Link Shell Extension (see link below) or you can simply copy the `Config` and `Sound` folders into `/Modules/Server/`. But then you have to keep them updated manually.

Here is the link: http://schinagl.priv.at/nt/hardlinkshellext/hardlinkshellext.html

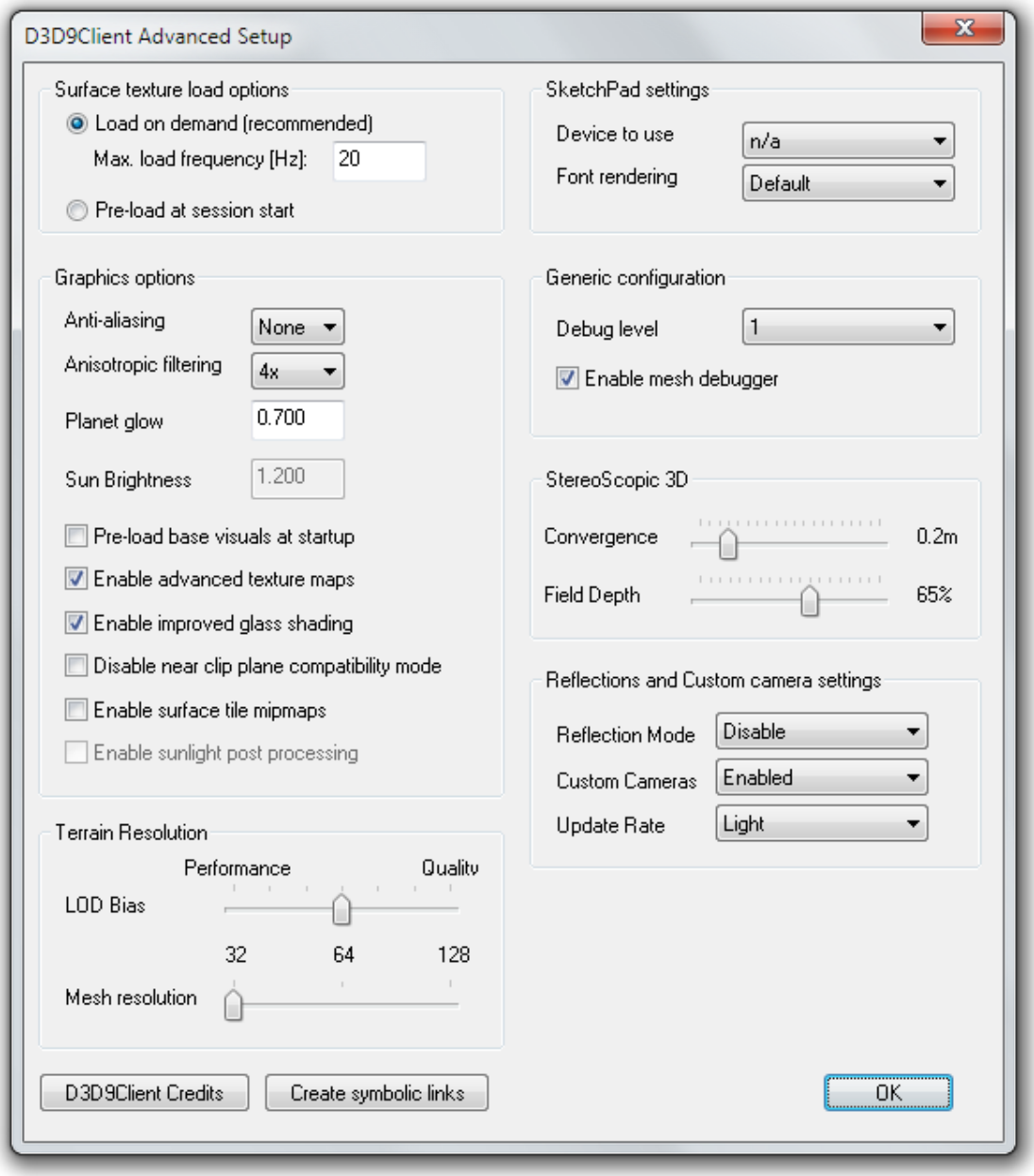
---

Fullscreen Mode and Alt-Tabing

D3D9Client doesn't support alt-tabing in a so-called "True Fullscreen Mode". Therefore, it's recommended that you use a windowed fullscreen mode that will run the orbiter in a fullscreen sized borderless window. There is also a conflict between a GDI based dialog (i.e. pop-up) windows and anti-aliasing in a true fullscreen mode which will disable the anti-aliasing.

D3D9Client Advanced Setup

Under the regular Video Tab of the Orbiter (Orbiter_NG) Launchpad you find the "Advanced" Button that will show the "D3D9Client Advanced Setup" Dialog. Here you can change several settings to tweak your experience with the D3D9Client.



On the lower left side of the Dialog you'll find the "Create symbolic links" button that will create the symbolic links that are needed for some add-ons. The symbolic links can only be created on an NTFS filesystem, so if you have installed your Orbiter installation for example on an FAT32 of extFS filesystem this feature will not work. In this case you have to copy the according folders as explained in the "Orbiter Sound, Spacecraft3.dll" chapter.

Surface texture load options

Here you can change the behavior how the D3D9Client will load surface textures. The two options available are either "Load on demand (recommended)" or "Pre-load at session start":

- **Load on demand (recommended)**

  With this recommended option selected the D3D9Client will only load surface textures when they come into view while you are orbiting a planet. The value in the "Max. load frequency [Hz]" input field lets you tune the maximum frequency the D3D9Client will check whether some new surface textures have come into view.

- **Pre-load at session start**

  With this option selected the D3D9Client will load all surface textures at startup of a scenario which results in a longer loading time.

Graphics options

Here you can change settings according to your graphic hardware:

- **Anti-aliasing**

  Depending on your hardware you can select the anti-aliasing feature that will "smoothen" the visual artifacts that occur when displaying edges.

- **Anisotropic filtering**

  Depending on your hardware you can select the level of anisotropic filtering. For further details about this topic take a look at Anisotropic filtering in the wikipedia.

- **Planet glow**

  The value set here is a factor that lets you brighten or dim the "shine" produced by the reflection of a planet. The higher the value the more the current orbiting planet acts like a additional light source. The supported value range is from 0.01 to 2.0. Although you might be able to enter bigger or smaller values, those will be clipped to that range.

- **Sun brightness**

  Currently disabled

Following checkboxes can be checked (enabled) or unchecked (disabled) to further fine-tune your D3D9Client experience:

- **Pre-load base visuals at startup**

  With this option ***enabled*** the D3D9Client will load all base visuals at startup of a scenario which results in a longer loading time.
  With this option ***disabled*** the D3D9Client will only load base visuals when they come into view while you are e.g. flying through the atmosphere or orbiting a planet.

- **Enable advanced texture maps**

  With this option enabled D3D9Client will try to add additional texture information for meshes that supply them. This advanced texture maps define for example how "rough" or "shiny" a texture appears.

- **Enable improved glass shading**

  With this option enabled D3D9Client will try to add improved glass shading (Fresnel reflection) for meshes that supply them.

- **Disable near clip plane compatibility mode**

  If the near clip-plane compatibility mode is enabled then the minimum clip-plane distance is 1.0 meters as is in the Orbiters internal engine. If the compatibility mode is disabled then the client can reduce the clip distance down to 0.1 meters, if there is a graphics close to the camera. This setting will only effect in so-called exterior pass. Virtual cockpit near clip-plane distance is defined in D3D9Client.cfg and the default value is 0.1 meters.

- **Enable surface tile mipmaps**

  In some situations planetary (or minor body) surfaces appear to be noisy and are flickering. Mipmaps are one solution to address these problems. By enabling this option these artifacts should greatly be reduced and the surface will appear smooth.
  The impact of enabling mipmaps is an increased load time because the mipmaps are generated during texture loading.

- **Enable sunlight post processing**

  Currently disabled

Terrain Resolution

Here you can change settings that will affect the rendering of planetary terrains.

- **LOD Bias**

  The Level of Detail (LOD) ...

- **Mesh resolution**

  The mesh resolution ...

SketchPad settings

The SketchPad is used in Orbiter to draw 2D graphics onto surfaces. These are for example MFD Displays or announcements in the Simulation.

- **Device to use**

  The two options you can choose from at "Device to use" are "GDI/DirectX" and "GDI Only".

  GDI/DirectX
  will use the DirectX 2D drawing capabilities of your graphic hardware to draw 2D surfaces. Normally this is the recommended setting, because it will not produce so much CPU load that the "GDI Only" option.

  GDI Only
  will only use GDI to draw 2D surfaces which might be the option when you experience any glitches or graphical artifacts in MFD screens. This mode is used for older graphic hardware to be able to run Orbiter.

- **Font rendering**

  The four options you can choose from at "***Font rendering***" are:
  "***Crisp***", "***Default***", "***Cleartype***" and "***Proof Quality***".

  Each setting will effect how the fonts are pre-rendered for sketchpad use. Choose what ever gives the best visual results. No performance impact.

Generic configuration

The generic configuration contains options that will change the general behavior of D3D9Client. For most of the time the default setup should be fine ("Default" Shader set and Debug Level "1"). But you can for example change the verbosity of the internal logging system to be able to report more detailed issue information when you experience an error or failure.

- **Debug level**

  The five options you can choose from here [0...4] represent the level of information that will be written into the log-file. The log-file is called D3D9ClientLog.html which can be found in *Modules\D3D9Client* directory. Higher values will create more detailed output.

  Until you have any problems and like to have more detailed information what's going on, you should keep this level reasonably low as higher values will result in more disk I/O what slows down the Simulation.

- **Enable mesh debugger**

  With this option enabled D3D9Client will provide additional mesh debugging options at runtime. The "D3D9 Debug Controls" dialog will provide a user interface to access several options. For further details see the D3D9 Debug Controls Dialog section below.

StereoScopic 3D

The stereoScopic 3D settings allow you to tweak the 3D experience if you are using a NVIDIA graphic card that provides this feature.

- **Convergence**

  The convergence value lets you choose "how far your eyes are apart". The default value is 0.2 meters (20 cm / 7.8 inches) which is round about the average distance between your eyes. Increasing this value might result in a view with "more depth".

- **Field Depth**

  In optics, particularly as it relates to film and photography, depth of field (DOF) is the distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image. In the context of Stereoscopic 3D it defines the distance where the relative position between the "right eye objects" and the "left eye objects" are overlapping exactly and switch their position when further away or closer that that "point". For further details about this topic take a look at Depth of field in the wikipedia.

Reflections and Custom camera settings

The reflections and custom camera settings allow you to tweak the parameters that define the behavior of the environment mapping feature. Environment mapping is a feature used to draw reflections of the surrounding environment on reflective surfaces. Custom camera settings will allow add-ons to render images on isolated surfaces (camera screens).

- **Reflection Mode**

  The reflection mode selection offers three settings:

  Disable will completely disable the environment mapping feature.
  Planet Only will only "reflect" the currently orbiting planet or moon.
  Full Scene will "reflect" the complete scene, that is all other ships, moons and planets.

- **Custom Cameras**

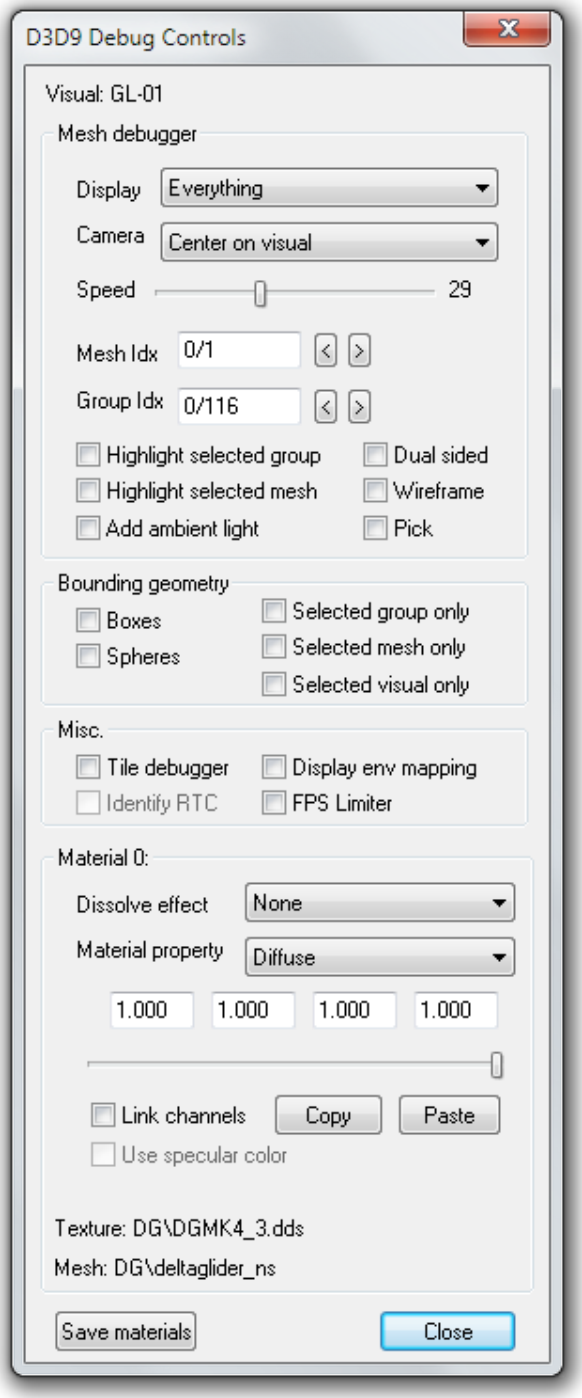
  The custom camera interface can be used by add-ons to create e.g. payload-bay cameras, docking cameras and other things like that. The camera image rendered by a custom camera can be displayed in an MFD screen or any other screen in a virtual cockpit or in a panel. Developer note : See ogci.h for additional details.
  The custom camera selection offers therefore only two settings: ***Enabled*** or ***Disable***.
  (Note: This has nothing to do with environment maps; it merely uses the same code and update mechanism to do the rendering)

- **Update Rate**

  Selects the number of environment map faces that will be rendered per frame. The update rate selection offers three settings: ***Light***, ***Medium*** or ***Heavy***

D3D9 Debug Controls Dialog

The D3D9 Debug Controls Dialog allows you look at different things while the simulation is running.
Such things are for example the highlighting of Meshes or Groups, so their location and look can be easily checked. Looking at a complete scene in wireframe mode might for example help in finding any mesh parts that are not placed correct.
The D3D9 Debug Controls Dialog also allows you to change the settings of materials at runtime, so that changes can be 'seen' while changing them.



Mesh debugger

The mesh debugger group contains elements to inspect a mesh. It gives a mesh developer some tools at hand to better inspect a mesh in its 'natural environment', the Orbiter Simulation environment.
With the options available in this group one can for example separate individual parts of the shown scene and leave out all the rest, so it might be easier to see what really happens at rendering.
Other settings change the environment so that the current lighting conditions do not apply, so when the mesh is currently in the shadowed part of the orbit it can still be seen with some artificial ambient light.
For all possibilities, please read the descriptions of the individual GUI- Elements, below.

- **Display**

  This combo box lets you select what parts of a scene is rendered. The possible options are: "Everything", "Selected Visual", "Selected Mesh" or "Selected Group".

  Everything will, as the name suggests, display everything. This will show the scene as it will be shown normally when running a Orbiter simulation session.
  Selected Visual will display only the currently selected visual. Planets and moons for example are excluded.
  Selected Mesh will display only the currently selected mesh. If for example two vessels are in the complete scene, only one of them will be displayed. All other vessels, planets and moons are excluded.
  Selected Group will display only the current selected group. This is often just a "part" of a vessels mesh. All other groups, vessels, planets and moons are excluded.

- **Camera**

  This combo box lets you select the camera-mode. It can be either "Center on visual" or "Wheel Fly/Pan Cam".
  Center on visual will select the 'normal' Orbiter camera operations where with pressed right mouse button you can 'drag' the camera around the current selected visual.
  Wheel Fly/Pan Cam will select another camera-mode where with pressed left mouse button you can pan the camera left/right or up/down and with the right mouse button pressed it can be 'tilted'.

- **Speed**

  This control lets you change the speed of the camera movements when you are in "Wheel Fly/Pan Cam" camera-mode (see Camera).
  The value is a kind of 'factor' applied to the movement of the mouse (or the mouse-wheel). It ranges from 1 (being the lowest speed) to 8192 (beeing very fast).

- **Mesh Idx**

  This control lets you select one mesh of a multi-mesh vessel by its index. If a vessel consists of a "hull"-mesh and a cockpit-mesh, you will be able to select the two meshes individually by selecting the according mesh index here.

- **Group Idx**

  This control lets you select one group out of the current selected mesh (see Mesh Idx) of a multi-group mesh by its index. If a mesh consists of multiple groups, you will be able to select the individual groups by selecting the according group index here.

- **Highlight selected group**

  With this option ***enabled*** the current selected group will be permanently highlighted (green).

- **Highlight selected mesh**

  With this option ***enabled*** the current selected mesh will be permanently highlighted (blue).

- **Add ambient light**

  With this option ***enabled*** the current selected mesh will be lit by artificial ambient light independent of the current 'global' lighting conditions. The mesh is then not only lit from the sun (lighting only parts of the mesh), but from all sides. This also applies to situations when the mesh is usually not lit at all, when on the night (dark) side of an orbit.

- **Dual sided**

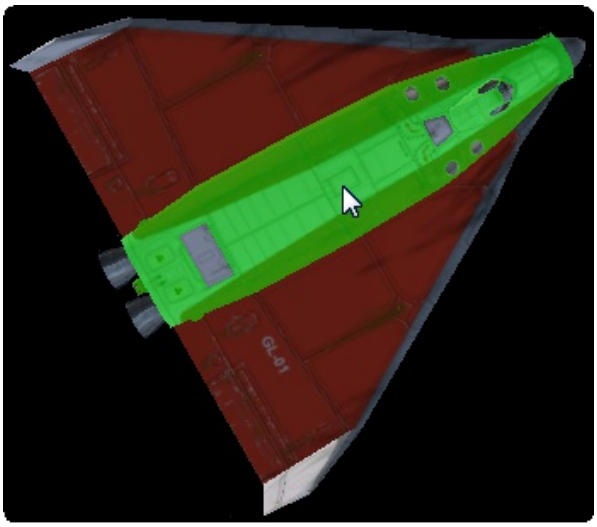
  Faces are normally only rendered from one side (the "backside" is kind of completely transparent). With this option ***enabled*** all faces are rendered opaque from both sides.
  This option might help developers to identify parts of the mesh that are "sticking out" of another mesh-part, that are normally only to be seen from the "inside". A cockpit mesh inside a hull mesh can be for example much bigger than the hull and it will not be easy to find out, cause from the "inside", the hull faces are transparent so the (to big) cockpit mesh can be seen. And from the "outside" the cockpit faces are transparent although "covering" the hull.

- **Wireframe**

  With this option ***enabled*** the scene (or only parts of it, see Display) will be rendered as wireframe model. In a wireframe model only the edges (the lines connecting vertices) of the meshes are drawn. This can be helpful to identify 'useless' parts a mesh.

- **Pick**

  The Pick option is one of the most useful tools for selecting materials. With this option ***enabled*** you can just select a material by clicking onto it with the left mouse button.



Pick in action

Whenever a material is picked it will light up in green, so it is easy to see which material is chosen. After a material is chosen all the settings in the lower part of the D3D9 Debug Controls Dialog apply to that material.

Bounding geometry

The options in this group offers additional rendering of bounding boxes or spheres of individual mesh parts.

- **Boxes**

  With this option ***enabled***, objects of the scene will have their bounding boxes also drawn.

- **Spheres**

  With this option ***enabled***, objects of the scene will have their bounding spheres drawn. The sphere represents the smallest volume that still fits the complete geometry.

- **Selected group only**

  This is one of three options that let you limit the parts of the mesh that are rendered. With this option ***enabled***, only the currently selected **group** will be rendered.

- **Selected mesh only**

  This is the second of three options that let you limit the parts of the mesh that are rendered. With this option ***enabled***, only the currently selected **mesh** will be rendered.

- **Selected visual only**

  This is the third of the three options that let you limit the parts of the mesh that are rendered. With this option ***enabled***, only the currently selected **visual** (vessel mesh) will be rendered.

Misc.

This group contains some debugging options to assist on more severe problems or let you see the current environment, that is used for the environment mapping feature of the D3D9Client.

- **Tile debugger**

  Client development utility/debugger. Unimportant from user point of view.

- **Identify RTC**

  Identifies GDI - Render Target conflicts by flashing surfaces in red and yellow. These conflicts can cause a major framerate impact.

- **Display env mapping**

  With this option ***enabled*** the environment mapping "box" will be displayed as a flattened box. This will become a cross-like area that shows the environment of each of the sides of that "virtual box".
  This box can be imagined as a box with mirror-surfaces which show the environment the face of the box 'sees'

- **FPS Limiter**

  With this option ***enabled*** a frames per second limiter is enabled that will limit the number of frames that will be rendered per time interval (per second that is). Limiter FPS value is set using D3D9Client.cfg
  This feature is created to reduce unnecessary GPU/CPU load. Not to produce a stable frame rate. Improper setting can cause extreme tearing on the screen. Setting the frame rate limiter to 200fps usually works pretty well. It is recommended to use vertical sync feature from a video tab instead of this.

Material 'X'

This group contains all the elements that enable you to tweak the different parameter of materials.

- **Dissolve effect**

  This field is used to select a special effect texture for dissolve effect.

- **Material property**

  Diffuse
  Defines diffuse material color [RGBA]. Texture is modulated with this color. The range for every property in this section is [0.0 to 1.0] unless otherwise noted.

  Ambient
  Defines ambient material color [RGB].

  Specular
  Defines specular material color [RGBP]. The last field is a specular power from [0.0 to 1000.0].

  Emissive
  Defines emissive material color [RGB].

  Reflect
  Defines reflection color [RGB] for a metallic (mirror a-like) reflection which intensity is independent from a viewing angle. This is also used to define minimum reflection offset for a fresnel reflection. Due to some implementation problems a non-zero value must be entered to enable fresnel reflections. For a mirror like reflections a typical range is [0.0 to 1.0] and for a fresnel reflections a typical range is [0.05 to 0.2].

  Dissolve
  Dissolve is an experimental technique that can be used to create blurry or noisy reflections. Currently it can be only applied to non-normal mapped surface that has a texture. It would be possible to use a screen-space coordinates to create a similar effect to a non-textured materials as well. The first field is the effect scale factor (i.e. particle size). The second field is the effect strength. If either parameter is zero the effect will be disabled. This property requires dissolve effect texture.

  Fresnel
  The first field in fresnel parameters is a power value. A typical range for the power is [2.5 to 4.5]. The power value will effect in a viewing angle dependency of the reflection. The second field is a multiplier. It will define a maximum reflection intensity when viewed from a shallow angle. This value must be zero to disable fresnel reflections otherwise a typical range is [0.8 to 0.95].

- **(4) Channel values & Slider**

  Meaning of the fields will depend about the selected material property. See properties from above.

- **Copy & Paste**

  Copy and paste buttons can be used to copy [RGB] values from different material properties for an example from specular color to reflect color.

- **Link channels**

  The link checkbox allows the link the R, G and B fields and after that they can be adjusted simultaneously with the slider.

- **Use specular color**

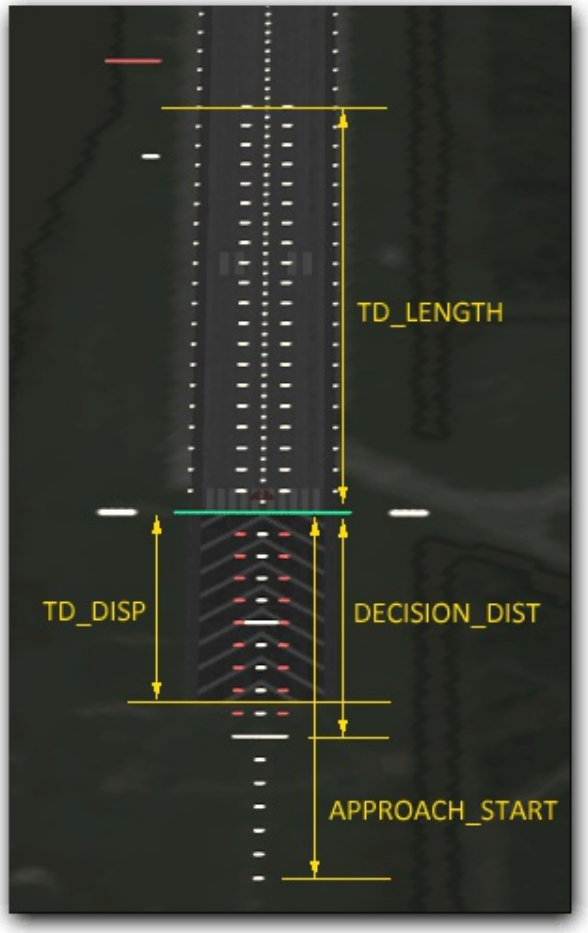
  Not implemented.

Save materials (*Button*)

The ***save materials*** button in the "D3D9 Debug Controls" Dialog does just that: it saves the materials ;)
Once you are happy with the result you have adjusted in the Material 'X' group, click on "save materials" and the changed material specifications will be saved in the `Config\GC\` folder of your Orbiter installation.
Then of course you can tweak these file(s) in `Config\GC\` folder with notepad etc.
So next time you fly with a ship of the same class it will have the properties, you have defined or changed, applied.
If you want to have the "default" look back, just delete the according file from that folder (e.g. `Config\GC\DeltaGlider.cfg`) and the standard Orbiter appearance will be restored.
The install ZIP of D3D9Client contains some of those files already, that contain some nice "advanced looks" for some of the standard vessels. You can always take those and place it back into that folder to get the "D3D9-Default" look.

Base Configuration settings

The D3D9Client allows you to to configure some more details when settings up runway lights.
Additionally to the parameters that Orbiter itself defines (see *Doc\OrbiterConfig.pdf* for further details) there are some extra parameters you can define.
For a better understanding how some of those parameters will affect the rendering of the runway lights this image might help:



<RUNWAYLIGHTS>

- **<END1 V>**

  First end point of runway (center line).

- **<END2 V>**

  Second end point of runway (center line).

- **<WIDTH F>**

  Runway width (m)
  Note: Two different light configurations exists (wide>59m) and (narrow<59m)

- **<PAPI F F F I I>**

  Precision Approach Path Indicator (PAPI).
  Parameters:
  Designated approach angle (deg)
  Approach cone aperture (deg)
  Offset of PAPI location from runway endpoints. (m)
  PAPI Mode (int:0-3) (this is optional parameter)
  PAPI runway end point (int: 0-1) (if this parameter is not defined then PAPI is added in both ends)

  PAPI Modes:
  0: On center line
  1: On left side
  2: On right side
  3 or none: On both sides

  PAPI Endpoint:
  0: PAPI lights added to END1 only
  1: PAPI lights added to END2 only
  Not Defined: PAPI lights added in both ends

  Note: Runwaylights can have 12 PAPI lights. Orbiter's inline engine will ignore the last parameter. If more than one PAPI entries exists in the runwaylights the inline engine will only use the last one.

- **<VASI F F F I>**

  Visual Approach Slope Indicator (VASI).
  Parameters:
  Designated approach angle (deg)
  Distance between white and red indicator lights (m)
  Offset of VASI (red bar) location from runway endpoints (m)
  VASI runway end point (int: 0-1) (if not defined then VASI is added in both ends)

  Note: Runwaylights can have 2 VASI lights.

- **<TD_DISP F>**

  Touch Down displacement (m). (default: 0m)
  Displacement between runway endpoint and the green line.

- **<TD_DISP2 F>**

  Touch Down displacement for the other end of the runway (m). (default: 0m)
  Displacement between runway endpoint and the green line. If this value isn't specified then TD_DISP is used for both ends of the runway.

- **<TD_LENGTH F>**

  Length of the Touchdown zone (m). (default: 600m)
  Two columns of lights on a runway each containing 3 parallel lights.

- **<DECISION_DIST F>**

  Length of the "red lights" zone (m). (default: 257m)
  This zone contains 2 x 3 x 9 red lights and the spacing between lights will depend about the length of the zone. The touchdown zone will use the same spacing about 30m.

- **<APPROACH_START F>**

  Length of the approach lights from the green line (m). (default: 900m)
  It's the long column of 5 parallel lights.

- **<SINGLEENDED>**

  If the singleended keyword is defined then the lights are only rendered when approaching a runway from END1 towards END2. If you want a asymmetric runwaylights then you need two runwaylight sections in a base configuration file.

- **<CATEGORY>**

  Defines the category of runway lights. 1 = SSALR, 2 = ALSF-II, If this value isn't specified then the category is automatically selected based on a runway width. ALSF-II is used if the width is greater than 59m.

Example of runway lights for KSC:

```
RUNWAYLIGHTS
  END1 -8220 -3 -600
  END2 -12670 -12 -3155
  WIDTH 100
  PAPI 5.0 3.0 257 3      ; both sides of the green line, in both ends
  PAPI 20.0 3.0 -2000 0 0 ; on a center line 2km before rwy in END 1
  PAPI 20.0 3.0 -2000 3 1 ; both sides of center line, 2km before rwy, in END 2
  VASI 1.5 152 671
  TD_DISP 257
  TD_LENGTH 600
  DECISION_DIST 257
  APPROACH_START 900
END
```

Advanced Texture Maps

Additional texture maps can be automatically assigned for a mesh simply by placing additional textures into a texture folder. Additional textures are identified by using an identifier in the end of the textures name like "dgmk4_1_bump.dds" where "dgmk4_1.dds" is the name of the base texture.

A fully specified texture set could for example consist of these files:

```
cube.dds       <= 'base' texture
cube_bump.dds  <= 'advanced texture Bump-map'
cube_spec.dds  <= 'advanced texture Specular-map'
cube_emis.dds  <= 'advanced texture Emission-map'
cube_refl.dds  <= 'advanced texture Reflection-map'
```

Available identifiers are:

- **<_norm>**

  Tangent space normal map and the valid formats are:

  <R8G8B8> 3-bytes per pixel. Best quality (uncompressed)
  <V8U8> 2-bytes per pixel. Good quality (uncompressed)
  <DXT1> 1-byte per pixel. Bad quality (compressed)

  Note: The configuration of this (*_norm*) identifier is ignored if a *_bump* configuration is also found. However, the *_norm* configuration is the preferred one that should be used when you have the choice (see also _bump).

- **<_spec>**

  Specular map controls a specular reflection in per pixel basis. Alpha channel is containing a specular power setting. Value 255 is mapped to maximum matarial defined power and 0 is mapped to 0.0. Specular map is modulated by specular material color. Valid formats are <R8G8B8A8>, <DXT3> or <DXT5>.

- **<_bump>**

  Bump maps are also supported by the D3D9Client. They are automatically converted into normal maps during loading of bump maps (see note).
  The **recommended** formats are <A8> and <L8>:

  <A8> 1-byte per pixel alpha
  <L8> 1-byte per pixel luminance

  From the **not recommended** multi-channel textures only the **red** channel is used;
  They are:
  <R8B8G8> 3-bytes per pixel (only red is used anyway
  <DXT1> 1-byte per pixel compressed
  <R16F> 2-bytes per pixel (might work, not tested!)

  Note: The configuration using the (*_bump*) identifier will overwrite any *_norm* configuration in case both identifiers are found. The *_norm* configuration is however the one that should be used when you have the choice (see also _norm).

- **<_emis>**

  Currently emission map works more like a light map and the simplified equation is:

  ```
  pixel_color.rgb = texture.rgb * clamp(emission_map.rgb +
                                        sun_light.rgb + local_lights.rgb)
  ```

  Alpha channel is ignored therefore the recommended formats are <R8G8B8> or <DXT1>.
  The exact implementation would require some discussion with an add-on developers to define the function for the emission map. An other possibility is:

  ```
  pixel_color.rgb = texture.rgb * clamp(sun_light.rgb + local_lights.rgb)
                  + emission_map.rgb
  ```

  Of course, the clamp function can be changed to a different kind of color curve manipulation function to control contrast and lightness.

- **<_refl>**

  Reflection map controls material reflectivity and color in a per pixel basis. Fresnel reflection works independently from a reflection map. However, a zero (black) reflection map value will mask off the fresnel reflections as well. A non zero value will apply full fresnel reflections for that pixel. Only [RGB] channels from a texture are used.

- **<_transl>**

  Translucence map controls the material translucency on a per pixel basis. The texture behaves much like a diffuse texture, except that it is illuminated from behind. Illumination is based on the angle of the sun relative to the surface; if the sun is directly behind the surface, the illumination is strong, and if the sun is at a low angle but still behind the surface, the illumination is dim. If the sun begins to illuminate the surface from in front, the translucence effect is no longer visible.
  Only [RGB] channels from a texture are used.

- **<_transm>**

  Transmittance map controls the amount of light that passes through a translucent material without being diffused. The effect is very similar to a specular reflection, except that it simulates the bright spot on a semi-transparent material that is directly between the sun and the observer.
  The [RGBA] channels from a texture are used. The RGB channels provide color and brightness, while the alpha channel changes the size of the bright spot. This is analogous to how the specular map is used.

Note, that not all objects support all advanced texture maps:

Type	Identifier	Vessel	Base	Planetary Surface
Bump-map	`_bump`	supported	supported	not supported
Specular-map	`_spec`	supported	supported	not supported
Emission-map	`_emis`	supported	supported	not supported
Reflection-map	`_refl`	supported	not supported	not supported
Translucence-map	`_transl`	supported	not supported	not supported
Transmittance-map	`_transm`	supported	not supported	not supported