



# Indigo Renderer & Indigo RT Manual



## Indigo Manual

Welcome to the Indigo Renderer Manual.

This manual is for Indigo Renderer and Indigo RT, versions 4.2 or newer.

This manual can be found both online at <https://www.indigorenderer.com/documentation/manual>, and in PDF format at [https://downloads.indigorenderer.com/dist/manual/Indigo\\_Manual.pdf](https://downloads.indigorenderer.com/dist/manual/Indigo_Manual.pdf).

It can also be found in the Indigo Renderer and Indigo RT distributions in PDF format.

If you have Indigo installed on your OS X computer, you can access the Indigo Manual PDF in the 'Indigo Renderer' folder in your Applications folder.

If you have any comments, corrections or questions about this manual, please contact us at [support@indigorenderer.com](mailto:support@indigorenderer.com)

## Overview

This manual covers the Indigo user interface, scene attributes and functions.

It also covers all supported exporters and their integrated functions.

- **Getting Started**

This section covers installing Indigo and activating your licence.

- **Indigo Interface**

This section covers elements of the standalone Indigo user interface.

- **Rendering with Indigo**

This section covers various aspects of rendering with Indigo, such as the render modes, environment settings and material types.

- **Techniques**

This section covers specific techniques and tips for rendering with Indigo.

- **Indigo for SketchUp**

This section covers integration with Google SketchUp.

- **Indigo for Blender**

This section covers integration with Blender.

- **Indigo for 3ds Max**

This section covers integration with Autodesk 3ds Max.

- **Indigo for Cinema 4D**

This section covers integration with Maxon Cinema 4D.

- **Indigo for Revit**

This section covers integration with Autodesk Revit.

- **Indigo for Maya**

This section covers integration with Autodesk Maya.

An in-depth Technical Reference Manual is also available (included with the Indigo distribution), aimed mainly at exporter developers.

## Principles of physically-based rendering



Image by Lemo

Indigo Renderer delivers photorealistic results through strict adherence to the physics of light. This has some consequences for how the scenes it renders should be modelled, which will be outlined in this section.

Furthermore, the progressive nature of its rendering process means that an image never “finishes rendering” - the image quality improves until you are satisfied with the result and end the rendering process.

### Relationship to photography

Rendering with Indigo Renderer is conceptually more similar to photography than it is to using conventional computer graphics applications.

Exporters for Indigo convert the virtual worlds you create in your 3d program into the real world representation used by Indigo.

### Units of measurement

Indigo works in units of metres, and your Indigo exporter will export your scene based on the set scene scale – you can work in either metric or imperial.

The use of physical units is crucial in Indigo; for example, window panes should have thickness so that they can properly absorb light as it travels through the glass. If the thickness were specified as 1.2km instead of 1.2cm the glass would appear black, as all the light has been absorbed in travelling through it.

### Realistic materials



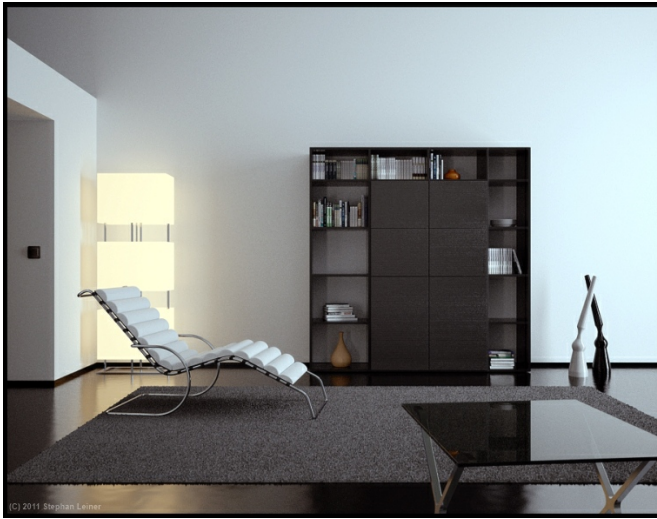


Image by Camox

Because Indigo is a physically-based renderer, the material properties of objects must be specified in physical terms. Your Indigo exporter will provide the necessary tools and options to allow you to do this in your 3d program. Indigo also comes with the Indigo Material Editor for direct control of materials.

### Progressive rendering

Exactly as happens in the real world, Indigo simulates photons emitted from light sources which interact with the scene before entering the camera; the longer a render is left running, the more photons contribute to the final image.

Shorter renders will have a grainy appearance, much like using a high ISO setting and fast shutter time in a real camera. This grain decreases with time: The amount of noise is initially quite high, but very sharply decreases. Eventually a plateau is reached where the image quality is not noticeably improved with further rendering, and rendering can be stopped.



1 second



30 seconds (Scene by Arthur Liebnau)

In general, most people will just leave their scene rendering until it looks clear enough for their purposes. As time goes on, the image will become clearer and clearer.

### Image quality, samples per pixel

Indigo measures its render progress as the number of samples that have been calculated. You can imagine these as being the number of "light particles" that have bounced around the scene and hit the sensor in your camera.

Depending on the complexity of your scene, it may take up to 10000 samples per pixel before the image starts to become clear enough for your purpose. Many scenes will start to resolve at around 1000 samples per pixel, and some scenes look good after 100 samples per pixel.

## Getting Started

To render with Indigo you'll need to install the actual Indigo application, and an exporter for your preferred 3D modelling application.

The Indigo installer comes bundled with installers for the latest SketchUp, Blender, Cinema 4D and 3ds Max exporters. On running the Indigo installer, it will attempt to auto-detect which of these packages you have installed and launch the installers for the corresponding exporter modules.

The latest stable release of Indigo and the exporter modules can be downloaded from <http://www.indigorenderer.com/download>.

We also regularly provide beta versions, generally with good stability, on our [News and Announcements forum](#). These are recommended for confident users, or if you experience any problems in an older release.

## Exporters

It is strongly recommended that you install the main Indigo application before installing any exporters, so that they can find the existing Indigo installation and link to it automatically. (This is normally done for you by the Indigo installer since version 3.2.)

It's possible to have a mismatch between the installed Indigo version and the exporter version; this can result in Indigo displaying an error message about unknown data in the scene file. In this case updating to the latest versions of both will generally resolve the issue.

For more information about the various Indigo exporters, please consult the corresponding manual or our [Exporters forum](#).

# Installing Indigo

To download Indigo, go to <http://www.indigorenderer.com/download/> and select the appropriate version of Indigo for your platform (Windows, Macintosh or Linux). You should download the latest available version of Indigo, which will be listed at the top of the page.

## System Requirements

Indigo will run on most modern computers (generally Pentium 4 or newer).

Please see the final section for GPU acceleration requirements and suggestions.

### Minimum system specifications:

#### Windows:

- 64-bit x86 CPU with SSE4
- 4GB of RAM
- 150MB of hard drive space
- Windows 7 or newer

#### Mac:

- 64-bit x86 CPU with SSE4
- 4GB of RAM
- 150MB of hard drive space
- OS X 10.8 Mountain Lion or newer

#### Linux:

- 64-bit x86 CPU with SSE4
- 4GB of RAM
- 150MB of hard drive space

### Recommended system specifications:

- 16GB+ of RAM

### GPU acceleration

#### Required:

- Either an NVIDIA GPU supporting OpenCL 1.1 (GeForce 9800 GT or newer), or an AMD GPU supporting OpenCL 1.1 (Radeon 4xxx or newer)
- 256MB or 512MB of onboard GPU memory, depending on OS (more info)
- 2GB of system memory

**Recommended:**

- Either NVIDIA GeForce GTX 5xx / Quadro 4000 / Tesla C2050 or newer, or AMD Radeon 5xxx / FirePro 3D V3800 / FireStream 9350 or newer
- Quad-core or greater Intel or AMD CPU

### GPU memory requirements

We recommend 512MB instead of 256MB as the minimum on Mac OS and Windows Vista/7 using the Aero desktop, since it uses more GPU memory for desktop window management.

If you are using an older GPU with relatively modest specifications, we encourage you to try the trial version before purchasing.



## Indigo for Windows

### Downloading and installing

Download the newest version of [Indigo for Windows](#), then double click the installer file.

#### 1. Agree to the licence

You will be presented with the Indigo End User Licence Agreement. You should read through it and click "I Agree" if you agree with the terms.

#### 2. Choose components

We recommend you leave the check-boxes selected but you are able to disable each component if you have particular requirements on your system. Press Next to continue.

#### 3. Choose an installation location

We recommend you use the default installation location, as most Indigo exporters will expect to find Indigo here – however you are able to change this path if you need to. Indigo will write a registry key that exporters will use to find Indigo automatically if Indigo is installed in a non-standard location. Press "Install" to complete the installation.

### After installation

Once the installation has completed, you can find Indigo from the start menu under the "Indigo Renderer" or "Indigo RT" sub-menu, along with links to this manual and other program shortcuts for convenience.

## Indigo for Macintosh

1. Download the newest version of Indigo for Macintosh then double click the disk image mount it.
2. A dialog will pop up, prompting you to drag the Indigo icon to your Applications directory:



3. Drag the Indigo icon to the Applications directory.
4. Indigo is now ready to be used on your Mac.

## Indigo for Linux

1. Download the latest Indigo for Linux from <http://www.indigorenderer.com/download>
2. Extract the archive that you downloaded, for example with the command:

```
tar -xzf IndigoRenderer_x64_v3.4.18.tar.gz
```

This will extract the archive into the directory `dist/IndigoRenderer_x64_v3.4.18`

3. Indigo is statically linked as far as possible and should be ready for use. Run `./indigoconsole -v` to see if Indigo installed correctly, for example:

```
cd IndigoRenderer_x64_v3.4.18/  
./indigo_console -v
```

Which should print out something like

```
Indigo Renderer v3.4.18, Linux 64-bit build.
```

To run the graphical user interface, run `./indigo`.

You are now ready to run Indigo.

## Indigo Licensing

You are welcome to learn Indigo and use it for non-commercial renders without paying for a licence. The following restrictions are present in the free version of Indigo:

- Maximum resolution of 1.0 Megapixels – e.g. 1000 pixels by 1000 pixels.
- An Indigo logo is placed in the bottom right of the image.
- May not be used for commercial work.
- No customer support beyond that given in the Forum.

If you need to create renders at higher resolutions or produce renders as part of your business, then you need to buy a commercial licence for Indigo. You can buy licences online at:

<http://store.glaretechnologies.com/>

Once you have purchased a licence, you can instantly enable the full features of Indigo. Your licence will be locked to the hardware of your computer, and you must contact [support@indigorenderer.com](mailto:support@indigorenderer.com) if you need to move the licence to another computer.

The licence for your computer is based on the CPU model of your processor and MAC address of your network card. You should avoid changing your network card regularly when using Indigo – for example avoid enabling and disabling your network card, as this may confuse the licensing software that Indigo uses.

If you wish to purchase multiple licences, you can use the network floating licence feature, managed by the Network Manager. To purchase these licences please contact us at [sales@indigorenderer.com](mailto:sales@indigorenderer.com)

There are two different kinds of Indigo licences:

- GUI Licence – for use on your desktop computer
- Node Licence – for use on networked render slaves

Having a GUI Licence does not mean that you can use unlicensed render slaves to generate a high resolution image, you will need a node licence for each computer that will be used to help you render your image.

If you have any problem with your Indigo licence, you can always contact us at [support@indigorenderer.com](mailto:support@indigorenderer.com) and we'll get you up and running as soon as possible!

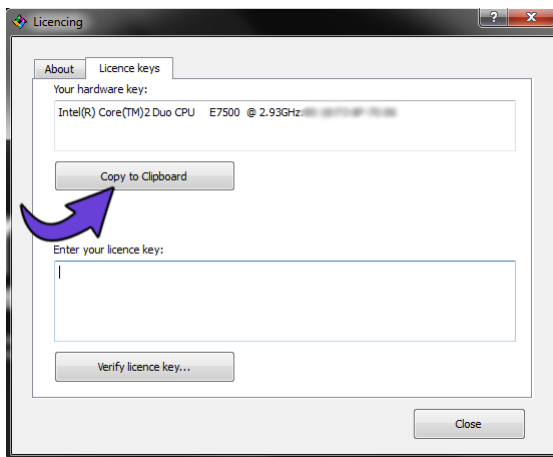
## Indigo Licence Activation

Once you have purchased an Indigo Licence from the Glare Technologies Store you need to go through a few steps to activate it with Indigo.

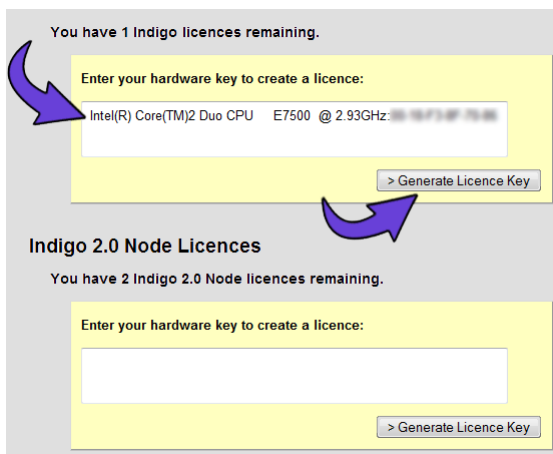
1. Open Indigo and click on the Licensing button.



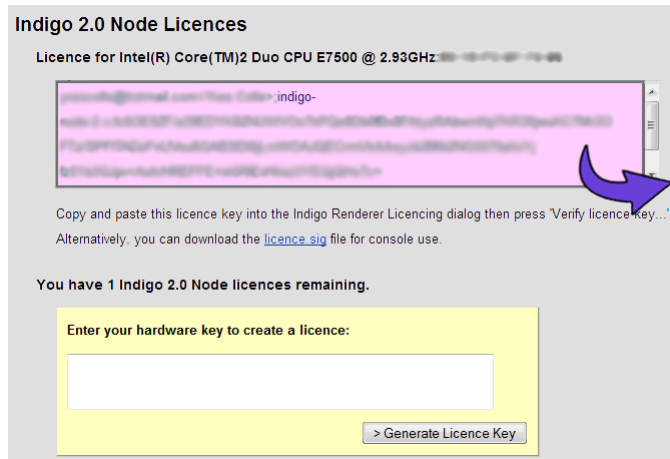
2. Press Copy to Clipboard to copy the hardware key. Do not copy the text manually.



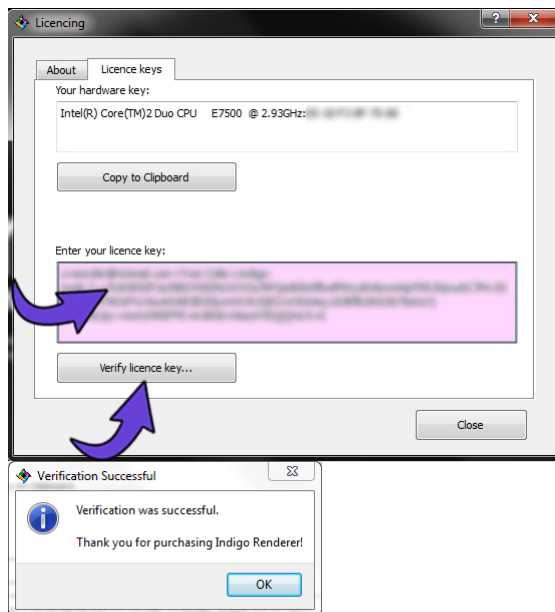
3. Upon purchase of a licence, you will receive an email with a link to the store page with your details on it. Keep this link somewhere secure, and check that your details are correct.
4. Scroll down to the bottom section. Paste in your hardware key into the appropriate box, depending on the type of licence you own, and press "Generate Licence Key".



5. A bunch of characters will be returned in a text box, copy it all.



6. Paste it back into the Indigo Licence dialog box at the bottom, and press "Verify licence key".



Success!

7. The Indigo background will also tell you if it is verified.



## Floating Licence Activation

### Getting the server Hardware ID

First, decide which computer you will use as a licence server.

Install Indigo Renderer (which can be downloaded from <http://www.indigorenderer.com/download>) on your server.

Run the "Indigo Network Manager" on the server.

(For example, on Windows, this can be accessed Start -> All Programs -> Indigo Renderer -> Indigo Network Manager)

Press the Licensing button, then "Copy to Clipboard" to copy the hardware ID for the server.

Email the hardware ID to [support@indigorenderer.com](mailto:support@indigorenderer.com)

If you have purchased floating licences, we will reply with the licence file based on the hardware ID and the number of licences purchased.

### Using the Licence Key

Once you have received your licence key by email, open up the Licensing dialog in the Indigo Network Manager again. Then paste the licence key into the text field labelled "Enter your licence key".

Finally, press "Verify licence key" and if the licence key is valid, the "licence status" field will display a message similar to the following:

```
Licensed to 'Nick Chapman'  
Email: 'nick@indigorenderer.com'  
5 x Network Floating Full
```

If the licence key is not valid, an error message will be displayed instead in the licence status field; please contact [support@indigorenderer.com](mailto:support@indigorenderer.com) for assistance.

### Using the Floating Licences

#### Obtain the Licence server's hostname

A hostname is a name used to identify a device connected to a computer network.

See the [Network Rendering tutorial](#), step 3, for more information on this.

**Make sure the Indigo Network Manager is running on the licence server, and the licence key is loaded into it as described earlier.**

## Set up Indigo on your workstation(s)

On your workstation computers, install Indigo Renderer.

Start Indigo Renderer.

1. Click on the "Options" tool-bar button.
2. Select the "Network" tab in the options dialog.
3. Ensure that the "Use network manager" option is checked.
4. In the "Network manager hostname" field, enter the licence server hostname.
5. Ensure that the "Use floating licence" option is checked.
6. Ensure that the "Do master search broadcast" option is unchecked.
7. Click "OK" to save the changes and exit the options dialog.

## Test retrieval of the floating licence

Restart Indigo on your workstation.

When Indigo starts, the Indigo logo and some information will be displayed in the main window.

If the floating licence retrieval was successful, the Licence type will be 'Network Floating Full' and the Licensed to will be 'Network Floating Licence User', as shown:



## Troubleshooting

### Network ports

Some ports will need to be opened on the licence server, so that Indigo running on the workstations can connect to it: TCP Port 7200.

### Network manager logging



The Indigo Network manager writes out reasonable comprehensive logs to *network\_manager\_log.txt* in the appdata directory, for example *C:\Users\nick\AppData\Roaming\Indigo Renderer\network\_manager\_log.txt*.

Looking at this log can give useful information if floating licences are not working.

## Upgrading from Indigo 3 to Indigo 4

To upgrade your Indigo 3 licence to an Indigo 4 licence, follow these steps:

When you purchased an Indigo 3 licence, you will have received an email from [billing@indigorenderer.com](mailto:billing@indigorenderer.com), which contains a link to your private licence page on [store.glaretechnologies.com](http://store.glaretechnologies.com).

If you can't find the email, please contact [support@indigorenderer.com](mailto:support@indigorenderer.com) for assistance.

Please follow this link to go to your licence page.

Then click on the "Buy Now" button in the "Upgrade to Indigo 4" section.

Thank you for purchasing Indigo

We have sent an email to [redacted] with your order details and a link you can use to access this page at any time.

Your order has been processed and your licence keys are ready for download.

**How do I generate my licence key?**

Read the Indigo Manual to [learn how to access your hardware key](#). You then copy and paste your hardware key into the licencing text areas below, and click 'Generate Licence Key'. Copy and paste the licence key into the Indigo Licencing dialog then press 'Verify licence Key' to unlock the full version of Indigo.

Alternatively, you can download the licence.sig file for console use.

**Upgrade to Indigo 4**

Upgrade your Indigo 3 licence to an Indigo 4 licence

[Buy Now](#)

Upon clicking the "Buy Now" button, you will be taken to an order summary page for your upgrade order, which will summarise the cost and allow you to select the payment method to complete the order.

## Upgrading from Indigo RT to Indigo Renderer

To upgrade your Indigo RT licence to an Indigo Renderer (Full) licence, follow these steps:

When you purchased an Indigo RT licence, you will have received an email from [billing@indigorenderer.com](mailto:billing@indigorenderer.com), which contains a link to your private licence page on [store.glaretechnologies.com](http://store.glaretechnologies.com).

If you can't find the email, please contact [support@indigorenderer.com](mailto:support@indigorenderer.com) for assistance.

Please follow this link to go to your licence page.

Then click on the "Buy Now" button in the "Upgrade to Indigo 4" section.

Thank you for purchasing Indigo

We have sent an email to [redacted] with your order details and a link you can use to access this page at any time.

Your order has been processed and your licence keys are ready for download.

**How do I generate my licence key?**

Read the Indigo Manual to [learn how to access your hardware key](#). You then copy and paste your hardware key into the licencing text areas below, and click 'Generate Licence Key'. Copy and paste the licence key into the Indigo Licencing dialog then press 'Verify licence Key' to unlock the full version of Indigo.

Alternatively, you can download the licence.sig file for console use.

**Upgrade to Indigo 4**

Upgrade your Indigo RT 4 licence to an Indigo 4 licence

[Buy Now](#)

Upon clicking the "Buy Now" button, you will be taken to an order summary page for your upgrade order, which will summarise the cost and allow you to select the payment method to complete the order.

## Online Licensing

Online licensing allows you to use your Indigo licence on any computer connected to the internet. You can only use a single licence on one computer at a time however.

Online licensing is new in Indigo 4.4.

### To use Online licensing

#### Make an account

Go to <https://store.glaretechnologies.com/> and make an account with the 'Signup' link on the top right of the page.

#### Link your Indigo order to your account

Once you have done that, you will need to go to your individual order page, the link to which was sent to you when you purchased an Indigo licence.

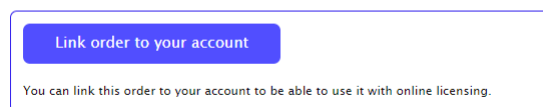
The link will look something like <https://store.glaretechnologies.com/orders/xxxxxxxxx>

On your order page, click the 'Link order to your account' link.

Thank you for purchasing Indigo

We have sent an email to [nick@indigorenderer.com](mailto:nick@indigorenderer.com) with your order details and a link you can use to access this page at any time.

Your order has been processed and your licence keys are ready for download.



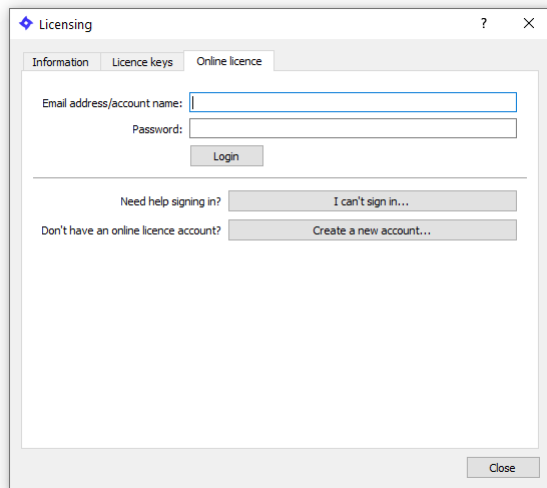
You should see a message like "Succesfully claimed order #12345".

#### Log in from Indigo

Assuming you have a new enough Indigo build (version 4.4), run Indigo, then select the menu command

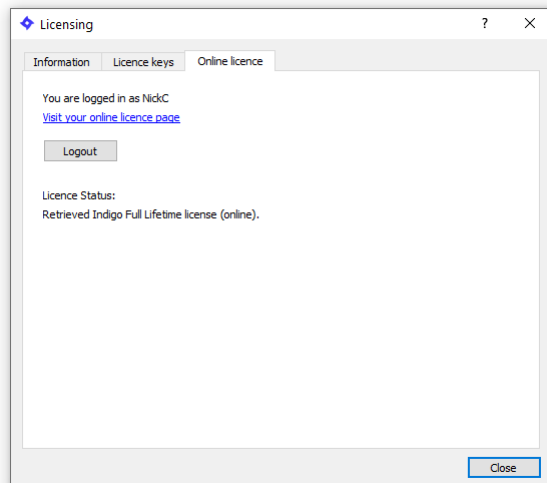
Help > Licensing

select the 'Online licence' tab, and enter your username and password you used to create your account on the store page.



After that it should automatically fetch the needed licence from our server, which will take a few seconds.

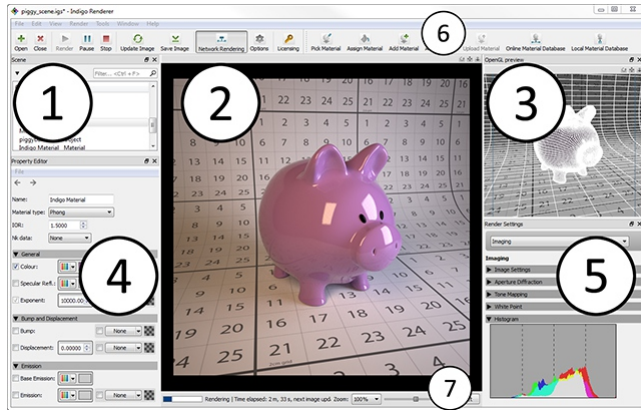
If your Indigo licence was fetched successfully, a message should be shown:



## Indigo Interface

### A first look at Indigo

We'll begin with a "fully loaded" view of the main Indigo interface, showing almost all the available views while rendering a scene:



If you find this is displaying too much information at once, many of the views can be closed independently (and re-opened via the Window menu). Also note that this screenshot was taken on the Windows platform; other platforms may look slightly different.

We'll cover the numbered regions sequentially:

#### 1. Scene view

Displays a list of the scene elements such as models, materials and renderer objects such as the camera, tone mapping and background settings.

#### 2. Image view

This is where the rendering or rendered image is displayed. You can zoom and pan the image using the scrollwheel and by dragging the mouse, respectively. To view the image in full screen, go to View - Full screen, or hit Alt-Enter. The escape key will close the full screen mode.

#### 3. OpenGL preview

Displays a simplified view of the scene using OpenGL, which generally provides much quicker visual feedback for real-time changes than the normal rendered view.

#### 4. Property editor

When a scene element is selected in the scene view (Item 1) and it has editable properties, they can be looked at and edited here.

#### 5. Render settings

Contains a number of tabs for setting pertaining to the imaging (tone mapping, white point etc.), rendering modes and light layers, plus sections with diagnostic information.

This view is only visible once a scene has been opened.

### 6. **Toolbar**

Provides buttons for quickly opening and closing scenes, starting and stopping renders, picking and assigning materials, and other commonly used functionality.
















### 7. **Status bar**

The status bar displays information regarding Indigo's current state, i.e. whether it's waiting for a scene to be loaded, loading a scene or currently rendering.


When rendering it displays how long the current job has been running and how long until the next automatic image update, besides information on how many samples per pixel have been taken (a measure of image quality; see [Principles of Physically Based Rendering](#) for more information).

## Toolbar

Many commonly used functions can be accessed from the toolbar for ease of access, and they can also be found in the various program menus.

	Open Scene	Opens a scene in the current window. If a scene is already open, it will be closed (following a prompt).
	Close Scene	Closes the current scene, terminating the current render (following a prompt). Generally you'll want to save your rendered image before closing the scene.
	Render	Start rendering the current scene. Upon pressing this button, the scene will be built and then rendering will start.
	Pause Render	Pauses or resumes the current render, freeing up your computer's CPU for other tasks. You can resume the render at any stage. If you need to run a short, processor-intensive task while rendering, pausing and then resuming the render is effective (Indigo gives itself below normal priority by default, so this is normally not a problem).
	Stop Render	This stops the current render, freeing the used CPU and memory resources for the currently rendering scene. You can still tone-map and save the rendered image after the render has been stopped, but unlike with Pause, you can't easily resume rendering (see <a href="#">Resuming a render</a> if you need to stop and later resume a render).
	Update Image	Indigo only updates the displayed image occasionally, to avoid wasting processing power on image updates which show little visible difference. However, you can at any time press the Update Image button (or the F5 shortcut key) to force an image update.
	Save Image	Once a rendered image is displayed, you can click Save Image to save it to disk in a number of standard formats such as PNG or JPEG, as well as the special Indigo Image (IGI) format for <a href="#">resuming the render</a> later; the PNG format is recommended since it doesn't degrade the image quality as JPEG does.
	Network Rendering	Enables network rendering. See the <a href="#">Network Rendering</a> section for more information.
	Options	Opens the options dialog. See the <a href="#">Options Dialog</a> section for more information.
	Licensing	Use this button to open the Licensing window. You can use this to buy an Indigo licence, immediately removing the restrictions of the free version. See the <a href="#">Licensing</a> section for more information.
	Pick Material	This tool allows you to select a material by clicking on an object in the image view, whose currently applied material appears in the property editor.
	Assign Material	This tool allows you to apply the currently selected material (in the scene view) to an object in the image view by clicking on it.
	Add Material	Creates a new material in the scene view.
	Add Medium	Creates a new medium in the scene view.
	Upload Material	When editing a material, it uploads the material applied to the preview object together with the currently rendered preview image. The preview must be sufficiently clean (at least 200 samples per pixel) otherwise the upload will not succeed.



 Online Material Database	Accesses the Indigo online material database, allowing you to download materials into your scene and the local material database.
--	---

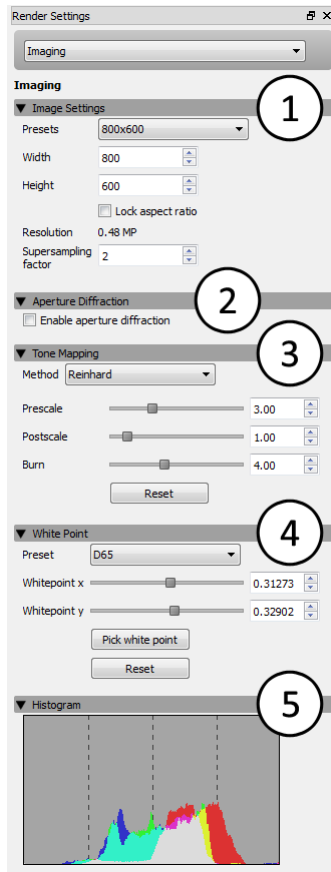
# Render Settings

The render settings are only visible when a scene is open. There are several sub-sections selectable from the drop down control; these are documented in the following manual sections.

## Imaging

"Imaging" collectively refers to the process of converting Indigo's internal physical light data into a normal RGB image which can be displayed on a computer screen.

The following image shows the fully expanded Imaging view; we'll cover with numbered sections sequentially:



### 1. Image Settings

Here you can set the width and height of the image Indigo will render, along with the image super-sampling factor. The aspect ratio (ratio of image width to height) can be kept constant while editing either field by checking the "Lock aspect ratio" option.

If the super-sampling factor is greater than 1, then the image is rendered at a higher resolution internally and down-sampled before the render is displayed on-screen or saved to disk. This can help to reduce aliasing around high contrast edges and produce a sharper final image. Note that higher factors require a lot more memory (RAM), which scales by the square of the super-sampling factor: with a factor of 2, it will use 4x as much memory, with a factor of 3 it will use 9x as much, etc.

### 2. Aperture Diffraction

Aperture diffraction causes small bright light sources to "bloom" (diffract) through the simulated camera's aperture, creating a distinctive rainbow-coloured glow.

The exact shape of the diffraction pattern depends on the scene's aperture shape and obstacle map. For more information please see the [camera documentation](#).

### 3. Tone Mapping

Tone mapping is the process whereby the high dynamic range (HDR) image internally stored by Indigo is converted to a low dynamic range (LDR) image for display on a normal computer screen.

This form of range compression is necessary because in real life, the sun is many thousands of times brighter than a dimly lit room, however on a standard computer screen we can only perceive approximately 200 brightness levels.

- Reinhard: The default "auto-exposure" mode, which handles images with very high dynamic range well, but can have lower contrast than the other modes.
- Camera: This mode allows you to select from a number of pre-defined camera response profiles, which accurately model the response of the camera's optics for various manufacturers and camera models.
- Linear: This mode is the simplest of all, providing only a linear scaling (or gain) factor and providing no dynamic range compression.
- Filmic: Like Linear, this tone mapper has only one control, but it responds quite differently with significantly less contrast at high brightness (Scale) values.

### 4. White Point

A white point (aka "reference white") is a colour which serves to define what "white" should look like in image. This is related to the human vision system's adaptation to different colour temperatures, and allows you to change how "warm" or "cold" an image appears.

### 5. Histogram

This section shows a histogram of the colours present in the image, which is useful to analyse the imaging settings for over-exposure or under-exposure.

## Render Settings

### 1. Mode Configuration

Render mode: This drop-down box allows you to select the render mode with which you'd like to render the scene. There are four [normal rendering modes](#) and two rendering modes aimed at compositing applications: alpha and material ID rendering.

If you are unsure which rendering mode to use a safe default is bidirectional path tracing, and there is also a [render mode guide](#) to help you choose.

Foreground alpha: This render mode sets background pixels to transparent (alpha zero). See [Foreground Alpha](#) for more details.

Halt time: Specifies the number of seconds, from the beginning of the render, until rendering is halted (instead of the usual unbounded rendering time).

Halt SPP: Specifies the number of samples per pixel (SPP) at which rendering is halted (instead of the usual unbounded number of samples per pixel).

### 2. OpenCL (CPU and GPU) Rendering

This section is only visible if a supported GPU (or CPU device, if CPU devices are enabled) was detected. Please see the [System Requirements](#) page for more information on GPU acceleration requirements.

Enable OpenCL Rendering: This checkbox specifies whether OpenCL should be used for rendering. Note that Indigo currently does not support bidirectional path tracing with OpenCL rendering, so enabling this will temporarily force single-directional path tracing.

Device Selection: This control allows you to select one or more devices to use for OpenCL rendering.

When a device is selected, some additional information about the device is listed below.

Max path depth: This sets the maximum number of bounces for a path. Setting this to a smaller number will speed up the rendering, but may result in complicated glass geometry rendering incorrectly. Likewise, increasing this setting will slow down the rendering.

### 3. Metadata

This section displays assorted information about the currently loaded scene.

## Foreground Alpha

*Foreground alpha* is a rendering mode that sets the pixel alpha (opacity) based on if the pixel is considered to be in the scene foreground or background.

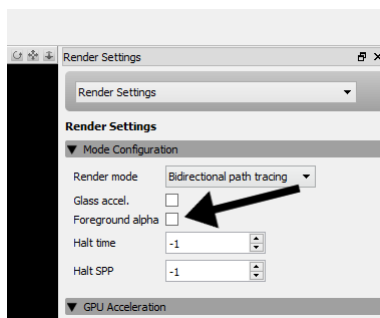
If you enable foreground alpha, then when you save your image to PNG or EXR, or any format that supports an alpha channel, then transparency information will be included in the saved image. This allows for easy compositing of the objects in your scene over a background image.

For an example, see the [Compositing with shadow pass tutorial](#).

Here is a render made with *foreground alpha* enabled. The grey checkerboard pattern indicates alpha zero (fully transparent) pixels.



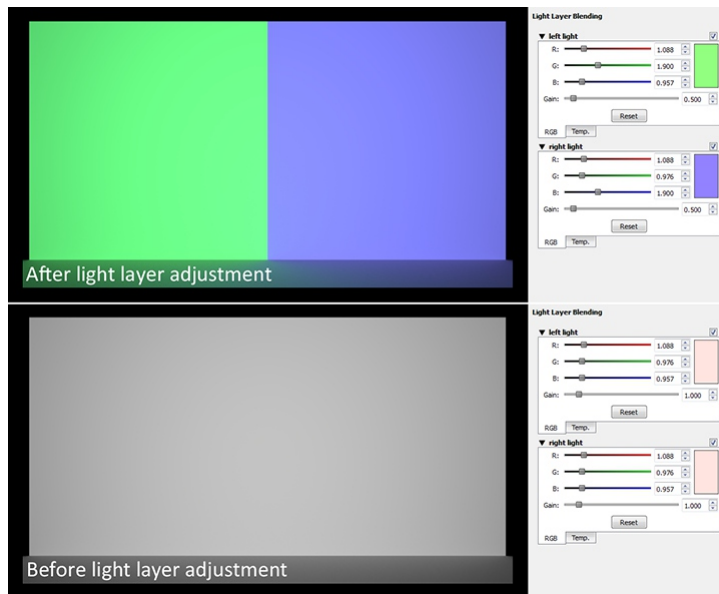
The foreground alpha option is found in the render settings section:



## Light Layers

The controls in this section allow you to view and modify the light layers present in the scene, and their contributions to the final image.

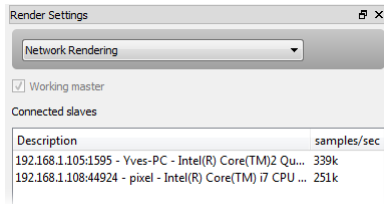
For each light layer present in the scene, a corresponding set of controls is available allowing you to adjust the overall gain and colour tinting, either via a standard RGB colour or using a colour temperature.



Example of light layer interface (click to enlarge). [Click here](#) to download this example scene.

## Network Rendering

The network rendering section of the Render Settings view lists all the connected slaves, and allows you to toggle whether or not the master should render as well.





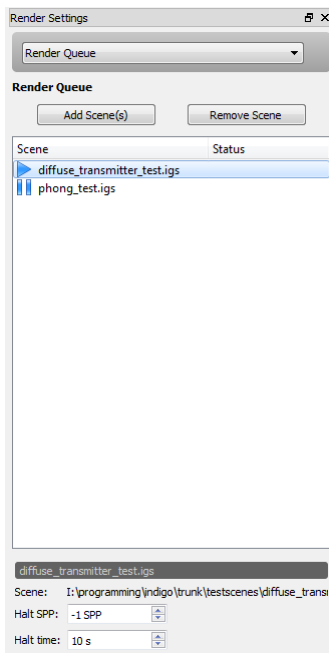
# Render Channels

See [Render Channels](#).

## Render Queue

The render queue lists the current batch of scenes to be rendered.

When loading a scene normally from inside the Indigo GUI, only a single item is added to the render queue, however multiple scenes can be added, with a halting condition (on rendering time, samples per pixel or both) to specify how long they should render for.



A Render Queue can be saved by clicking File -> Save Queue. A Indigo Queue file is automatically created when exporting animations from Indigo's 3D modelling package plugins. [More information about saving and using Render Queues.](#)

### Render Log

While rendering, the Indigo core will output diagnostic information about the scene and the render status. This can be useful in diagnosing problems, and including a render log with an error report is always welcome.

## Resuming a Render

While Indigo is rendering a scene, you can save the HDR buffer into an Indigo Image (file extension .IGI) to resume rendering later.

The Indigo Image stores all the information about a render in progress, so if the same scene is loaded again later (even after the rendering has been stopped and Indigo closed), the current rendering state can be fully restored using the IGI file.

To resume rendering a scene using a saved IGI, follow these steps:

### Saving the render progress to an IGI

1. Start rendering the scene.
2. In the menu under "Render" click "Save IGI for resuming"
3. Select the Indigo Image file type in the Save File dialog.
4. Choose a file name for your IGI file, for example "test.igi", and click Save.
5. You can now close Indigo.

### Resuming

1. When you wish to resume your render, open Indigo, and select "Resume Render from IGI" from the File menu.
2. You will be prompted to locate the IGI file. Navigate to where you saved the IGI file, and press the "Open" button after selecting it.
3. Indigo will read the location of the original scene file from the IGI. If it is no longer present, you will then be prompted to locate the original scene file, which the saved IGI was made with.

The IGI doesn't store information about the actual scene to be rendered (i.e. models and textures), only about the state of rendering it, so the scene file is still necessary to carry on rendering. You must open the same scene that you were rendering when you saved the IGI.

4. You can confirm that the render resumed successfully by looking at the status bar – the time elapsed should include all the time spent rendering before the IGI was saved.

## Indigo File Types

Indigo has several of its own file types for use specifically with Indigo.

IGM Indigo Material. Contains material information, but nothing else.

PIGM Packed Indigo Material. Contains material information and anything else relevant to the material, such as textures. Can be unzipped with compression programs such as 7-Zip. This is the preferred format for distributing Indigo materials.

IGI Indigo Image. Contains information saved from an Indigo render. Used for [resuming renders](#).

IGS Indigo Scene. Contains information about an Indigo scene saved on disk.

PIGS Packed Indigo Scene. A self-contained archive with everything needed to render a scene, including all referenced models and textures etc. Can be unzipped with compression programs such as 7-Zip. This is the preferred format for distributing Indigo scenes.

IGQ Indigo Queue format. Lets the user render multiple scenes after each other. The rendering process and halt settings of the different frames can be controlled in the Render Queue window. Used for [rendering sequences and animations](#).

PIGQ Packed Indigo Queue. A self-contained archive with everything needed to render a sequence of scenes, including all referenced scenes, models and textures etc. Can be unzipped with compression programs such as 7-Zip. This is the preferred format for distributing Indigo animations.

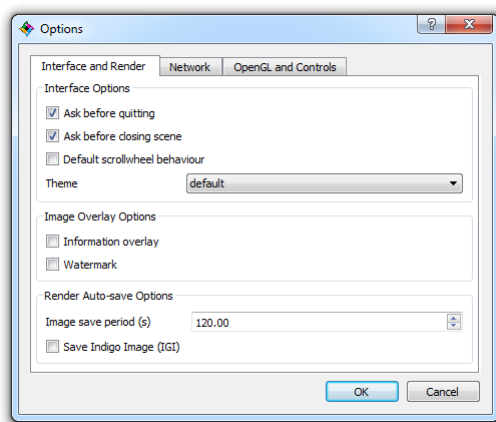
# Options Dialog

## Options Dialog

The options dialog holds settings for the Indigo user interface and rendering options, networking configuration and OpenGL / input controls.

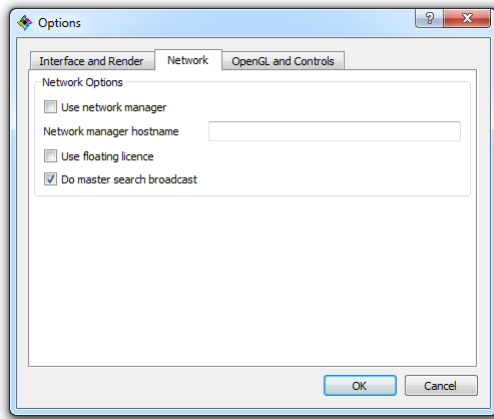
We'll cover these sections sequentially, corresponding to tabs in the options dialog:

### 1. Interface and Render



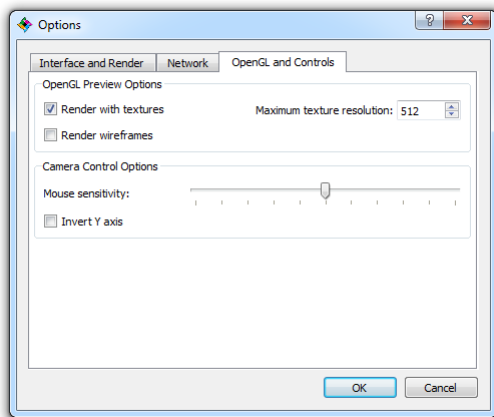
- Ask before quitting - Prompts the user for confirmation before exiting Indigo.
- Ask before closing scene - Prompts the user for confirmation before closing scenes.
- Default scrollwheel behaviour - By default, when scrolling the mouse wheel over a control which can itself scroll, it will scroll the contents instead of the parent window. When this option is unchecked, it instead scrolls over the control in the parent window without transferring focus.
- Theme - Changes the appearance of the Indigo user interface according to a selection of pre-defined themes.
- Information overlay - Displays information about the render in a small black rectangle in the bottom left corner of the image.
- Watermark - Displays an Indigo watermark in the bottom right corner of the image. Cannot be disabled in trial / unlicensed mode.
- Image save period - Automatically saves the image periodically, according to how many seconds are specified in the input field.
- Save Indigo Image - When automatically saving images, this option toggles whether or not a full Indigo Image is saved (rather than only a normal PNG).

### 2. Network



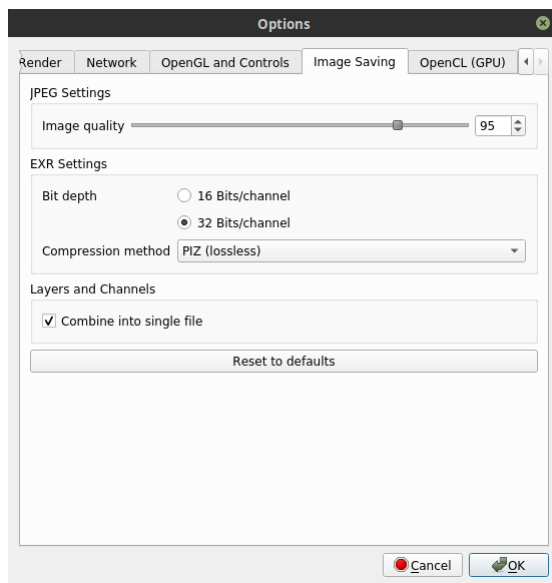
- Use network manager - Specifies that Indigo's network rendering should be coordinated by a [Network Manager](#) instead of running independently.
- Network manager hostname - The network computer name for the network manager; this cannot be "localhost" or "127.0.0.1" since the name is passed on to other computers, which will incorrectly try to connect to themselves, instead of the particular computer on which it was specified. Other computer names or IPs are valid.
- Use floating licence - Specifies that Indigo's licensing system should attempt to use a [network floating licence](#) instead of the normal per-computer licensing system.
- Do master search broadcast - If this option is enabled, Indigo will periodically search the local area network (LAN) for Indigo masters wanting rendering slaves.

### 3. OpenGL and Controls



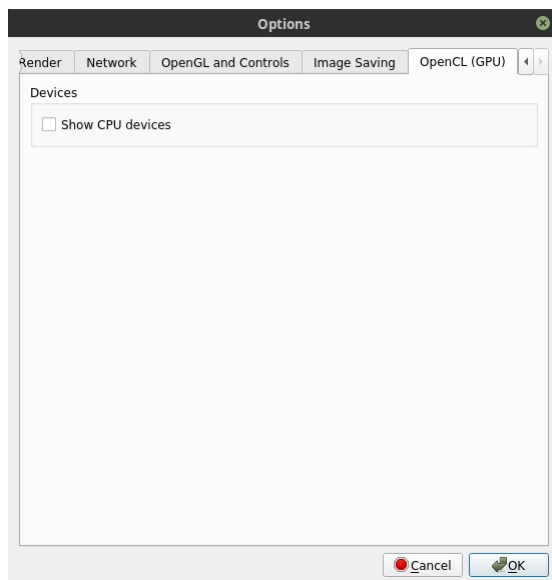
- Render with textures - Specifies whether the OpenGL preview should use texture maps from the scene. This uses extra GPU memory and increases scene loading time.
- Maximum texture resolution - Specifies the maximum width or height of textures used in the OpenGL preview. This is useful for reducing the amount of additional memory used by the OpenGL preview.
- Render wireframes - When enabled, the OpenGL preview shows the scene polygons using wireframe rendering, which can be useful to see the geometric detail in a model.
- Mouse sensitivity - The slider position specifies how fast the camera will rotate, pan or zoom when in real-time mode; further to the left means less sensitive, further to the right means more sensitive.
- Invert Y axis - If this option is enabled, mouse movement on the vertical (Y) axis will be inverted, i.e. moving the mouse up will make the camera look down.

## 4. Image Saving



- JPEG Image Quality - Controls the quality setting for JPEG images saved by Indigo.
- EXR Bit Depth - Allows you to set the bit depth of saved EXR files.
- EXR Compression method - Choose the compression method used by when saving EXR files.
- Combine Layers and Channels into single file - When enabled, layers and channels are combined into a single EXR file, when disabled, Layers and Channels are saved separately to files suffixed with the layer/channel name.

## 5. OpenCL (GPU)



- Show CPU devices - When enabled, CPU devices are listed and available for OpenCL rendering.




## Realtime camera control

Indigo's ability to dynamically edit scene data allows for realtime camera movement, which is very useful for final tweaking of a shot, or simply exploring the scene with full lighting.


There are two ways to effect realtime camera changes: via the on-screen buttons, and via keyboard and mouse controls.

### Using the on-screen buttons

To use the on-screen buttons, click and drag on the buttons in the top-right corner of the main render and OpenGL preview windows:

 **Rotate** Rotates the camera's view direction.

 **Pan** Pans the camera.

 **Dolly** Moves the camera forwards and backwards along the view direction.

### Using the keyboard and mouse

In the main render or OpenGL preview window:

Hold Alt + left mouse button, drag      **Rotate** Rotates the camera's view direction.

Hold Alt + middle mouse button, drag      **Pan** Pans the camera.

Hold Alt + right mouse button, drag      **Dolly** Moves the camera forwards and backwards along the  
up/down view direction.

Holding the Shift key while performing these actions reduces the mouse sensitivity, for more fine-grained control.

Mouse sensitivity and inversion options can found in the Options dialog, in the "OpenGL and Controls" tab.

## Network Rendering

Indigo has built-in support for network rendering, which allows all the computers on a network to work together to render a single Indigo scene more rapidly.

You will need one master computer, that will coordinate the rendering process. Please note that it is not possible to start or coordinate the rendering process from a slave computer (registered with a node licence).

You will also need one or more slave computers, that will be helping to render the scene. For the purposes of this tutorial, the slave computers must be on the same local area network (LAN), and able to communicate with each other.

### On the Master computer

1. Launch the Indigo application.
2. Open the scene that you want to render in the Indigo Renderer GUI using the "Open Scene" button.
3. Press the "Network Rendering" button to enable network rendering mode.
4. Press the start render button.

### On the Slave computer

- Run the Indigo Network Render Slave.  
The network slave can be launched by the shortcut (Windows), the Indigo Networks Slave app (OSX), or by running `indigo` or `indigo_console` executables with the command line switch `"-n s"`:

```
./indigo -n s
```

If the network slave can find the master and connect to it on the network, it will then download the scene from the master and start rendering it.

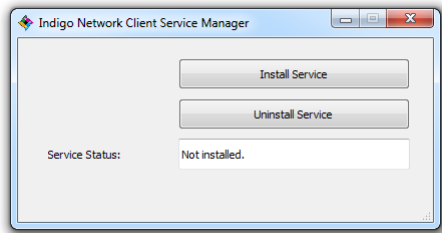
### Viewing connected slaves on the master GUI

You can view all network slaves that are helping with the current render by selecting "Network Rendering" from the combo box on the right hand edge of the Master GUI.

## Automatic Slave

Indigo for Windows has the ability to install itself as a system service, which enables your computer to run a Network Slave during screensaver. This is useful for utilising idle office workstations to accelerate network renders.

To enable the Windows service, navigate to your Indigo installation directory and run the "network\_client\_service\_manager.exe" application, and click the "Install Service" button.



Uninstalling the service can also be done from this dialog, by clicking the "Uninstall Service" button.

Once the service is installed, when the computer goes into screensaver mode, it will start an Indigo Slave and contribute to any active network renders. When the computer exits the screensaver, the rendering is halted to free up the computer for use.

## Network Manager

The Network Manager does two things:

### 1. Floating Licences

If you have purchased floating Indigo licences, the Network Manager hands out floating licences to computers on your network. However, you don't require floating licences to use the Network Manager for network rendering coordination.

To purchase Indigo floating licences, email us at [sales@indigorenderer.com](mailto:sales@indigorenderer.com)

### 2. Managing Network Rendering

Indigo supports network rendering, which means that additional computers (slaves) can help other computers (masters) render an Indigo scene.

The Network Manager can control and coordinate this network rendering, by assigning slaves to masters. This is ideal for an office or render farm situation where there are multiple masters and multiple slaves on the same network.

## Network Rendering Tutorial

This tutorial will cover network rendering with the Indigo Network Manager.

### Set up the network manager

#### 1. Choose the manager computer

Choose a computer to run the network manager on. Ideally this computer would be running at all times, essentially acting as a server; we will call this computer the "Manager computer".

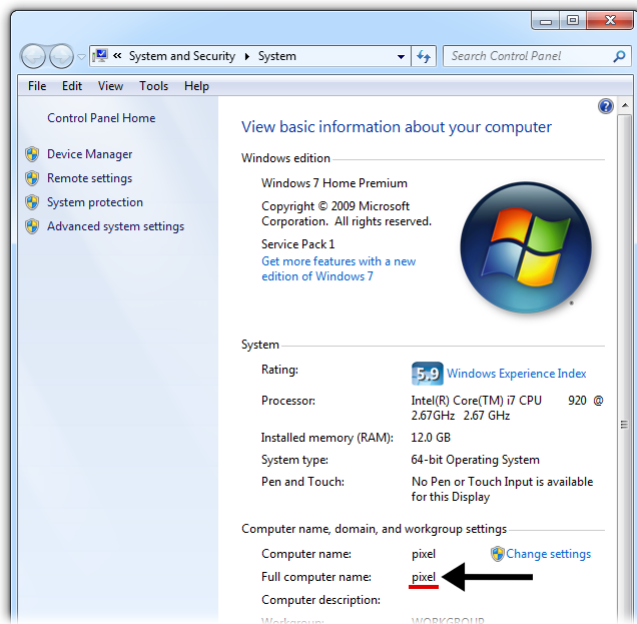
#### 2. Install Indigo on the Manager computer

Install Indigo on the Manager computer. For detailed information about this step please see the [Installing Indigo](#) section.

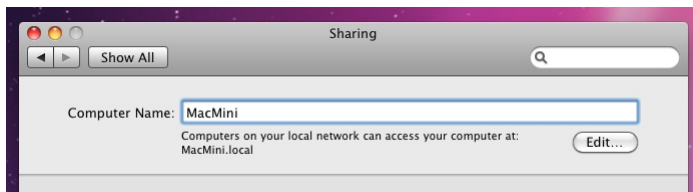
#### 3. Obtain the Manager computer's hostname

A hostname is a name used to identify a device connected to a computer network. To find the hostname of the Manager computer:

On Windows: From the Start menu, right click on "Computer" and select "Properties". The hostname is listed as the "Full computer name", which is "pixel" in this example:



On Mac: The hostname is listed as "Computer Name" under System Properties → Sharing:



#### 4. Run the network manager

Run the Network Manager on the manager computer. If the operating system prompts for permission to allow incoming connections, answer "Yes". Without this permission, other nodes will not be able to communicate with the Manager computer.

On Windows: Start → All Programs → Indigo Renderer → Indigo Network Manager

On Mac: Finder → Applications → Indigo.app → Indigo Network Manager

#### 5. Running a slave

On another computer that has Indigo installed:

1. Start Indigo and click on the "Options" button.
2. In the Network tab, make sure the "Use network manager" option is checked.
3. In the "Network manager hostname" field, enter the Manager computer hostname.
4. Ensure that the "Do master search broadcast" option is unchecked.
5. Click "OK" to save the changes and exit the options dialog.
6. From the tools menu, select "Start Network Slave".

Leave the network slave running for now.

#### 6. Running the master

On the computer that you wish to use as the master computer, e.g. the computer that you will be starting a render from, start Indigo:

Start → All Programs → Indigo Renderer → Indigo Renderer

1. Click on the "Options" button.
2. Ensure that the "Use network manager" option is checked.
3. In the "Network manager hostname" field, enter the Manager computer hostname.
4. Ensure that the "Do master search broadcast" option is unchecked.
5. Click "OK" to save the changes and exit the options dialog.

Warning: When setting up the master, using "localhost" or "127.0.0.1" as the hostname for the network manager will prevent other slaves from connecting to the master.

This is due to the fact that the master will then connect to the network manager through the loopback interface and the network manager will pass the IP from which the master connected on to the slaves. For loopback, this is always 127.0.0.1.

#### 7. Start the render

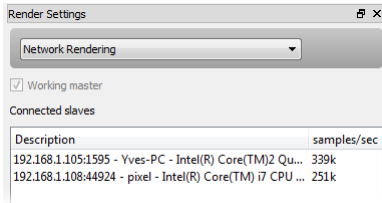
On the master computer, open a scene in Indigo that you wish to render, and press the "Render" button.

Now click the "Network Rendering" button to enable network rendering.

### Verifying network rendering is working from the Master

On the Master computer, select "Network Rendering" from the drop down box in the Render Settings view.

If the network rendering configuration is correctly set up, there should be a client listed (with IP address and the hostname) in the "Connected slaves" list:



It will also show the rendering speed in samples per second for each slave. Note that this speed is not known until the first frame is transferred from the slave to the master, and so will show "Unknown" initially.

### Verifying network rendering is working from the Network Manager

The Network Manager should show one slave in the "Slaves" list, and one master in the "Masters" list. Additionally, the "Assigned master ID" for the slave should be the ID of the master. This means that the Network Manager has assigned the slave to the master.

# Rendering with Indigo

This section begins with a simple overview of rendering with Indigo, followed by sections for various features and settings used for rendering.

Comprehensive coverage of Indigo's material system can be found in the "Materials" section.



## Render Tutorial

In this tutorial we will render an example scene that comes with Indigo to illustrate the basic render settings.

1. Start Indigo and click the Open button in the toolbar. Browse to the "testscenes" subdirectory in your Indigo Renderer installation, which on Windows is usually C:\Program Files\Indigo Renderer\testscenes.

Open the "Caterpillar" example; this scene, by Paha Shabanov, showcases a number of Indigo's more complex features (e.g. displacement, subsurface scattering) and won a competition held on our Forum to be included with the Indigo distribution.

2. This scene renders quite slowly using the default Bi-directional Path Tracing render mode, so let's set it to use the simpler, non-Bi-directional (i.e. single-directional) path tracing mode.

In the Render Settings view (on the right side of the image), from the drop-down menu at the top, select the "Render Settings". From the "Render mode" drop-down, select "Path tracing".

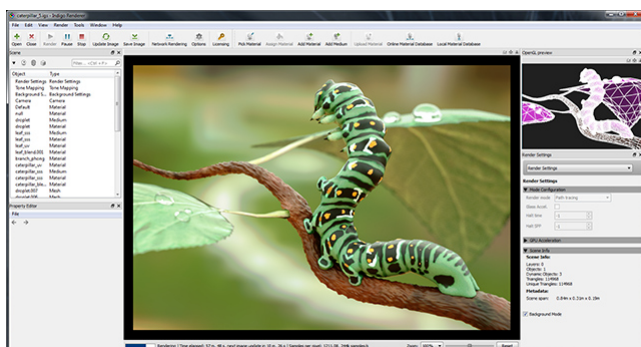
3. Now we're ready to begin rendering. Hit the "Render" button on the toolbar, and Indigo will begin "building" the scene (preparing it for rendering).

For simple scenes, this build process will be nearly instantaneous, but for larger scenes (with many polygons, subdivision surfaces etc.), building the scene can take a little while. Indigo displays the build progress in the status bar, and you can see the full log by clicking the "Render Log" drop-down option from the Render Settings view.

4. Once the scene has started rendering, the status bar will continually update with information about the render in progress.

Particularly relevant is the number of samples per pixel, which can be roughly thought of as the image quality; every so often (with decreasing frequency) the image will automatically update as it's rendering. You can update the image at any time either via the Update Image toolbar button or by pressing F5.

5. After some minutes of rendering the "noise" (or graininess) in the image will go away, leaving a nice clean render:



6. Next we'll illustrate some of the imaging settings; these affect the appearance of the final image from the physical light computation Indigo performs, and can be adjusted without restarting the render.

In the Render Settings view, select "Imaging" from the drop-down at the top. The default setting for this scene is Camera tone mapping with the FP2900Z preset, and we can change its exposure (EV) and film ISO as with a real camera. If we switch the method to "Reinhard", Prescale to 2 and Burn to

3.6, we get the following result with less saturation, but also less "blow out" in the bright regions:



7. In the White Point section we see that the scene's default white point is the "E" preset. Typically we use a D65 ("daylight") white point, and selecting this option produces a noticeably "warmer" image:

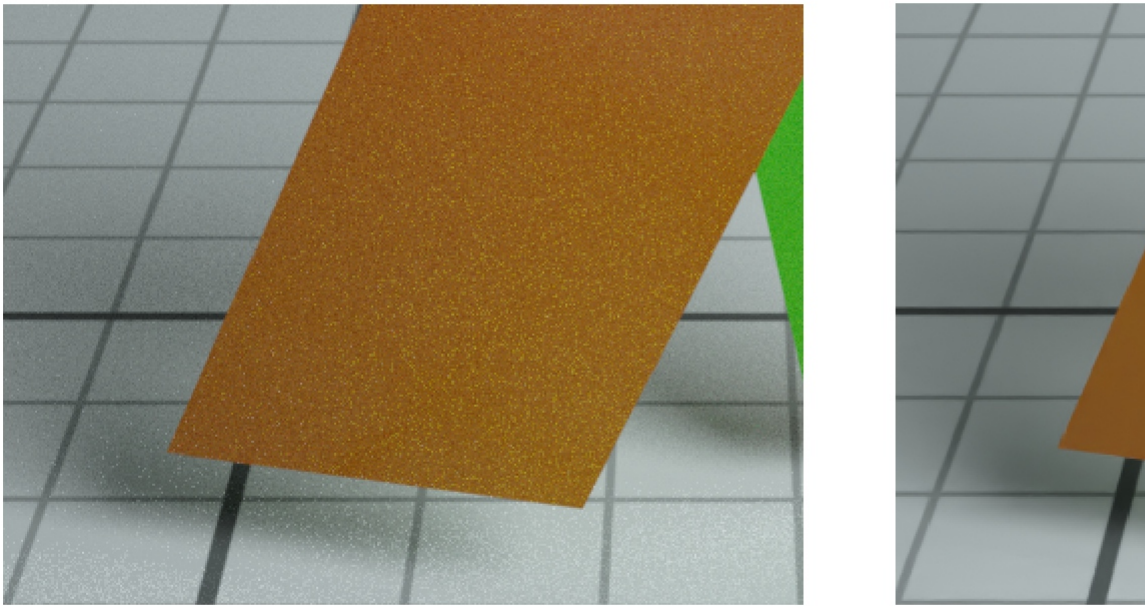


8. Finally, let's save this image to disk; click the "Save Image" button in the toolbar, and either give the image a new file name or leave it as is.

Saving as PNG is generally recommended instead of JPEG unless the image will be directly uploaded to the Internet and needs to be compressed, since every time a JPEG image is saved the image quality is reduced (as it is a "lossy" format, as opposed to PNG which is "lossless" i.e. a perfect copy).

## Denoising

Since Indigo 4.4.1 beta, Indigo has integrated Intel's Open Image Denoise.



Indigo output (132 s/px)

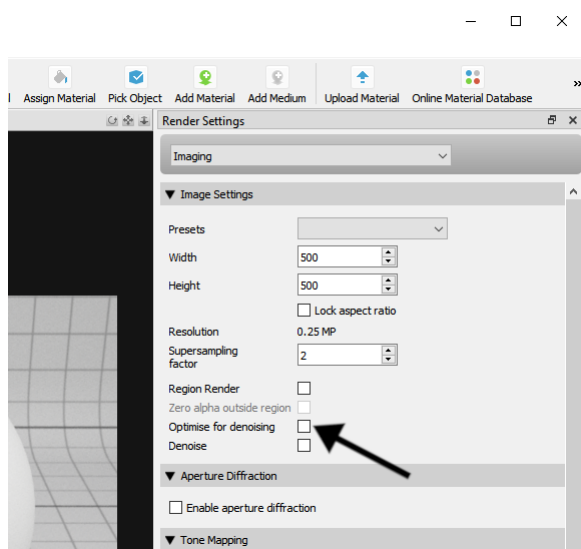
Denoising currently only works at full effectiveness with OpenCL (GPU) rendering.

Denoising currently works with greatly reduced effectiveness for CPU rendering. This will be improved in future Indigo versions.

### Optimising for denoising

The denoiser requires some additional information to work at maximum effectiveness - in particular it requires the [normals and albedo render channels](#) to be enabled.

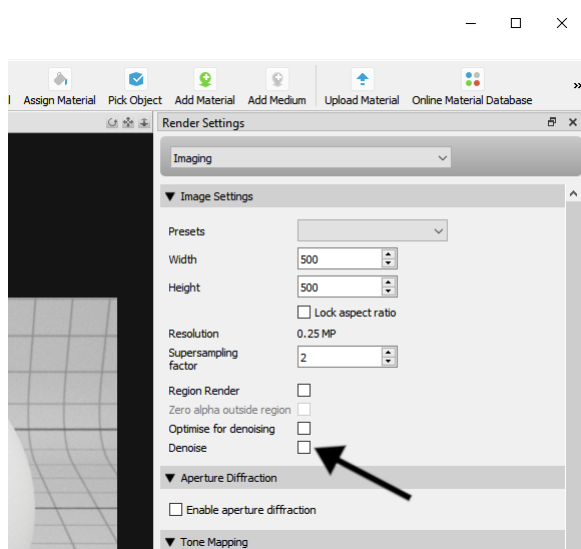
This can easily be accomplished by checking the 'Optimise for Denoising' checkbox:



Checking or unchecking this checkbox will restart the render.

## Enabling or Disabling denoising

Denoising can be enabled or disabled during rendering with the 'Denoise' checkbox. Checking or unchecking this checkbox won't restart the render.



Denoising is computed each time the image is updated and displayed.

Denoising takes some time to compute - especially for high resolution images, and high supersampling factors.

If it is taking too long, try reducing the supersampling factor (for example to 2 or 1).

## More denoising examples



A render without denoising. The room scene by Lal-O is included in the Indigo distribution.





The render with denoising.

## Environment Settings

There are several options that allow you to define the appearance of empty space around your scene. This listing is not exhaustive, since any material can be used as the background material, however we'll most list the commonly used settings here.

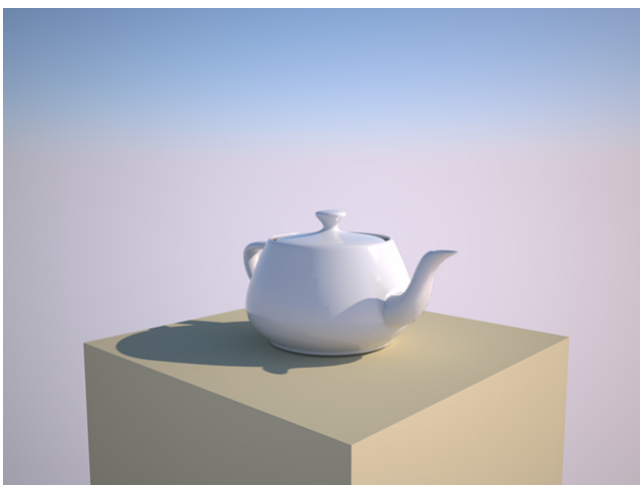
### Constant colour background



Illuminates scene with a uniform environment light.

### Sun & sky

Indigo comes with a Sun and Sky environment that realistically depicts the sky. Changing the sun's direction creates time-of-day effects: a low angle creates a sunrise/sunset with correctly coloured sky and brightness.



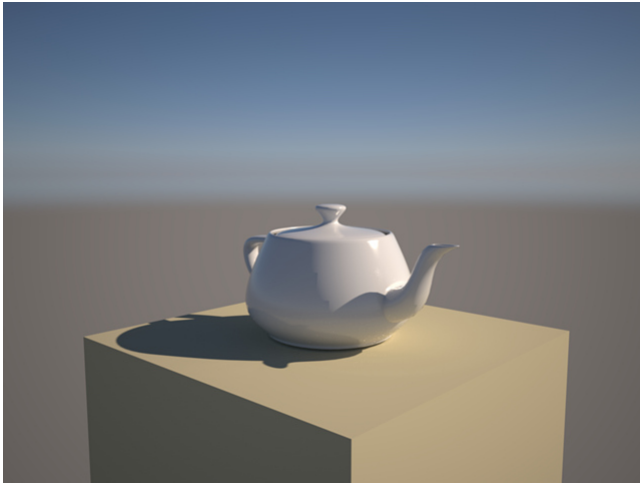
The classic Sun & Sky model.

The following options are available for the classic model:

**Turbidity:** The turbidity defines the haziness/clearness of the sky. Lower turbidity means a clearer sky. Should be set to something between 2 and ~5.

Extra Atmospheric: Removes the sky and renders only the sun. Good for renders in space.

Since Indigo 3.2, there is also a new "captured" sky model, which is actually simulated by Indigo and captured to disk with the distribution.



The captured Sun & Sky model.

### Environment map



Illuminates scene with an environment map, which usually is a high dynamic range (HDR) image.

Indigo can load HDR environment maps in three formats, EXR (file extension .exr), RGBE (file extension .hdr) and a raw data format (extension .float, a simple format exported by the HDR Shop program); the environment map must either be in spherical format or equirectangular.

There are a number of settings available currently for environment maps. The emission values should usually be quite high, approximately  $10^7$  to be similar to the sun's brightness. The "Advanced Mode" checkbox ticked gives the users more options (full control over the image including Gamma and Texture Mapping modes), while disabling it reveals easy to use controls for rotating and tinting the environment map.



## GPU Rendering Guide

### GPU drivers

The first and most important point is: you must update your GPU drivers.

GPU compute depends very strongly on the quality of GPU drivers, and the various GPU manufacturers have been doing a great job of updating their drivers to be faster and more robust.

NVIDIA driver downloads: <http://www.nvidia.com/drivers>

AMD driver downloads: <http://support.amd.com/en-us/download>

Intel driver downloads: <https://downloadcenter.intel.com/>

### Recommended GPU specifications

#### NVIDIA

GTX 660 Ti or better

Quadro K4000 or better

#### AMD

Radeon HD 7770 or better

FirePro W5000 or better

Older graphics cards might work with OpenCL rendering, but may cause problems and are unlikely to provide any significant performance gain compared to Indigo's highly optimised CPU rendering.

### Memory limitations

One of the major limitations for GPU-based rendering is the amount of onboard memory available, which is typically 2-4 GB for desktop GPUs, while CPUs can easily have 32 GB. Because of this, you might run out of memory when trying to render scenes with lots of geometry and high resolution textures.

The Max Individual Allocation reported by Indigo is the largest amount of memory that can be allocated at once with OpenCL, e.g. for textures. Indigo performs multiple allocations for every scene. The practical limitation at this point is that the total texture size is limited to 25% of total GPU memory on NVIDIA, and about 65% of total GPU memory on AMD.

### Using the computer while rendering

A well-known side effect of using your primary GPUs (ones which are connected to a display) while rendering is that your operating system can lag quite substantially. The best way around this is to have a GPU dedicated for display (usually a small inexpensive card, or perhaps the integrated GPU of some CPUs), while the others are dedicated for rendering. This also helps to reduce the memory overhead, which can be important in GPUs with  $\leq 2$  GB of memory running high resolution desktop and lots of web browser tabs etc.

### Kernel compilation

Indigo uses a combination of OpenCL and our own programming language, Winter, for the rendering core. This means that when you start rendering a scene, the GPU drivers must compile the OpenCL code, and this process can take some time, depending on the complexity of the scene.

When using multiple GPUs, the kernel builds for each GPU are done in parallel to make better use of the CPU cores, however most GPU drivers do not do multi-core compiling themselves, so future driver updates could improve this without any changes in Indigo.

We are working on making kernel builds faster and less frequent.

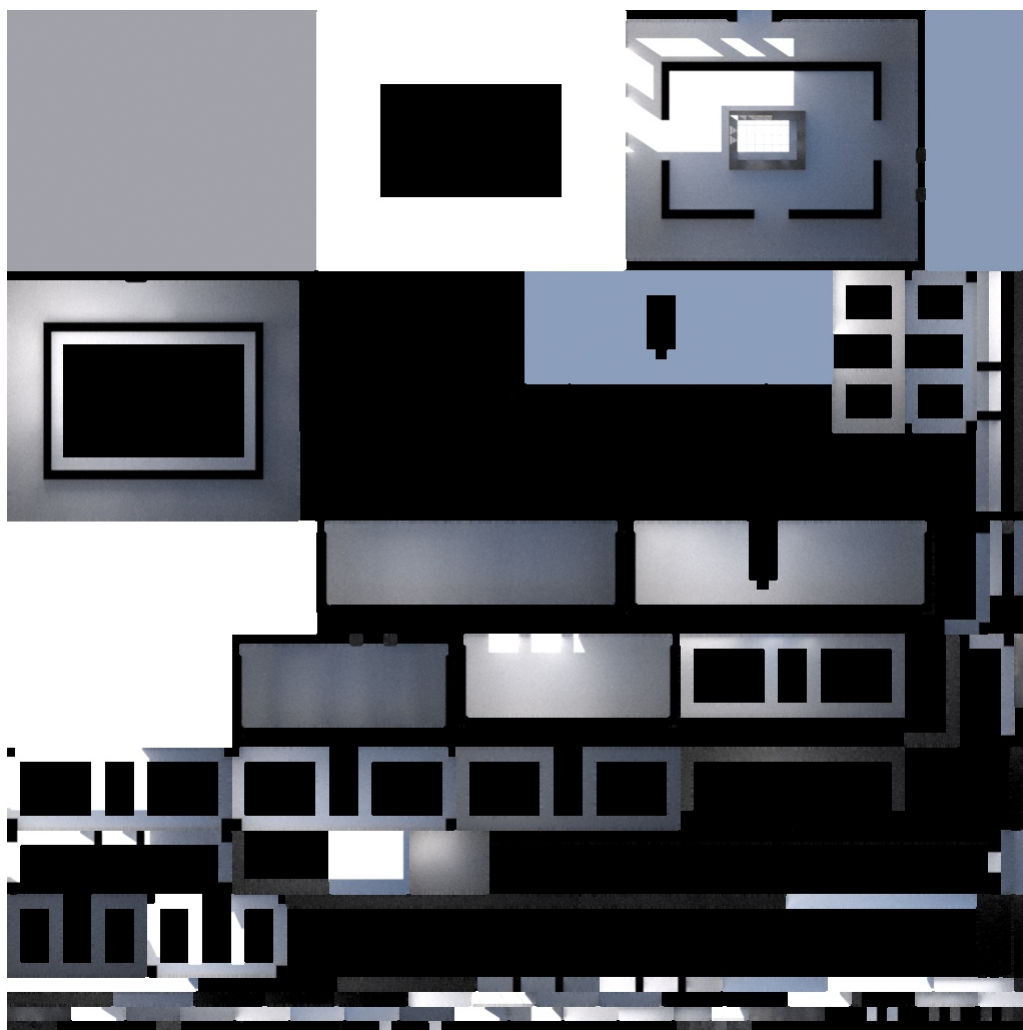
## Lightmap baking

### Lightmap baking

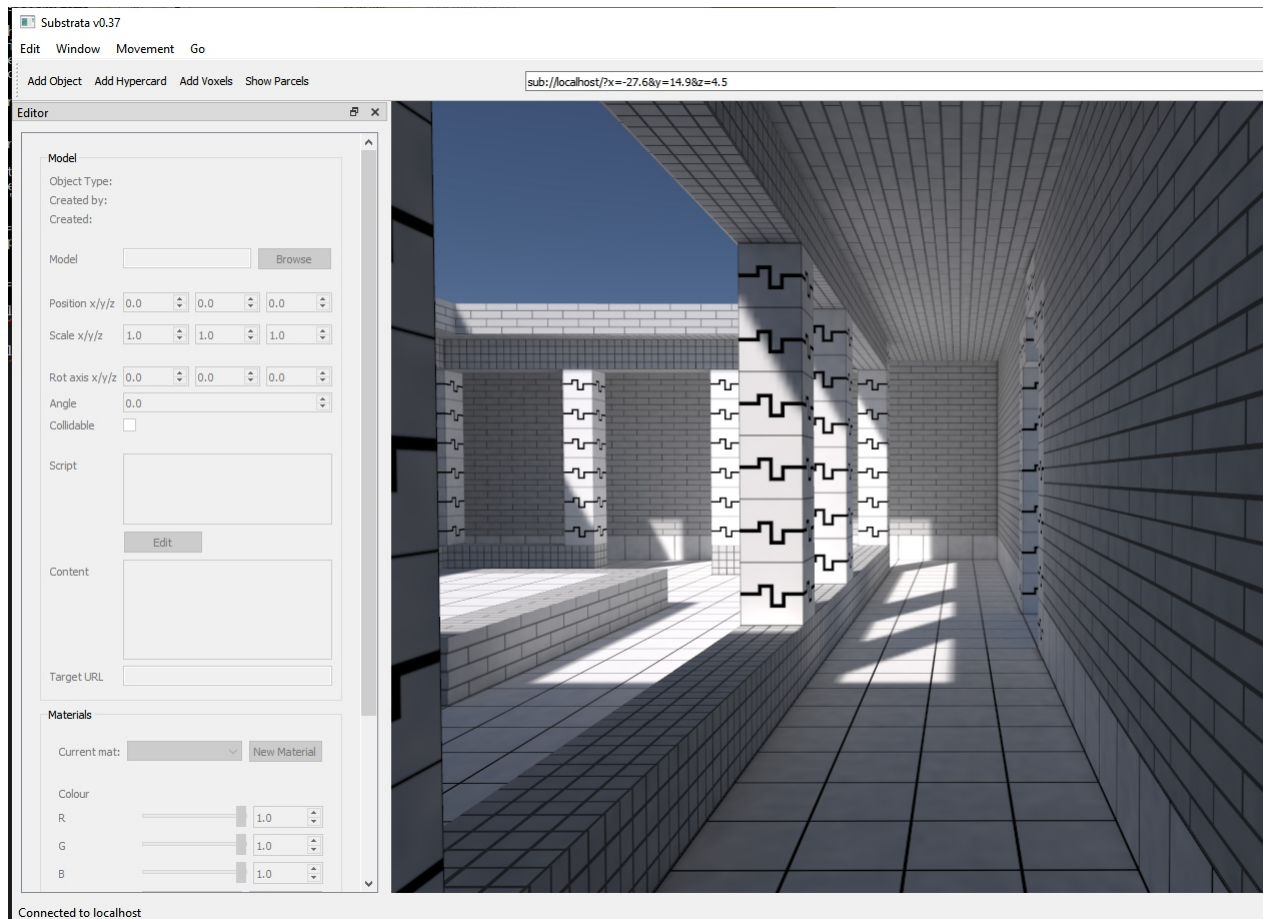
Indigo 4.4.10 introduced a lightmap baking feature.

Lightmap baking is the rendering of a lightmap. A lightmap is an image map that represents the light (or in particular, the irradiance) at points on the surface of an object. Lightmaps are usually used for realtime 3d rendering, for example in computer games or VR experiences.

Lightmap baking in Indigo allows a lightmap to be computed with full unbiased physically-based spectral rendering with global illumination.



A lightmap baked in Indigo



The lightmap used in a realtime OpenGL 3d engine.

## UV unwrapping

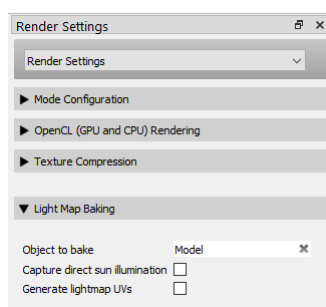
Lightmaps require a special UV mapping, generally in which each piece of geometry appears just one in the UV map, and there are no overlaps. The process of generating such a UV map is called UV unwrapping.

Indigo can optionally perform this UV unwrapping process, or it can use an existing UV mapping present in the mesh data.

## How to use lightmap baking in Indigo

To perform lightmap baking with Indigo, you will need to either enable lightmap baking in the render settings user interface, or alternatively edit your scene file.

To enable in the user interface, select an object in the 'Object to bake' box:



You should check 'Generate lightmap UVs' unless you know your object mesh already has a suitable UV map.

You can alternatively edit your scene file (.igs file, which is just XML), and set the following render settings in your <renderer\_settings> element: (see <https://www.indigorenderer.com/indigo-technical-reference/indigo-scene-f...>)

light\_map\_baking\_ob\_uid

If set to a valid UID ( $\geq 0$ ), Indigo will compute a lightmap for the object with the given UID.

type: int

default value: -1

generate\_lightmap\_uv

If set to true, Indigo will generate a UV mapping for the mesh that it is computing lightmaps for, if any (see light\_map\_baking\_ob\_uid). The new UV mapping will be added to the mesh after the existing UV mappings.

For example, if the mesh has a single UV mapping with index 0, then a new UV mapping will be created with index 1.

The modified mesh with the additional UV mapping will be saved to disk at

'mesh\_with\_lightmap\_uv.igmesh' in the Indigo Renderer application data dir. (e.g.

C:\Users\xx\AppData\Roaming\Indigo Renderer)

If generate\_lightmap\_uv is false, then the UV mapping with the highest index in the mesh will be used as the lightmap UV mapping.

type: boolean

default value: false

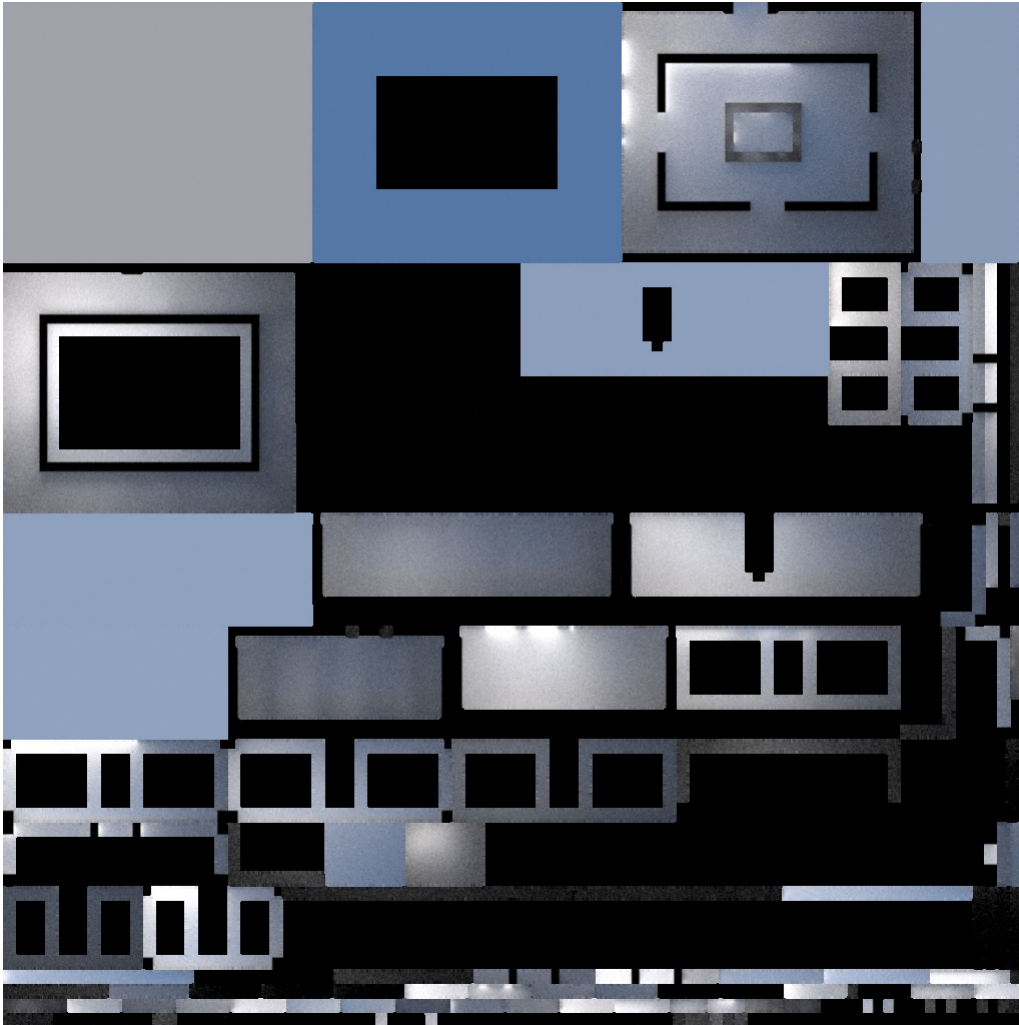
capture\_direct\_sun\_illum

This option only has an effect when computing a lightmap.

If set to false, direct illumination from the sun will not be captured in the light map. This is useful in the case that direct sun illumination will be rendered using some other technique at runtime, for example shadow mapping.

type: boolean

default value: true



A lightmap with `capture_direct_sun_illum` set to false; compare with the lightmap above.

### Saving your lightmap

Indigo renders lightmaps much like usual renders with a simulated camera. As such you can save the lightmap in the standard ways, including with the *Save Image* toolbar button, which saves tonemapped, LDR image. You can also save out a HDR un-tonemapped EXR, with the *Render > Save un-tonemapped image* command.

### Denoising

Indigo's integrated denoiser also works very effectively with lightmaps. You can enable denoising by checking the 'Denoise' button in the Image Settings controls in the Render Settings area.



Lightmap without denoising



Lightmap with denoising (same render time)



# Object Settings

There are some settings that apply to individual objects. (sometimes referred to as instances)

Please read on for details.

## Invisible to Camera

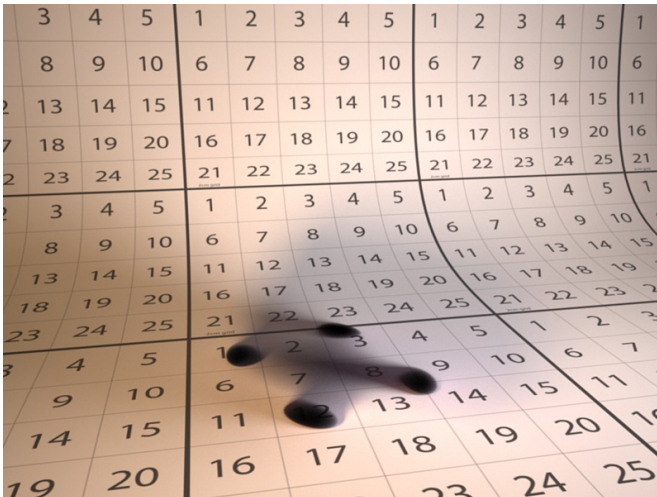
The *invisible to camera* option makes an object invisible to the camera, at least as seen directly. However, the object will still cast shadows, and will be visible in reflections from other objects.

The *invisible to camera* option is useful in a few different scenarios:

- Hiding a wall in an architectural rendering so there is more space for the camera.
- Hiding an object when doing a shadow pass, so that the object does not get in the way of the shadows cast on the ground. See the [Compositing with shadow pass tutorial](#) for more information.
- Hiding a light source, where the light source would usually be visible in the 'shot'.



Piggy bank with piggy visible to the camera



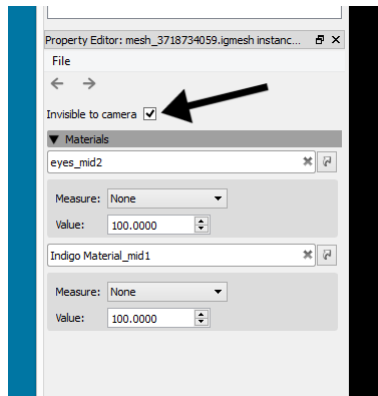
Piggy bank with piggy invisible to the camera. Note that shadows from the piggy are still present.

## Setting to invisible in the Indigo GUI

To make an object invisible to the camera in the Indigo user interface, first select the object with the *pick object* tool:



Then check the *invisible to camera* checkbox in the Property Editor:



The *invisible to camera* option is in Indigo Renderer only and is not available in Indigo RT.

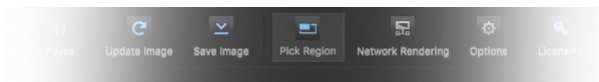
## Region rendering

Region rendering allows you to render only a small part of the scene. This is similar to moving the camera, but is useful when you need to focus the render on a part of the full image.

As of version 4, Indigo lets you render multiple regions sequentially and/or simultaneously. There is also an option to either render the region(s) on top of the full image buffer, or on an alpha background outside the regions for compositing in post production software.

### Using render regions

To use region rendering, simply enable the "Pick Region" button from the tool bar. This will enable Region Render under Imaging settings, and will let you select your region by using LMB + drag. To render multiple regions at once, hold Shift or Ctrl and drag.

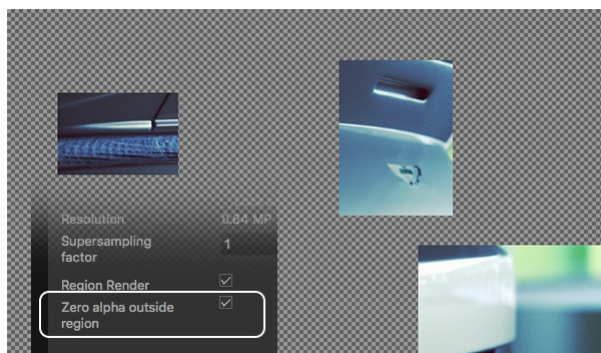


To render your regions on an alpha background, enable the "Zero alpha outside region" checkbox below the Region Render one. This option can be enabled and disabled without re-rendering.

Unchecking the Region Render checkbox will disable region rendering and render the full image buffer. Rechecking Region Render will use your previously used region(s). Camera movement will also disable region rendering for easier maneuvering.



*Multiple regions*



*Zero alpha outside region*

## Render Channels

Render channels allow extra information to be rendered from the scene.

Beauty render channels are components of the final 'beauty' render, where each render channel captures all the light from a particular class of path from the emitter to the camera.

Non-beauty render channels allow additional information (usually geometric information) to be generated, such as depth, normals, position, or material and object masks.

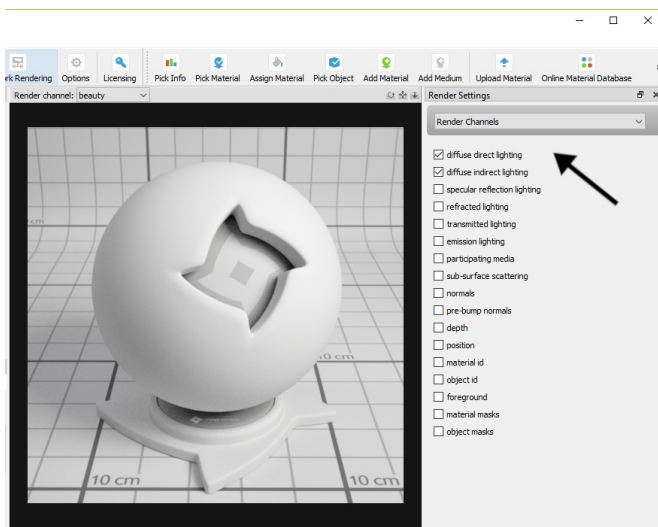
## Support for Render Channels

Render channel support was introduced in Indigo Renderer 4.2.

Render channel support is available in Indigo Renderer, but not in Indigo RT.

## Enabling Render Channels

Render channels can be enabled or disabled in the Render Channels page in the Render Settings widget:

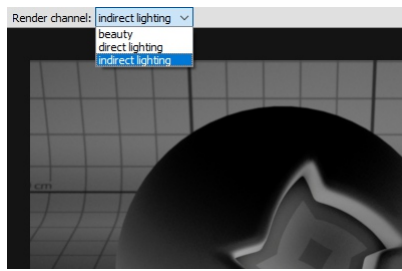


Render channel settings.

Note that enabling or disabling a channel will restart the render if it is rendering.

## Viewing Render Channels

To change which render channel is shown in the Indigo user interface, you can use the drop-down box above the render display:

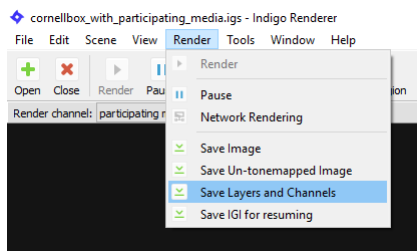


Select render channel to display.

Only enabled render channels will be available in the drop-down box.

## Saving Render Channels

You can save the render channels to disk with the *Save Layers and Channels* menu command in the *Render* menu.



Save Layers and Channels Menu command.

## Combine into single file option

This option can be found in the Indigo Options dialog (*Tools > Options*), in the *Image Saving* tab.

If enabled, all light layers and enabled render channels are combined into a single EXR file when saving with the *Save Layers and Channels* menu command.

If disabled, then one EXR file is saved for each light layer and for each enabled render channel. The filenames will have the channel name as a suffix, for example *render\_channel\_test\_depth.exr*, *render\_channel\_test\_normals.exr* etc..

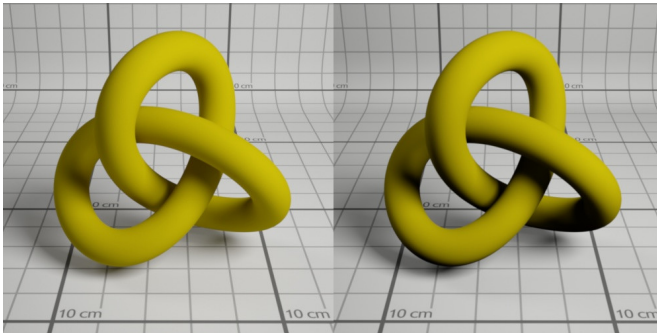
## Beauty Render Channels

Beauty render channels are components of the final 'beauty' render, where each render channel captures all the light from a particular class of light path from the emitter to the camera.

### Direct Lighting Channel

This render channel shows direct diffuse lighting.

In other words, it shows light that is emitted from a light source, and then bounces exactly once diffusely off an object and into the camera.

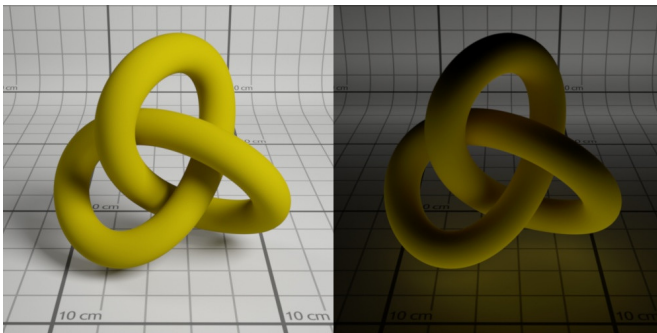


left image - beauty render. right image - direct lighting channel

### Indirect Lighting Channel

This render channel shows indirect diffuse lighting.

In other words, it shows light that bounces around a scene at least once, and then bounces diffusely off an object immediately before hitting the camera.

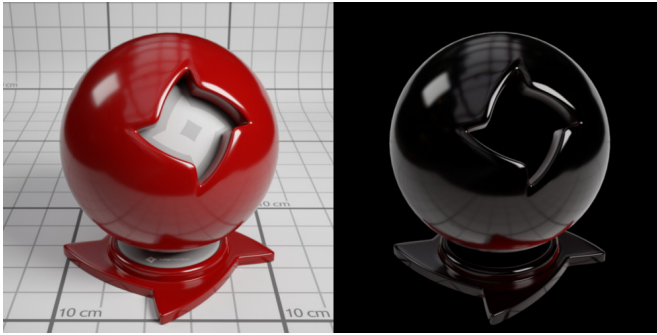


left image - beauty render. right image - indirect lighting channel

### Specular Reflection Lighting Channel

This render channel shows specularly reflected light.

In other words, it shows light that bounces around a scene, and then bounces specularly (e.g. bounces off a smooth surface) off an object immediately before hitting the camera.



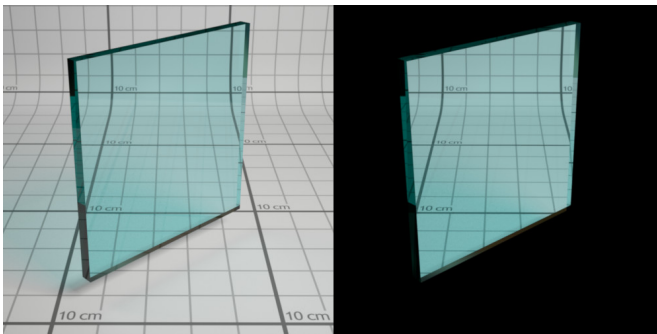
right image - Specular reflection channel

## Refracted Lighting Channel

This render channel shows refracted light.

In other words, it shows light that bounces around a scene, and then is refracted through a transparent object immediately before hitting the camera.

Glossy transparent and specular materials will contribute to this channel.



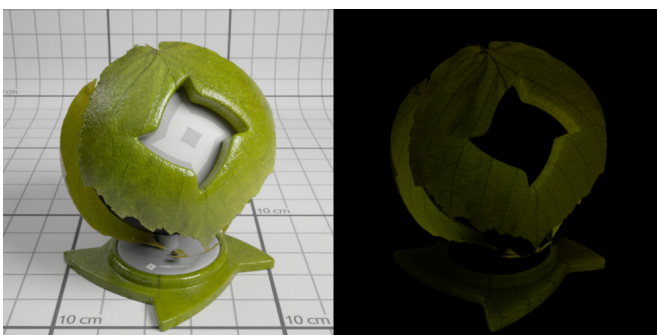
right image - refracted lighting channel

## Transmitted Lighting Channel

This render channel shows light diffusely transmitted through a surface.

In other words, it shows light that bounces around a scene, and then is diffusely transmitted through a partially or full transparent object immediately before hitting the camera.

Diffuse transmitter and double-sided thin materials will contribute to this channel.



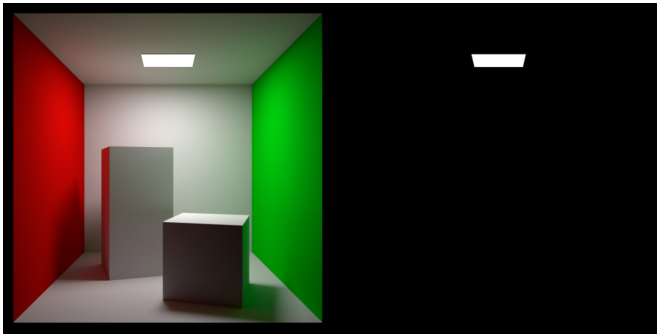
A double-sided thin material. Right image - transmitted lighting channel

## Emission Lighting Channel



This render channel shows directly emitted light.

In other words, it shows light that is emitted from a light source, and then directly strikes the camera.

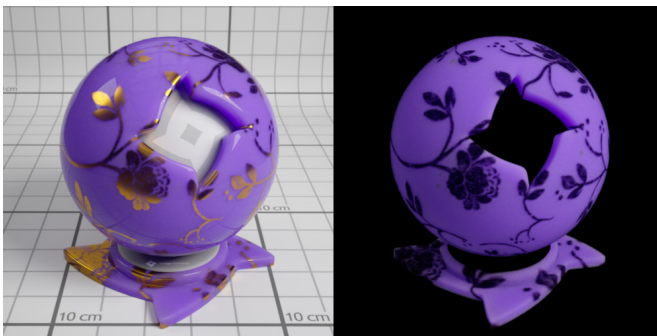


right image - emission lighting channel

### SSS Lighting Channel

This render channel shows light that has undergone sub-surface scattering.

In detail - it shows light that is emitted from a light source, bounces around the scene, undergoes SSS, exits a medium through a transparent material, and then directly strikes the camera.

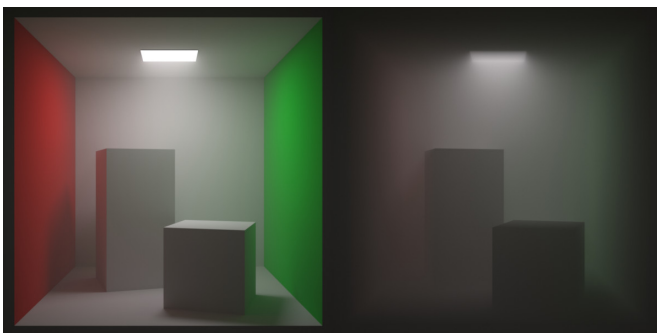


right image - SSS lighting channel

### Participating Media Lighting Channel

This render channel shows light that has undergone participating-media scattering.

In detail - it shows light that is emitted from a light source, bounces around the scene, undergoes participating media scattering, then directly strikes the camera.



right image - Participating Media lighting channel

## Non-Beauty Render Channels

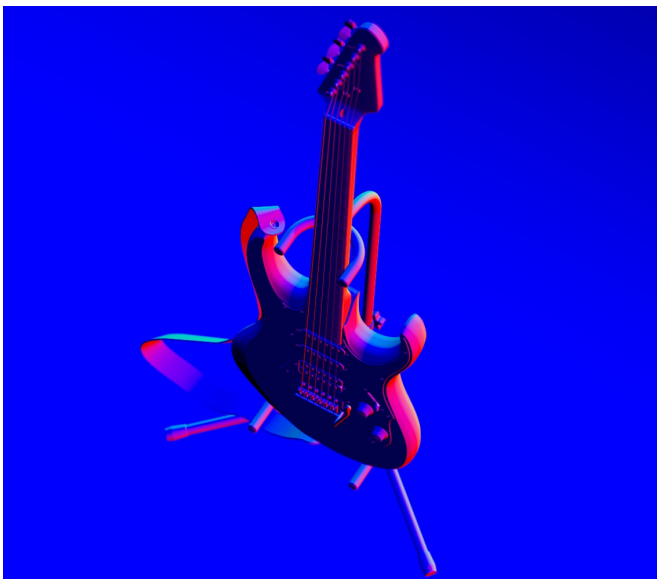
### Normals Channel

This render channel shows the normalised surface normal in world space. Channel values will range from -1 to 1.



### Normals pre-bump Channel

This render channel shows the normalised surface normal in world space, before bump mapping was applied. Channel values will range from -1 to 1.



## Depth Channel

This render channel shows the distance from the camera to the first hit point along a ray from the camera, divided by the diameter of the scene.

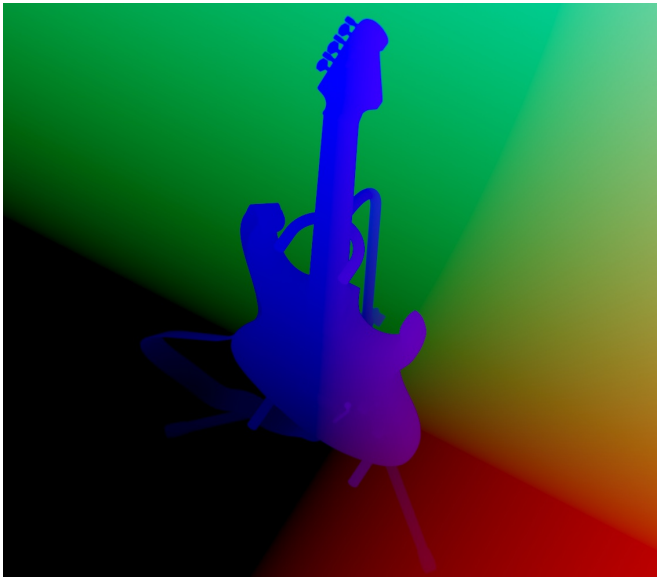
Channel values will range from 0 to 1.



## Position Channel

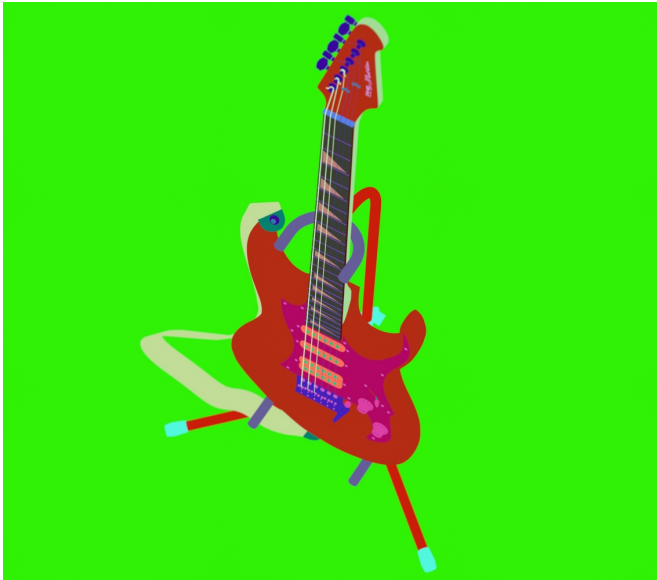
This render channel shows the world-space position of the surface.

Channel values have no fixed range and may be negative.



## Material ID Channel

This render channel has a pseudo-random colour assigned to each material.



## Object ID Channel

This render channel has a pseudo-random colour assigned to each object.



## Albedo Channel

This render channel shows the albedo (colour) of the hit material. For specular materials the path is continued and the reflected or refracted albedo is shown instead.

In Indigo 4.4.1 or newer.



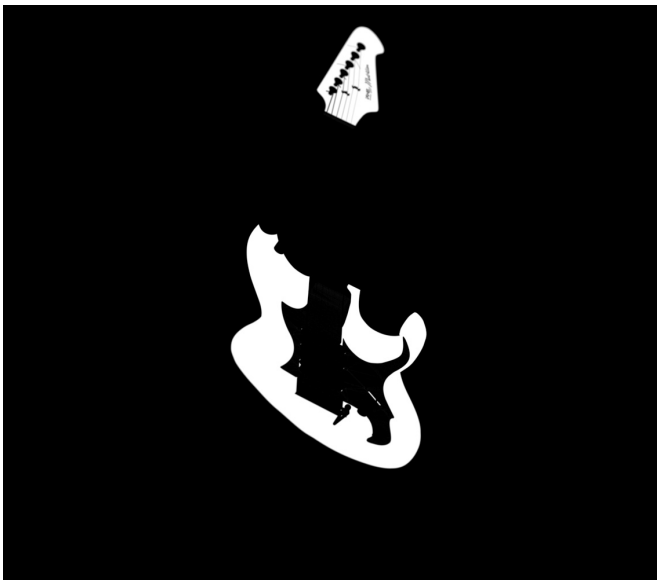
### Material Mask Channels

You can have multiple material mask channels.

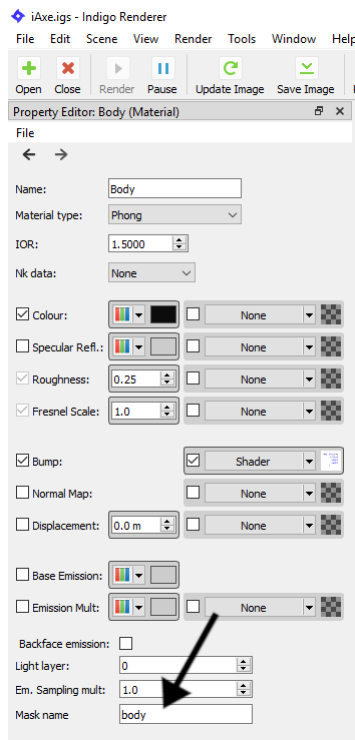
All materials that have a common mask name will be rendered white - everything else will be rendered black.

There will be a material mask channel automatically created for each unique material mask name set in the scene.

Don't forget to enable the *material masks* checkbox in the Render Channels tab to turn on the material mask channels!



You can set the material mask name in the Indigo UI in the property editor after selecting a material:



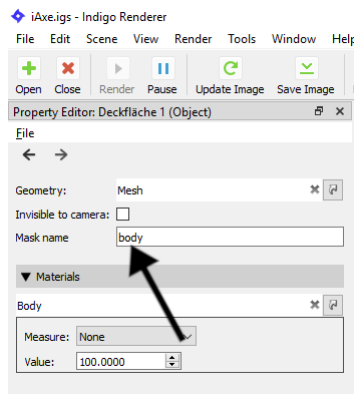
## Object Mask Channels

You can have multiple object mask channels.

All objects that have a common mask name will be rendered white - everything else will be rendered black.



You can set the object mask name in the Indigo UI in the property editor after selecting an object (for example with the *Pick Object* tool):



## Render Queue and Animation

Indigo has built-in animation and render queue support, thanks to the Indigo Queue (.igq) format. Render queues integrate smoothly with network rendering, which means you can use all the computers on your network to render an animation or set of images more quickly.

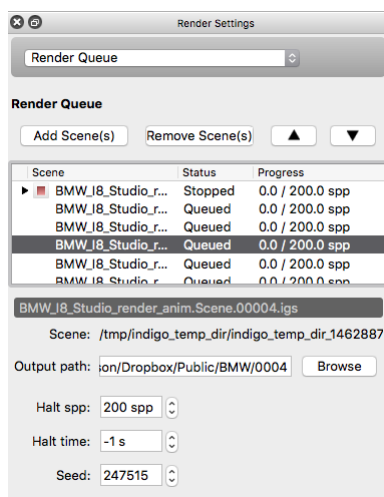
### Render Queue

A Render Queue enables rendering a sequence of frames, Indigo scene files, with full control over the rendering process and halt settings for the different frames.

### Creating a Render Queue

When exporting an animation from one of Indigo's 3D modelling package plugins, the Render Queue and an Indigo Queue file is created automatically.

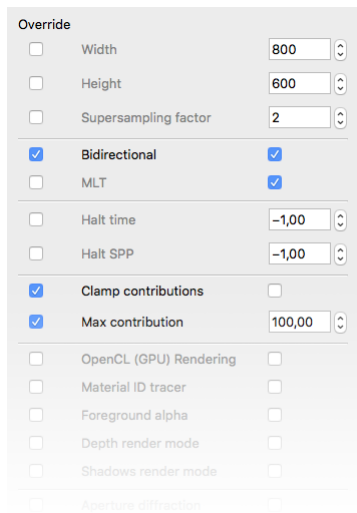
When loading a scene from inside the Indigo GUI, only a single item is added to the render queue, however multiple scenes can be added using the Add Scene(s) button, with a halting condition (on rendering time, samples per pixel or both) to specify how long they should render for. To change halt conditions for multiple frames, simply shift click to select the desired frames, or use ctrl - A (cmd-A on Mac OS X) to select all of them, then change the values to your liking. When changing output paths for the rendered frames, you can use '%frame', which will be replaced with the frame index (0001, 0002, etc.).



### Overrides

As of version 4, Indigo has override capabilities for all render settings. This enables the user to change parameters like resolution, render method and clamping for all frames in an animation or sequence. To change a setting, use the controls on the right of the setting. To use the new setting for the other frames as well - simply tick the "Override" tick-box on the left.



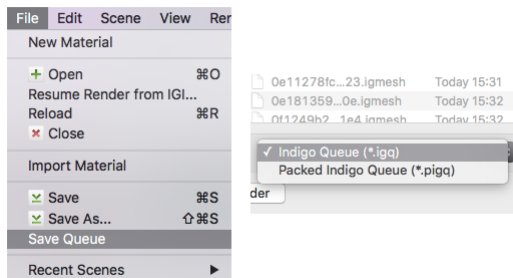


## Indigo Queue and Packed Indigo Queue formats

To save an Render Queue, go to File -> Save Queue. You can then choose between an Indigo Queue (.igq) format and a Packed Indigo Queue format.

- The Indigo Queue option saves the Render Queue data by itself and doesn't include any scene data.
- The Packed Indigo Queue is a self-contained archive with everything needed to render a sequence of scenes, including all referenced scenes, models and textures etc and with paths made relative. It can be unzipped with compression programs such as 7-Zip. This is the preferred format for distributing Indigo animations.

Opening either of these files in Indigo will add the referenced scene files to the Render Queue window.



## Render Settings

This section documents the render settings available in Indigo. Some of these settings may not be exposed directly, depending on which modelling package you're using.

### Render mode

Specifies the [render modes](#) Indigo should use to render the scene.

If you are unsure which to use, bi-directional path tracing is recommended. See also the [render mode guide](#).

### Foreground alpha

Renders and outputs PNG or EXR images with an alpha channel. The background are rendered completely transparent, whereas reflections and absorption in glass will show up semi-transparently in the render as expected.

### Clamp contributions

Contribution clamping basically works as a firefly filter.

If enabled, and the max contribution is low, it won't allow bright spots (fireflies) onto the render. As the max contribution gets higher and higher, brighter and brighter spots are allowed. Max contribution of infinity corresponds to contribution clamping being disabled.

Please note that, since it is clamping bright spots, it introduces bias into the render. Because of this it will be off by default. However it's a useful tool for artists, to remove fireflies from their images in a simple and efficient way.

Render settings: Pathtracing, Supersampling = 1



No clamping

Max contribution = 100

### Halt time

The number of seconds for which Indigo should render, after which the rendering is halted.

### Halt SPP

The number of samples per pixel (SPP) Indigo should render to, before rendering is halted.

### Network rendering

If network rendering is enabled, other computers on the network will assist in rendering the scene.

Normally the master computer also contributes in this process, however with the working master option disabled only the connected slave nodes will contribute to the rendering (leaving more resources available on the master).

### Save un-tone mapped EXR

An un-tone mapped EXR image is saved in the renders directory.

### Save tone mapped EXR

A tone mapped EXR image is saved in the renders directory.

### Save IGI

An un-tone mapped Indigo Image (.IGI) file is saved in the renders directory.

### Image save period

How often, in seconds, the rendered image(s) will be saved to the renders directory.

### Super sample factor

Super sampling helps to eliminate hard edges and fireflies in the render, at the cost of additional memory (RAM).

The amount of additional memory required to store the rendered image is proportional to the square of the super sample factor, i.e. for a factor of 2, 4x more memory is required, and for a factor of 3, 9x more memory is required. Note that this does not affect the size of the final image, and does not affect the rendering speed much (as long as the additional memory required is available).

### Watermark

If this is enabled, an Indigo logo is displayed on the bottom-right corner of the output render. This behaviour cannot be changed in the Free version of Indigo.

### Info overlay

If this is enabled, a line of text is drawn on the bottom of each render with various rendering statistics and the version of Indigo it was rendered with.

### Aperture diffraction

Selects whether aperture diffraction should be used. Please see the [aperture diffraction](#) documentation for more information.

### Render region

Specifies a subset of the image to be rendered; useful for quick previews in complex scenes.

### Render alpha

A render mode that sets the pixel alpha (opacity) based on if the pixel is considered to be in the scene foreground or background.

This allows you to composite your rendered image onto another image (such as a photographed background) in an image editing application. See [Foreground Alpha](#).

### Lens shift

Normally the lens is located in front of the middle of the sensor, however lens shifting allows you to move it. This is used to compensate for perspective effects when rendering with a relatively wide field of view.

### Advanced render settings

Typically users will not have to change any of these settings. Changing these from their defaults can lead to unexpected results and we recommended leaving them at their defaults.

### MNCR

MNCR stands for Max Number of Consecutive Rejections, a parameter used in the MLT rendering modes. A lower number can reduce "fireflies", but will introduce some bias into the render.

### Auto choose num threads

Automatically chooses all available CPU cores for rendering. Turn this off to manually specify number of threads to use for rendering, in conjunction with the "num threads" option below.

### Num threads

Define the number of threads for Indigo to render with.

### Logging

If true, a log from the console output is written to log.txt in the current working directory.

### Cache trees

If true, k-D trees and BVH data structures are saved to disk after construction, in the tree\_cache directory. If the

### **Splat filter**

Controls the filter used for splatting contributions to the image buffer. Either "fastbox" or "radial" are recommended; radial produces slightly higher quality images, but is blurry when used with a super-sampling factor of 1 (a factor of 2 or higher avoids this problem and delivers very high image quality).

### **Downsize filter**

Controls the filter used for downsizing super-sampled images. Only used when super sample factor is greater than 1. The same filters used for splatting can be used for the downsize filter, with a few extra options (please consult the technical reference for more information).

## Camera

Indigo implements a physically-based camera model which automatically simulates real-world phenomena such as depth of field (often abbreviated as DoF), vignetting and aperture diffraction.

This is a crucial component for Indigo's "virtual photography" paradigm, as it allows the user to use familiar settings from their camera in producing realistic renderings of their 3D scenes.

### Aperture radius

Defines the radius of the camera aperture.

A smaller aperture radius corresponds to a higher f-number. For more information on this relationship please see the excellent Wikipedia page on [f-numbers](#).

### Focus distance

The distance in front of the camera at which objects will be in focus.



### Aspect ratio

Should be set to the image width divided by the image height.

### Sensor width

Width of the sensor element of the camera. A reasonable default is 0.036 (36mm). Determines the field of view (often abbreviated as FoV), together with the lens sensor distance.

### Lens sensor distance

Distance from the camera sensor to the camera lens. A reasonable default is 0.02 (20mm).

### White balance

Sets the white balance of the camera. See White Balance.

### Exposure duration

Sets the duration for which the camera's aperture is open. The longer the exposure duration, the brighter the image registered by the sensor.

### Autofocus

When this option is enabled Indigo will perform an autofocus adjustment before rendering, automatically adjusting the focal distance based on the distance of objects in front of the camera.

### Obstacle map

An obstacle map texture is used when calculating the diffraction through the camera aperture, to change the way the aperture diffraction appears.

### Aperture shape

This allows a particular shape of camera aperture to be specified.

The allowable shapes are "circular", "generated" or "image". A preview of the final aperture shape will be saved in Indigo's working directory as "aperture\_preview.png". An image aperture shape must be in PNG format and square with power-of-two dimensions of at least 512 x 512. The image is interpreted as a grey-scale image with the white portions being transparent and black being solid.

For further information and example images please see the [aperture diffraction](#) sub-section.

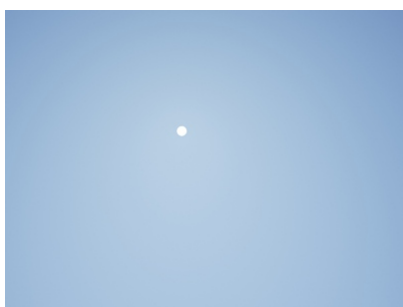
## Aperture diffraction

Aperture diffraction allows the simulation of light diffraction through the camera aperture. Such diffraction creates a distinct "bloom" or glare effect around bright light sources in the image. The shape of the glare effect is determined by the shape of the aperture.

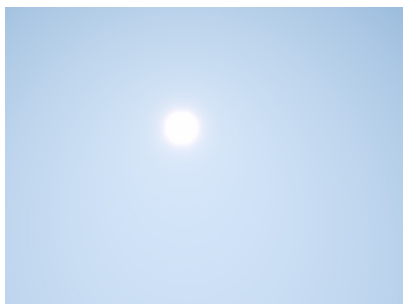
You can enable or disable aperture diffraction via the "Enable aperture diffraction" checkbox in the [imaging section](#) of the render settings view.

Post-process aperture diffraction dramatically increases the amount of memory (RAM) used, and the time taken to update the displayed image.

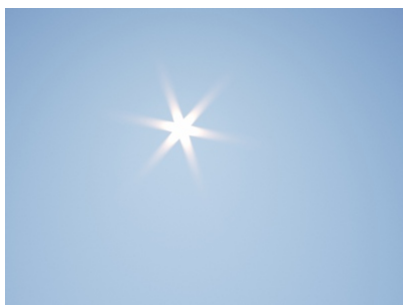
The following images illustrate the effect of aperture diffraction with various aperture shapes:



A simple render of the sun, without aperture diffraction enabled.

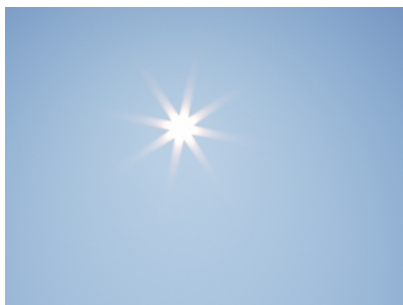


Aperture diffraction with a circular aperture.



Aperture diffraction with a 6-blade generated aperture.





Aperture diffraction with an 8-blade generated aperture.

## Camera types

In addition to the normal thin-lens camera model, Indigo supports two additional camera types: orthogonal/parallel, and spherical.

Please note that these additional camera types are professional visualisation features and are only available in Indigo Renderer, and not Indigo RT.

### Thin-lens perspective camera

This is the default camera mode, with many familiar settings from standard photography. For more information please see the parent [Camera page](#).



Concrete House scene by Axel Ritter (Impulse Arts)

### Orthogonal/parallel projection camera

In this mode, there is no foreshortening due to perspective, i.e. objects far away appear the same size as those near to the camera. This is commonly used in architectural visualisation for building plans.

Please note that since there is no perspective in this camera mode, the sensor width needs to be very large in order to be able to image large objects; this can lead to somewhat implausible (though not unphysical!) circumstances in which you have building-sized sensors.

For more information please see the SkIndigo [orthographic camera tutorial](#).



Garden Lobby scene by Tom Svilans (StompinTom)

### Spherical projection camera

The spherical projection camera renders a complete 360 degree view at once, which is useful for making environment map renders (for example to use as High Dynamic Range images for illuminating other scenes). It can also be used to render images for 360 degree panorama viewing applications.

Because spherical maps cover twice as many degrees in longitude as latitude, it is recommended to make your spherical renders with aspect ratio 2:1 in X and Y.



Weekend House scene by Axel Ritter (Impulse Arts)

## Tone mapping

Tone mapping changes the brightness and contrast of your image. It can be done at any stage during the render process. Changes to tone mapping will be applied immediately to the rendered image. Tone mapping is non-destructive, so you can play around with the different tone mapping settings without permanently effecting the rendered image. You may want to tone map your image using different settings, and press Save Image to save out several different images.

How it works: Indigo creates a high dynamic range (HDR) image as it renders, and during the tone mapping process this will be converted to a low dynamic range red, green and blue image that can be displayed on your computer monitor. When subsequently saving the image, the user can choose to either save the tone mapped image in JPG, PNG, TIFF or EXR format. Indigo also supports saving of the un-tone mapped HDR image in EXR format for external tone mapping.

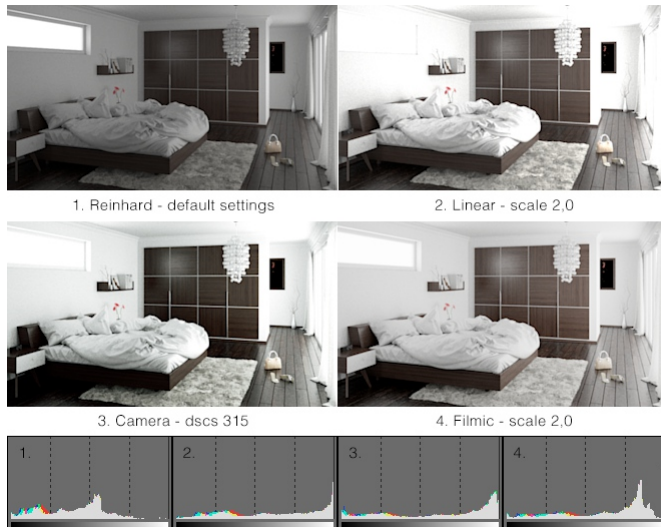
### Tone mapping method comparisons

Indigo has four different tone mapping techniques that you can choose from: Reinhard, Linear, Camera and Filmic. Here are some comparisons of the different methods, and their strengths and weaknesses, using example scenes from Zalevskiy and Zom-B respectively.



Note how the Camera tone mapping method gives the image a more photographic look with higher contrast and a slight colour tint.

Reinhard is the simplest to use, but once mastered, camera tone mapping can give a nice artistic feel to the renders.



The Filmic tone mapper does a good job with achieving a bright look without burning out the whites.

## Linear

The simplest tone mapping method, linear depends on just a single number. Every pixel in the HDR image will be multiplied by this number.

## Reinhard

Reinhard is a method based on a paper by Reinhard, Stark, Shirley and Ferwerda from the University of Utah. It is an easy to use tone mapping technique because it automatically adjusts to the amount of light in the scene. It can be tricky to get linear or camera tone mapping to look balanced in scenes where there is an extremely bright light source – the Reinhard method is a good choice for scenes like this.

The default Reinhard settings of prescale=6, postscale=1, burn=2 will give good results for most renders. If you want to adjust the Reinhard method, below is an approximate description of each parameter.

Prescale	Similar to a contrast control, works by increasing the amount of light in the HDR buffer.
Postscale	Works like a brightness control, increases the absolute brightness of the image after it has been tone mapped.
Burn	Specifies the brightness that will be mapped to full white in the final image. Can be thought of as gamma control.

## Camera

Camera tone mapping simulates the working of a photographer's camera. You adjust the exposure and ISO settings as you would in a real camera to modify the tone mapping.

The parameters you modify are:

The ISO number represents the speed of film that is used. The higher the ISO number, the more ISO light will be collected in the HDR Image. In low light situations, a fast film should be used, such as ISO 1600, and in bright lighting situations, a slow film can be used, such as ISO 100.

The exposure value can range from -20 to +20 and represents a correction factor that is applied to EV the collected light. The higher the EV, the brighter the final image will be. Increasing the EV by one will make the image twice as bright.

The final parameter is the response function. This specifies the type of film or digital camera to emulate. Different films and cameras emphasise different colours. The response functions are taken from real cameras – for example the images below use Ektachrome 100CD film which is famous for being used by National Geographic in their older photos.

A good default for sunny well-lit scenes is an ISO of 200 and an EV of -5.0.

Here are previews of each of the Camera Response Functions



Page 1   Page 2   Page 3

### Filmic

Like Linear, this tone mapper has only one control, scale, but it responds quite differently compared to Linear with significantly less contrast at high scale values. This makes it particularly suitable for creating bright images while (to an extent) preventing highlights to burn out.

## Colour correction

The Imaging window in Indigo features two controls for simple colour correction: White Point, and Colour Curves.

### White Point

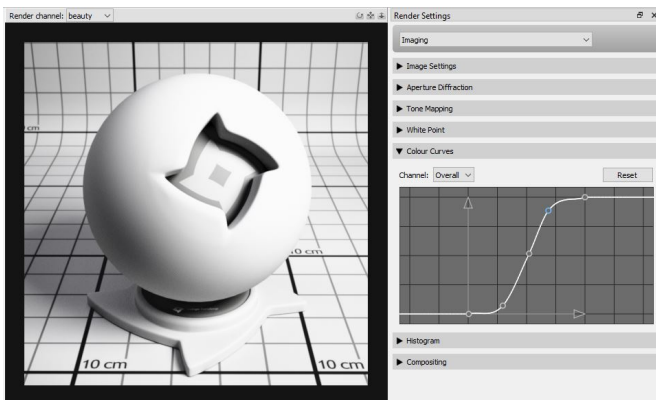
This control adjusts the white point of the image via chromaticity coordinates  $x$  and  $y$ . There are a number of presets to choose from, with the option to edit the white point completely manually.

### Colour Curves

Colour curves offer great control over tone and colour, right inside Indigo. There are curves for each RGB channel, as well as an Overall curve for control of overall brightness and contrast.

For each colour curve, the  $x$  coordinate (e.g. distance along the horizontal axis to the right) is the input value, e.g. how bright a pixel value is before it is transformed by the colour curve. The  $y$  coordinate (e.g. distance along the vertical axis upwards) is the output value, e.g. how bright a pixel value will be after it is transformed by the colour curve.

In this example image we are using the colour curve to generate more contrast in the image - in particular to make the blacks blacker and to saturate the brighter grey values towards white. This is achieved by using an S-shaped curve.



## Render Mode Guide

The different render modes in Indigo each have their own strengths and weaknesses. Different render modes perform differently on different scenes - some render modes will produce less noise, and some render modes will produce more. Since Indigo is an unbiased renderer, all render modes will eventually produce the same resulting render however.

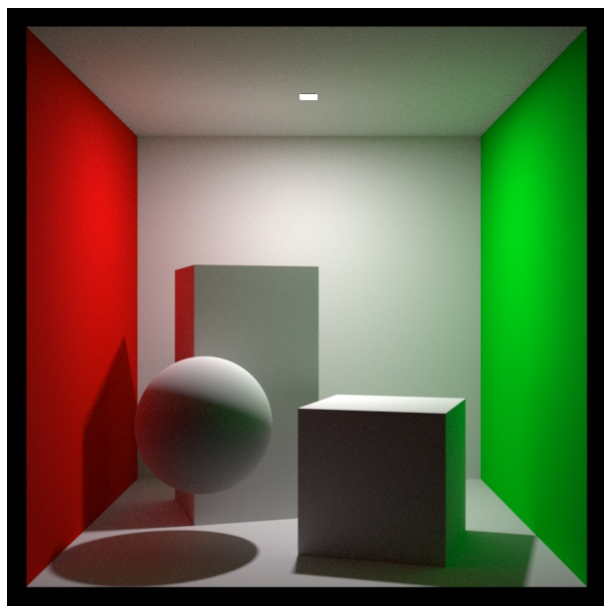
See the [Render mode description](#) page for more information on the various render modes.

In this section we will demonstrate the effect different render modes have on a couple of scenes. All images have been rendered for two minutes on a single computer.

The first scene shown here is an 'easy' scene: although there is a small light source, all materials are diffuse.

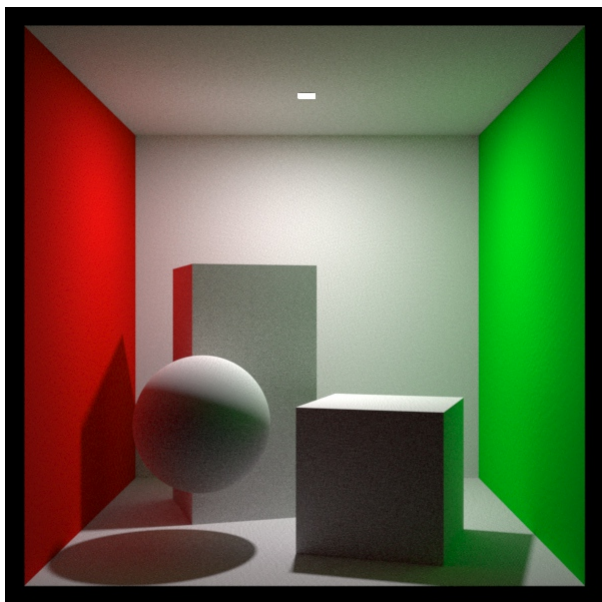
All render modes perform adequately on this scene, although the non-MLT modes (path tracing and bidirectional path tracing) perform better than the MLT modes.

This shows that MLT is not helpful on 'easy' scenes.

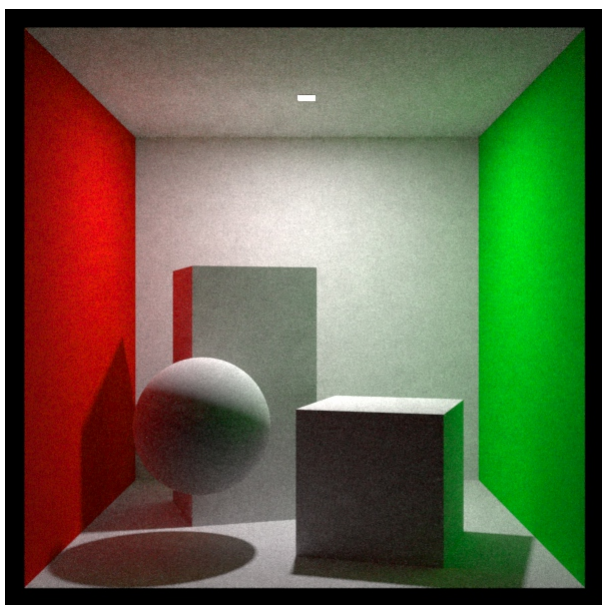


Path tracing render mode

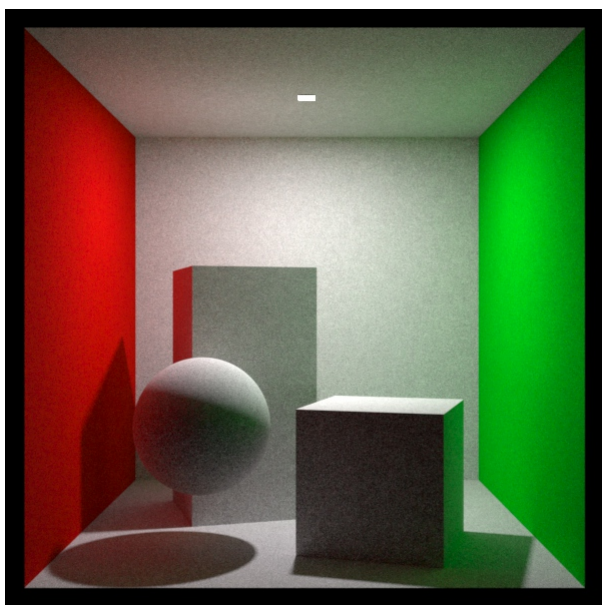




Bidirectional path tracing



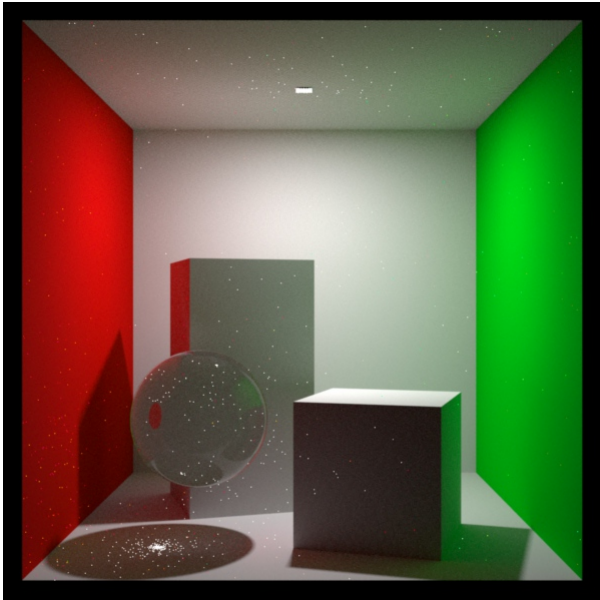
Path tracing with MLT



### Bidirectional path tracing with MLT

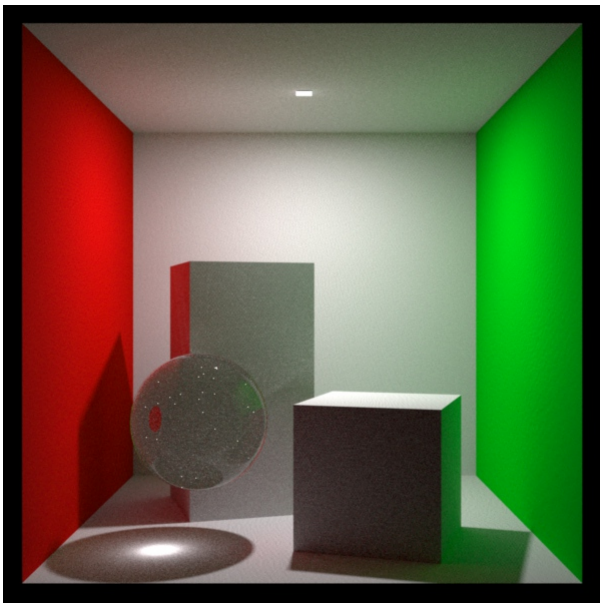
The second scene shown here is the same scene as before, except that the diffuse material on the sphere has been replaced with a specular material. The combination of a small, bright, light source and a specular material make this scene a 'hard' scene.

The path tracing render mode struggles, producing 'fireflies' (white dots) over the image. (These white dots are not errors, they are just 'noise' in the image, that will go away after a long enough render time)



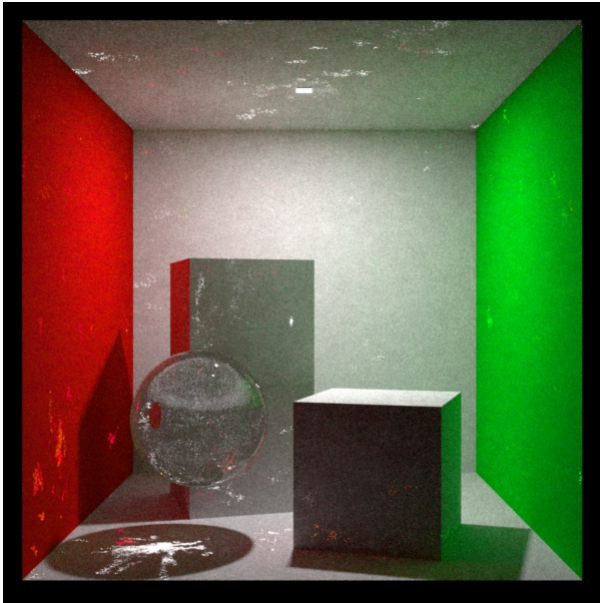
Path tracing render mode

The bidirectional path tracing mode does much better. There are still some fireflies on the sphere. These are from the notoriously difficult 'specular-diffuse-specular' paths that are a problem for unbiased renderers.



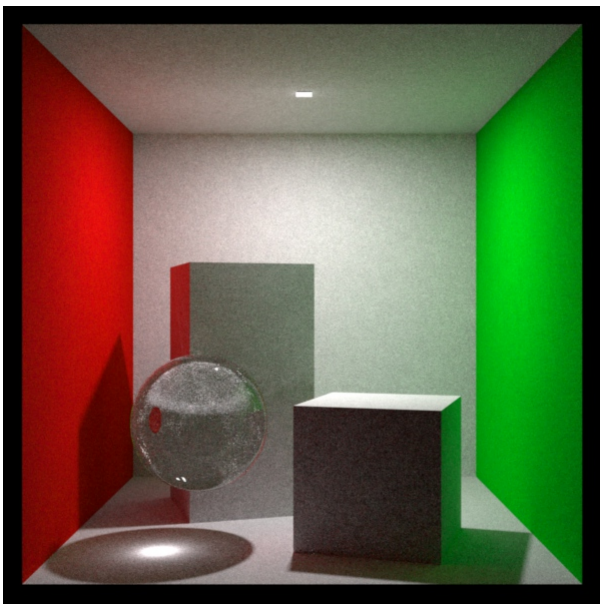
Bidirectional path tracing

Path tracing with MLT 'smears-out' the fireflies produced with the path tracing render mode. However the result is still not great.



Path tracing with MLT

Bidirectional path tracing with MLT is the most robust render mode in Indigo, as it performs adequately even in 'hard' scenes such as this one. There are still some smudges / splotches in this image, but they will smooth out after longer rendering.



Bidirectional path tracing with MLT

## Render mode description

Indigo is a ray tracer, which means that it renders scenes by firing out rays and letting them bounce around a scene to form ray paths. Different render modes control the way ray paths are constructed. Different render modes have different strengths and weaknesses for different kinds of scenes.

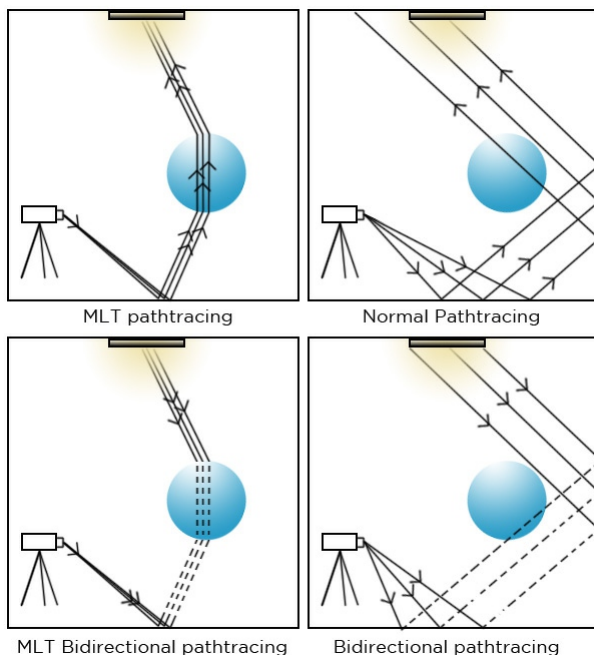
The main render modes are:

- Path tracing
- Bidirectional path tracing
- Path tracing with Metropolis Light Transport (MLT)
- Bidirectional path tracing with MLT

Generally, bidirectional path tracing should be used as it is the best all-round solution.

Metropolis light transport (MLT) can help rendering speeds in tricky scenes, such as scenes with small lights and glass panes, or sunlight reflected off glass or polished surfaces. See [this page](#) for more information about MLT.

Here is a diagram describing what they do:



Path tracing	Rays are fired from the camera, bouncing around the scene until they hit a light.
Bidirectional path tracing	Rays are fired from both the camera and light sources. They are then joined together to create many complete light paths.
Path tracing with MLT	Rays are fired from the camera, bouncing around the scene until they hit a light. When a successful light-carrying path is found, another is fired off on a similar direction. Gives good results for caustics.
Bidirectional path tracing with MLT	Rays are fired from both the camera and the lights, then connected to create many paths. When a successful light-carrying path is found, another is fired off on a similar direction. Gives good results for caustics and "difficult" scenes generally.

There are also other, more specialised, rendering modes available for use in post-production:



### Shadows

Only shadow catcher materials are rendered to an alpha layer, ready for compositing.

Shadows render mode is Indigo Renderer only and is not available in Indigo RT.

## Material database

The [online material database](#) is a shared repository where users can view, download and share Indigo materials. These are distributed as either .IGM (Indigo Material) or .PIGM (Packed Indigo Material) files, the latter being preferred since it automatically packages all required textures.

### Tutorials

- SketchUp: [Using the material database from SketchUp](#)

### Loading a material into the Indigo Material Editor

The first step is to download the material. To do this, click on the material thumbnail to go the material information page, for example <http://www.indigorenderer.com/materials/materials/11>.

Then click the download button above the material image. This will save an IGM or PIGM file to your computer.

Now click on the downloaded IGM or PIGM file. This should load the material into the Indigo Material editor and start rendering a preview of it.

### Uploading a material to the database

The Upload Material button in the toolbar will upload the material applied to the preview object together with the currently rendered preview image. The preview must be sufficiently clean (at least 200 samples per pixel) otherwise the upload will not succeed.

An alternative way to upload materials is through the Indigo website by hovering over Materials in the section bar, then clicking Upload a material.

# Materials

Accurately modelling the appearance of materials in a scene is crucial to obtaining a realistic rendered image. Indigo features a number of different [material types](#), each customisable with various [attributes](#), allowing great flexibility in material creation.

The material previews have been rendered in the main Indigo application (File -> New Material) using the default material ball scene.

# Material Attributes

Indigo materials can have a number of attributes or parameters to control their appearance. Some of these are relatively common and simple, such as the Albedo parameter, however others such as Absorption Layer Transmittance are particular to a material type and warrant a detailed explanation.

Many of the parameters can be given either as a constant, a [texture](#), or an ISL shader.

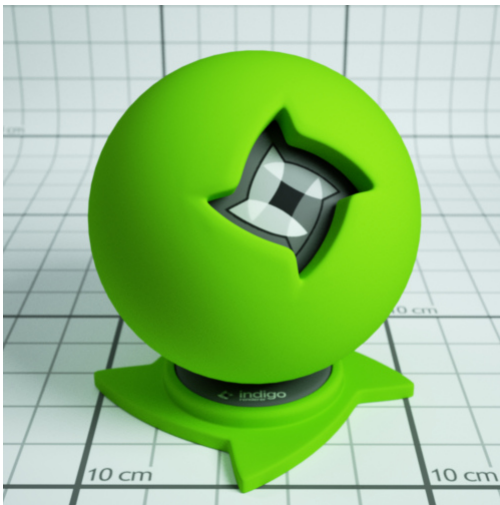


## Albedo

Albedo can be thought of as a basic "reflectivity colour" for a material.

For example, if a material has an RGB albedo with each component set to 0.0, it will be completely black and reflect no light; with each component set to 1.0, light would never lose energy reflecting off the material, which is physically unrealistic.

A comparison of various albedo values found in nature is [available](#) on Wikipedia.



An example diffuse material with a green albedo.

Materials which have an albedo attribute:

[Diffuse](#)

[Phong](#)

[Oren-Nayar](#)

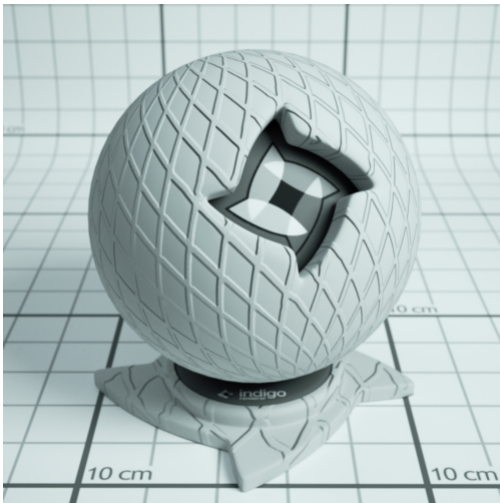
[Diffuse Transmitter](#)

## Bump

Bump mapping gives the illusion of highly detailed surface without actually creating more geometry; in other words, it's a shading effect which gives a good approximation to more a detailed surface.

When specifying a texture map, the texture scale (B value) tells Indigo how far the distance is from full black to full white in metres. Since bump mapping is only intended to simulate small surface features, this value will be quite small since it is specified in metres, usually on the order of about 0.002.

See the [Texture Maps](#) section for information on texture attributes.



An example material with a grating texture used as a bump map.

Materials which have a bump attribute:

[Diffuse](#)

[Phong](#)

[Specular](#)

[Oren-Nayar](#)

[Glossy](#) [Transparent](#)

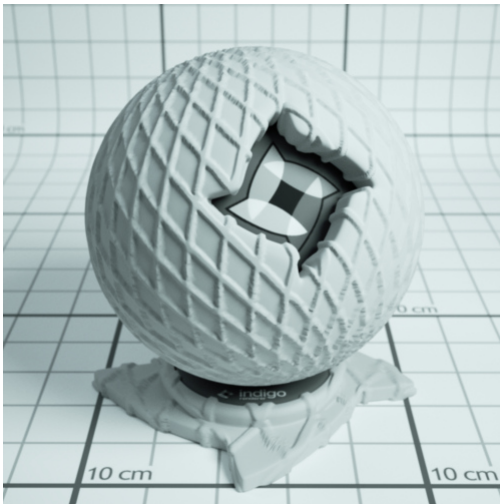
## Displacement

Unlike bump mapping, which is a shading effect and does not create actual geometry, displacement mapping correctly generates new geometry from a base mesh and specified displacement map, by displacing the mesh vertices along their normals according to the displacement map.

This ensures that object silhouettes are correctly rendered, and is recommended for large displacements where bump mapping would look unrealistic; even mountain ranges can be efficiently created with this technique.

A constant setting displaces the entire mesh evenly by the defined amount. A texture map displaces the vertices based on values in a grey-scale image.

See the [Texture Maps](#) section for information on texture attributes.



An example material a grating texture displacement map.

Materials which have a displacement attribute:

[Diffuse](#)

[Phong](#)

[Specular](#)

[Oren-Nayar](#)

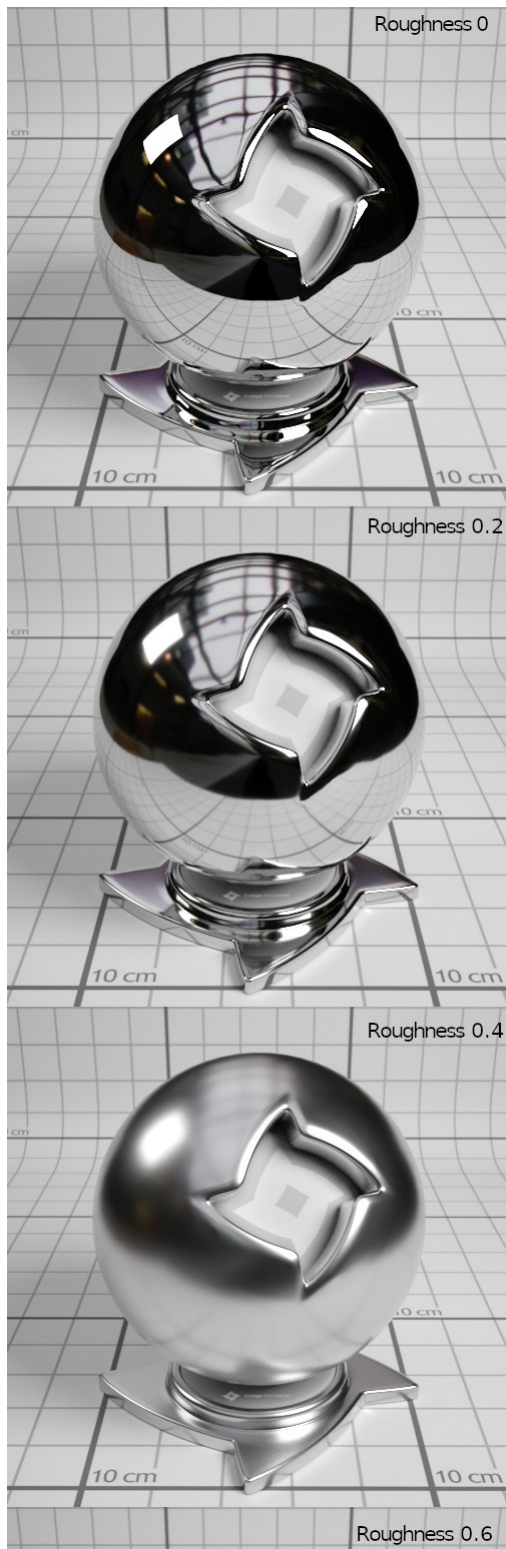
[Glossy Transparent](#)

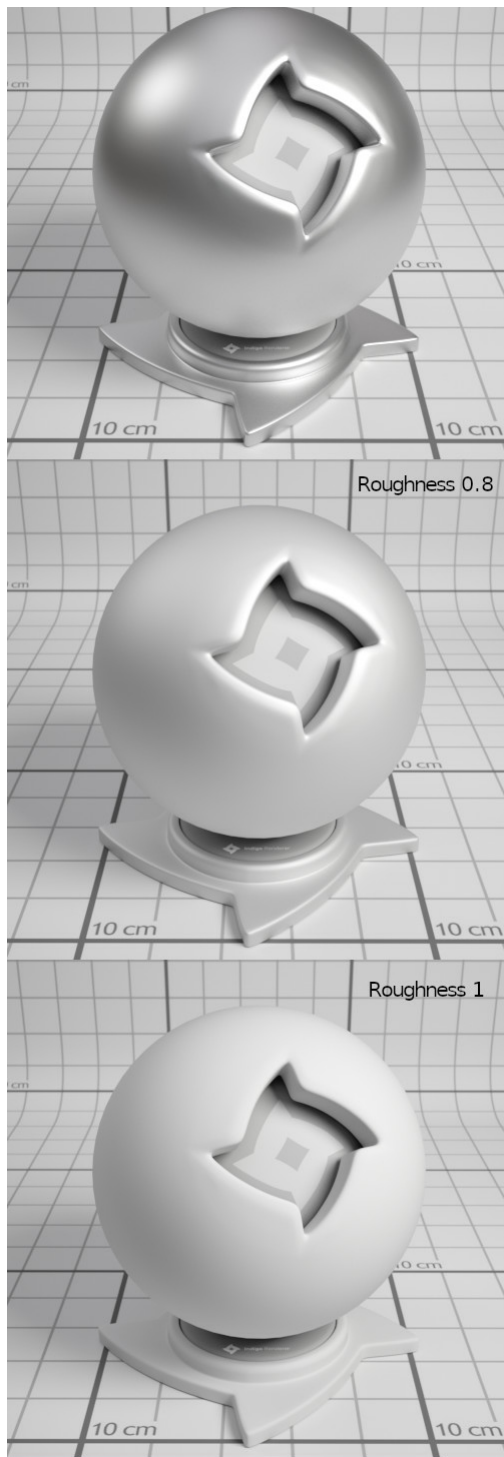
[Diffuse Transmitter](#)

## Roughness

The roughness parameter controls the roughness of the surface, with lower roughnesses corresponding to a smoother, more polished surface with mirror-like reflections.

The roughness varies from zero to one, and it can be set using a texture or a shader.





Example of Phong materials with different roughnesses.

Materials which have a roughness attribute:

Phong

Glossy Transparent

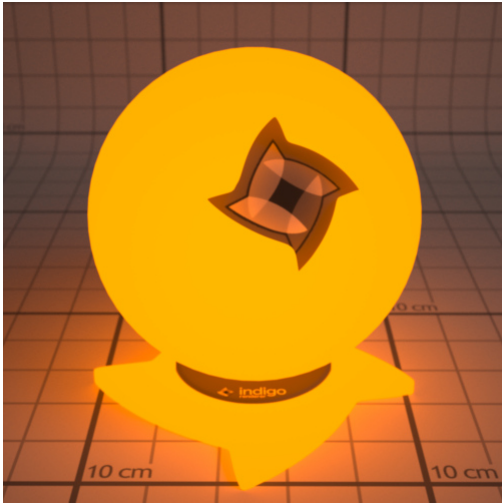
Coating

Double-sided Thin

## Base Emission

This parameter specifies the base amount of light a material emits (in units of Lumens), which can be modulated using the [Emission](#) parameter.

This is used to create light sources.



An example of a 1500 Kelvin blackbody emitter.

RGB: Light based on colour and brightness.

Uniform: A white light with intensity based on value given.

Blackbody: Light is based on the temperature. Measured in Kelvin.

Peak: Defines a band of wavelengths in which the material emits light.

Tabulated: The emission spectrum is specified at regular wavelength intervals, which is useful for entering lab-measured data when a very controlled simulation is required.

Materials which have a base emission attribute:

[Diffuse](#)

[Phong](#)

[Specular](#)

[Oren-Nayar](#)

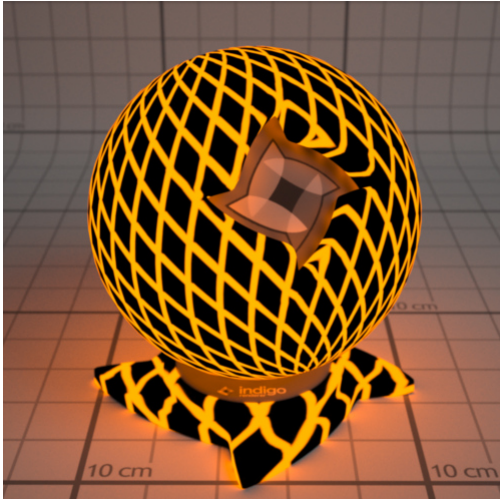
[Glossy Transparent](#)

[Diffuse Transmitter](#)

## Emission

The emission parameter multiplies the [Base Emission](#) to produce effects such as TV screens, where the brightness varies over the surface of the screen.

An emission scale parameter is available to scale the emission of the material by a given amount. Various photometric units are available.



An example material with a 1500 Kelvin blackbody emitter, modulated by a grating texture.

Materials which have an emission attribute:

[Diffuse](#)

[Phong](#)

[Specular](#)

[Oren-Nayar](#)

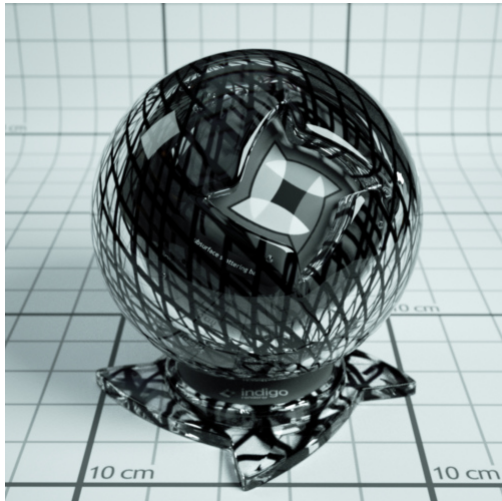
[Glossy Transparent](#)

[Diffuse Transmitter](#)

## Absorption Layer Transmittance

Absorption Layer Transmittance refers to the absorption of light at the transmitting surface by a given "layer", which allows further control over how specular materials appear without changing the medium's absorption properties (which is what usually creates the perceived colour).

This can be useful for producing a stained glass window effect for example.



An example material with a grating texture used as an absorption layer.

Materials which have an absorption layer attribute:

Specular

Glossy Transparent



## Layer

Light layers enable a rendered image to be split into additive "layers", in which each layer holds some contribution to the final rendered image.

How much a layer contributes to the final image can be adjusted interactively while rendering without restarting the process, and even after the render is completed, allowing for great flexibility in adjusting the lighting balance without having to do a lot of extra rendering. See [Light Layers](#) for more information on this subject.

The material's layer parameter specifies which light layer the (presumably light-emitting) material's contribution is added to.

Materials with attribute:

[Diffuse](#)

[Phong](#)

[Specular](#)

[Oren-Nayar](#)

[Glossy Transparent](#)

[Diffuse Transmitter](#)

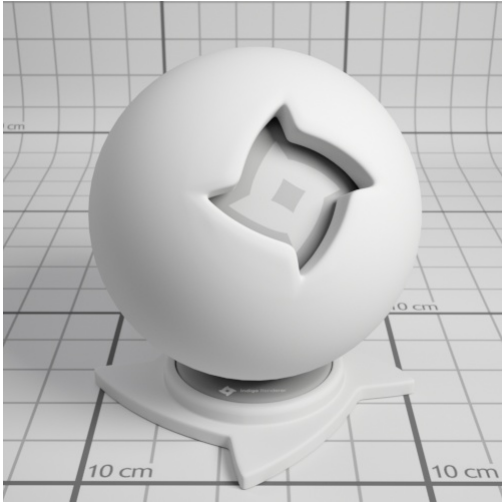
# Material Types

This section covers the various material types available in Indigo.

There are also video tutorials available for editing materials within Indigo, which include explanations of the material types, available [here](#).

## Diffuse

Diffuse materials are used for rough or "matte" surfaces which don't have a shiny appearance, such as paper or matte wall paint.



Attributes:

Albedo

Bump

Displacement

Base Emission

Emission

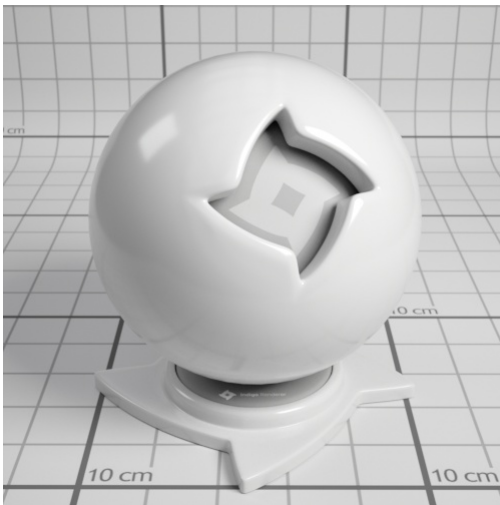
Layer

## Phong

The Phong material is a generalisation of the Diffuse material type, adding a glossy coating on top of the diffuse base (or "substrate"). The influence of this coating is controlled by its index of refraction (IOR), with higher values corresponding to a stronger specular reflection. (the default IOR is 1.5)

It is commonly used for materials such as polished wooden floors, car paints (when multiple Phong materials are blended together) and metals.

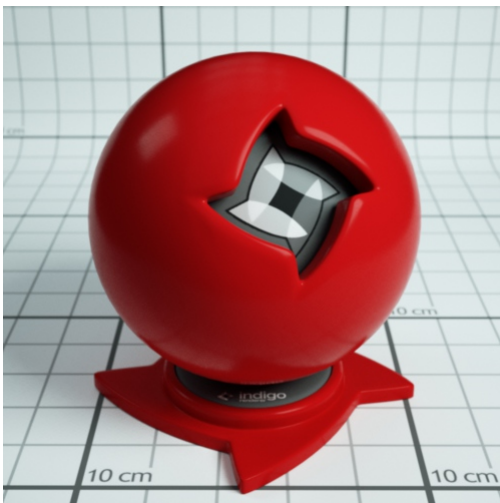
Metals in particular can be represented using either the "specular reflectivity" option (which allows a specular colour to be defined for the material), or via measured material data (referred to as NK data).



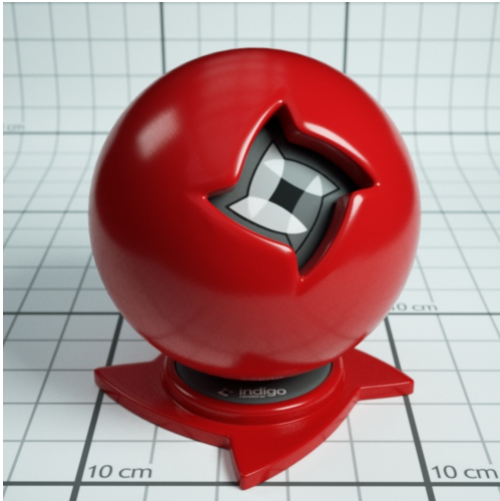
IOR (Index Of Refraction): Controls the influence of the glossy coating; higher values produce a stronger specular reflection. The IOR should be set to the real-world value for the material, if available.

For example, the [IOR of plastics](#) is around 1.5 to 1.6.

Oil paint binder has an [IOR of around 1.5](#).



Phong material with IOR 1.2



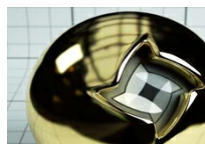
Phong material with IOR 1.5



Phong material with IOR 2.5

**NK data:** The Indigo distribution comes with a set of lab-measured data for various metals. If one of these data sets is selected, the diffuse colour and specular reflectivity colour attributes are ignored.

**Specular reflectivity colour:** When not using NK data, this allows you to set a basic colour for the metal; this is useful for uncommonly coloured metals such as Christmas decorations.



Green specular colour Au (gold) NK dataset Al (aluminium) NK dataset

**Attributes:**

Albedo

Bump

Displacement

Exponent

Base Emission

Emission

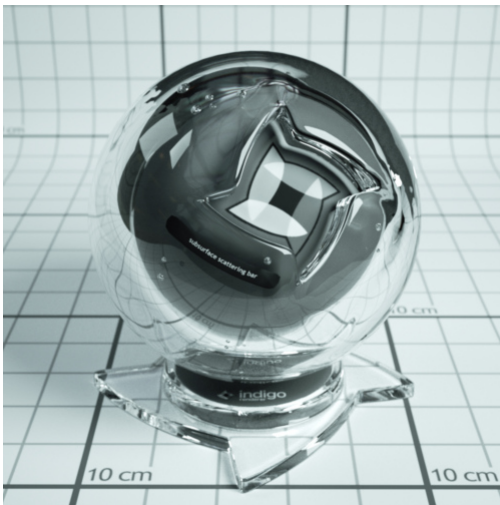
Layer

## Specular

Specular materials are idealised materials which refract and/or reflect as in classical optics for perfectly smooth or flat surfaces (e.g. mirror-like reflection).

A specular material can either transmit light, as is the case with glass and water for example, or not as is the case with metals. This behaviour is controlled by the material's "Transparent" attribute.

If a specular material transmits light, it will enter an internal medium whose properties define the appearance of the material (such as green glass, which has absorption mainly in the red and blue parts of the spectrum). For more information on this please see the [correct glass modelling](#) tutorial.



**Transparent:** If enabled, it allows light to pass through the material. Otherwise only reflected light is simulated.

**Attributes:**

[Albedo](#)

[Bump](#)

[Displacement](#)

[Base Emission](#)

[Emission](#)

[Absorption Layer Transmittance](#)

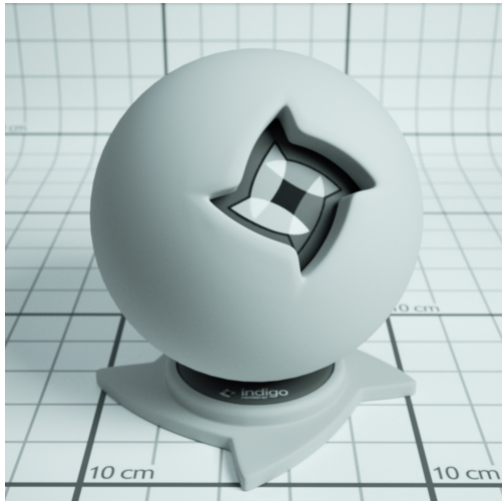
[Layer](#)

[Internal Medium](#)

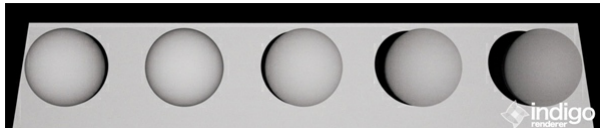
## Oren-Nayar

Oren-Nayar materials are another generalisation of the basic diffuse material type, except unlike Phong which generalises to shiny surfaces, Oren-Nayar generalises to rougher materials.

The appearance is quite similar to diffuse materials, but with less darkening at grazing angles; this makes it suitable for modelling very rough or porous surfaces such as clay or the moon's surface.



Sigma: Controls the roughness of the material. A higher sigma gives a rougher material with more back-scattering. The default sigma value is 0.3, and values higher than 0.5 primarily cause energy loss / darkening due to strong inter-reflection. A value of 0.0 corresponds exactly to diffuse reflection.



From left to right: Diffuse material, then Oren-Nayar with sigma values 0.0, 0.2, 0.5, 1.0.

Attributes:

Albedo

Bump

Displacement

Base Emission

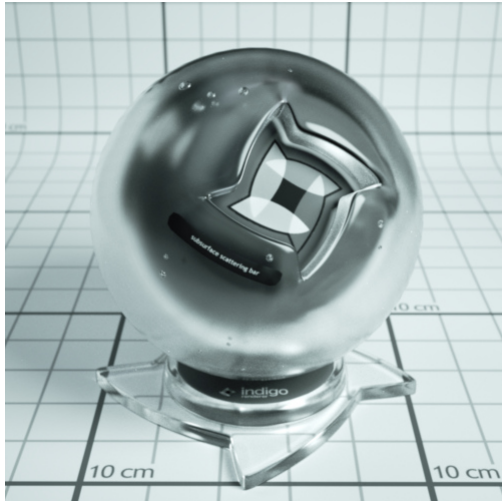
Emission

Layer

## Glossy Transparent

The glossy transparent material is a generalisation of the specular material, to allow non-perfect (i.e. rough) reflection and refraction, via an exponent parameter as with the Phong material.

It is commonly used for materials such as frosted glass or even human skin (with a low exponent value).



Attributes:

Exponent

Internal Medium

Absorption Layer Transmittance

Bump

Displacement

Base Emission

Emission

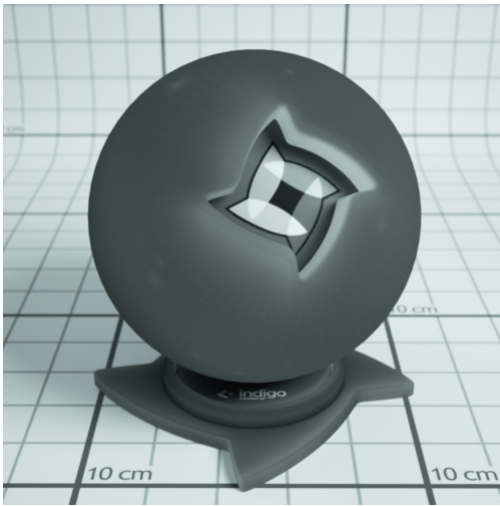
Layer



## Diffuse Transmitter

The diffuse transmitter material simulates a very rough transmitting material, which reflects no light back outwards. For this reason it is normally blended with a normal diffuse material to model surfaces such as curtains or lampshades.

It is meant to be used on single-layer geometry, and although it does not have an associated internal medium by default, it is possible to use one.



An example of the diffuse transmitter material on its own.



A blend between diffuse transmitter and normal diffuse materials to simulate the appearance of a curtain.

Attributes:

Albedo

Displacement

Base Emission

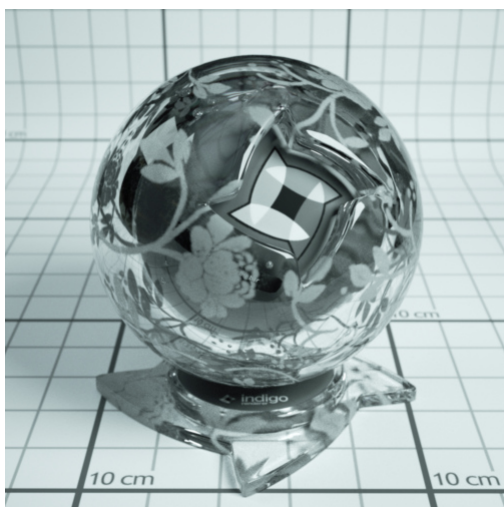
Emission  
Layer

## Blend

The blend material type isn't a material per se, rather it combines two sub-materials using blending factor.

This blending factor is a normal channel, and so can be constant, a texture or an ISL shader, allowing for great flexibility in material creation; a blend material can also use a blend as input, enabling so-called "shading trees" of arbitrary complexity.

Note that it's not possible to blend any combination of null and specular materials, except for the case where two specular materials with the same medium are being combined.



An [example blend material](#) from the material database, showing a blend between specular and diffuse materials.

Blend: Controls the amount of each material used. A value of 0 means only Material A is used, a value of 1 means only Material B is used, 0.5 implies a 50/50 blend, etc.

Step Blend: Instead of allowing partial blends, only one of Material A or Material B are selected, depending on whether the blending factor is below or above 0.5, respectively. This is recommended for "cut-out" clipping maps (such as for tree leaves), which produce less noise using this technique.

## Exit Portal

When rendering interior scenes, one frequently encounters an efficiency problem where the "portals" through which light can enter the scene from outside (e.g. an open window) are relatively small, making it quite difficult to sample.

Exit portals can greatly improve rendering efficiency in these situations by marking important areas through which light can enter the scene, which Indigo will directly sample to produce valid light carrying paths.

Although it is a material type, exit portals don't have any particular appearance of their own, they simply provide an "open window" through which the background material is seen, and through which it can illuminate the scene.

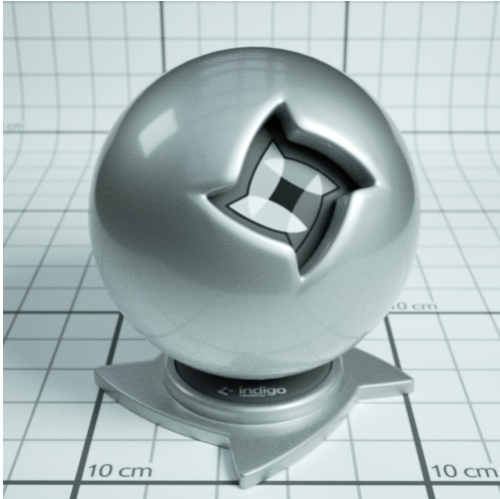
For more information and example images, please see the [SketchUp exit portal tutorial](#).

Requirements for exit portal usage:

- If any exit portals are present in the scene, then all openings must be covered by exit portals.
- The face normals must face into the interior of the scene.
- The exit portal material should only be applied to one side of a mesh (e.g. a cube), otherwise it will lose its effectiveness.

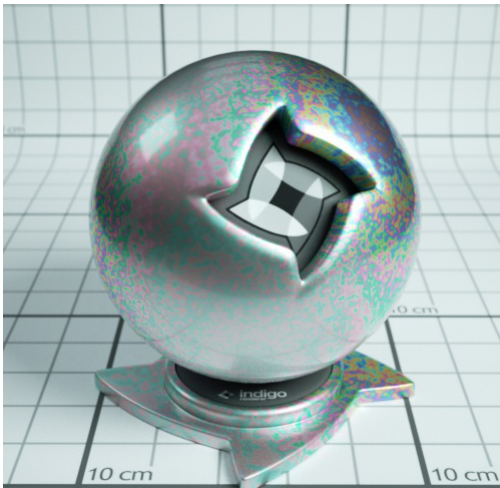
## Coating

The coating material simulates a thin coating of some kind of material over another material. For example, you can create a thin coating of varnish over a metal material.



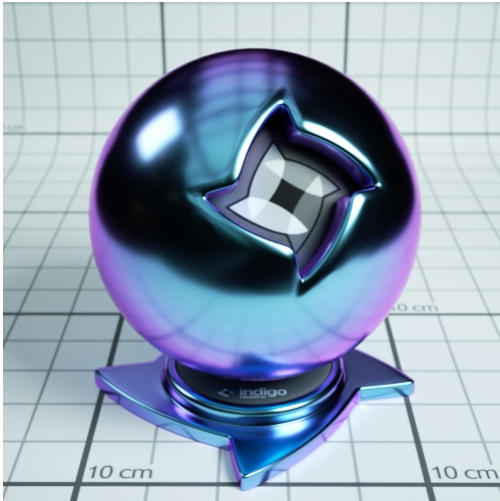
A coating material over a metal metal

The coating material can also simulate interference, which can give interesting effects when the thickness of the coating varies with position:



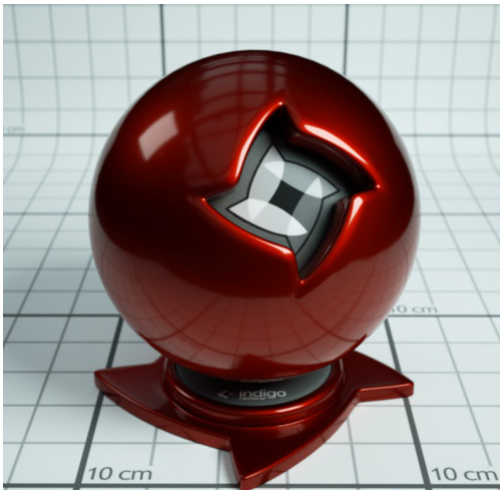
A coating material over a metal metal with interference enabled. Coating thickness is controlled with a shader.

Depending on the thickness of the coating, and how the thickness varies over the object (which can be controlled by a shader) you may get a kind of rainbow effect, or the material may just take on just a few colours:



400 nm thick coating showing colours from interference.

A coating can also absorb light, which will result in a tinted colour:

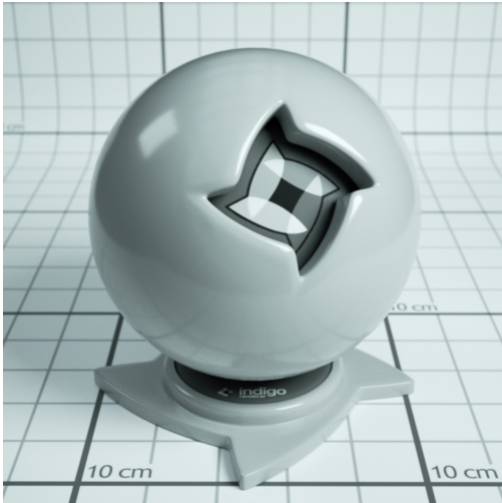


Coating with absorption over a metal material. The red colour comes from the absorption of non-red light as light passes through the coating layer.

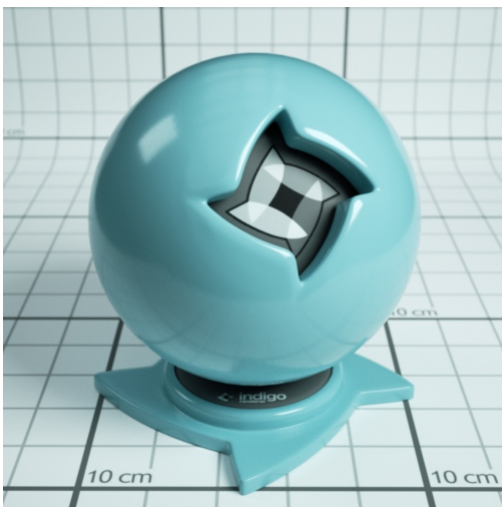
## Coating material attributes

### Absorption

Controls the absorption of light as it travels through the coating layer.



Coating without absorption

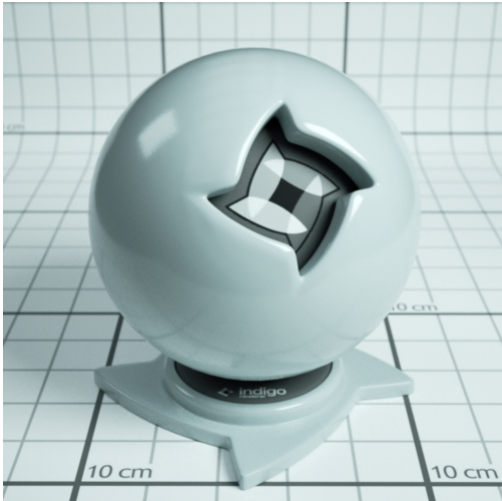


Coating with absorption

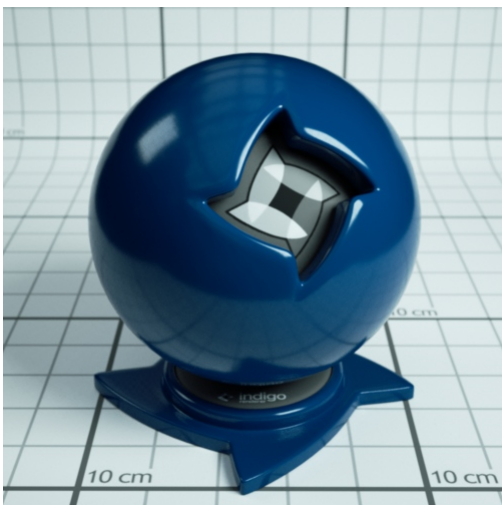
### Thickness

Controls the thickness of the coating layer. In the Indigo graphical user interface, the thickness is given in units of micrometres ( $\mu\text{m}$ ), or millionths of a metre.

A thicker layer will have stronger absorption.



Coating with absorption, thickness of 100  $\mu\text{m}$ .



Coating with absorption, thickness of 10000  $\mu\text{m}$ .

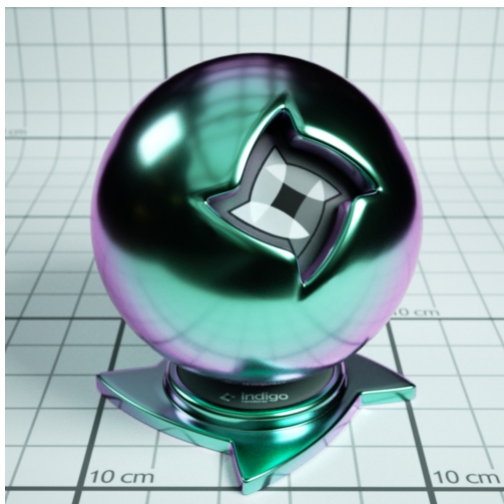
## Interference

If enabled, thin film interference is computed for the coating layer.  
The result will be most noticable when the coating layer is on the order of 1  $\mu\text{m}$  thick.



Coating without interference, thickness 0.6  $\mu\text{m}$ .

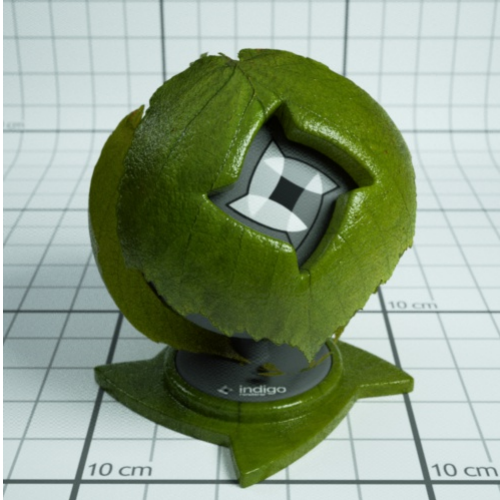




Coating with interference, thickness  $0.6\text{ }\mu\text{m}$ .

## Double-Sided Thin

The double-sided thin material is useful for modelling thin objects, such as paper or plant leaves.



The double-sided thin material uses two 'child' materials: the front material and the back material, which are used for the front and back of the surface respectively. This means that you can make, for example, a leaf material where the front uses a different texture from the back.

It is also possible to specify the transmittance colour, which controls how light is absorbed and coloured as it passes through the object.



Render using double-sided thin material by Enslaver

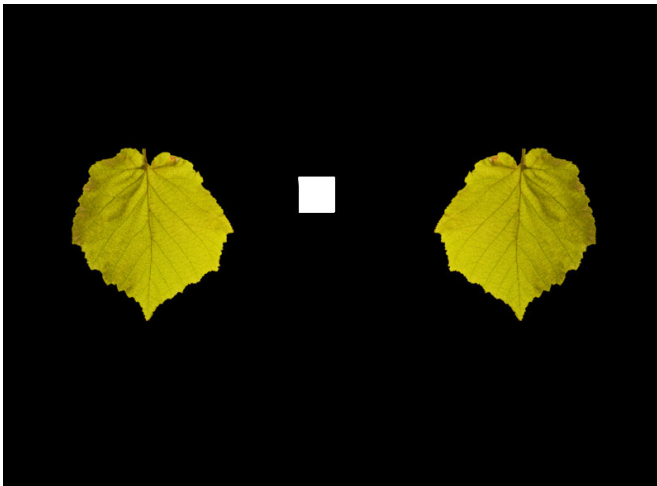
### Leaf example

When the leaf is lit from the front, you can see that the leaf colour differs based on if the leaf is front or back side towards the camera: (The leaf on the right has the front/top side towards the camera) The light is behind the camera.



Leaf lit from front

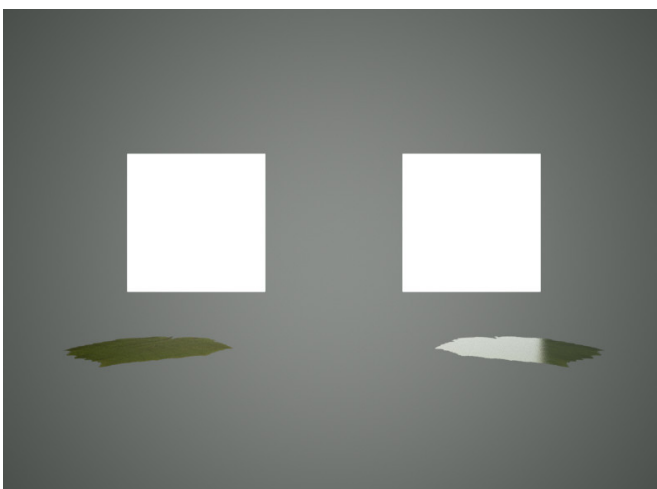
When lit from the back, the leaf looks the same from both the front and the back. This is also what happens with a real leaf (try it!)



Leaf lit from back

The roughness (exponent) of the top can be varied independently of the roughness of the bottom of the leaf.

In this render the leaf on the right is frontside-up, on the left backside-up:



Leaf lit from top

## Double-sided Thin Attributes

IOR: This is the index of refraction of the thin slab.

Front material: For light that is reflected diffusely back from the front surface, the front material controls the distribution of such light. Should be diffuse or Oren-Nayar. This material would be, for a leaf example, a diffuse material using the front-side albedo texture of your leaf.

Back material. For light that is reflected diffusely back from the back surface, the back material controls the distribution of such light. Should be diffuse or Oren-Nayar. This material would be, for a leaf example, a diffuse material using the back-side albedo texture of your leaf.

$r_f$ : this is the fraction of light that enters the slab that is reflected back to the side it came from. A good default value is 0.5

Transmittance: Some light scatters right through the slab and out the other side. The transmittance controls the absorption of such light. This is where you would use, for example, your transmittance map of a leaf.

Front Fresnel scale: A factor that controls the intensity of the specular scattering off the front interface of the slab. Default value is 1.

Back Fresnel scale: A factor that controls the intensity of the specular scattering off the back interface of the slab. Default value is 1.

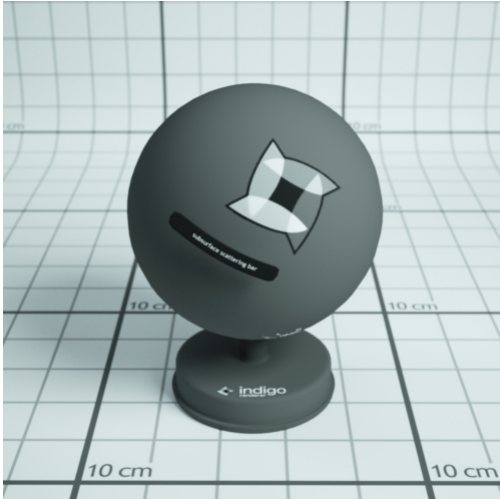
Front Roughness: Controls the roughness of the front interface of the slab.

Back Roughness: Controls the roughness of the back interface of the slab.

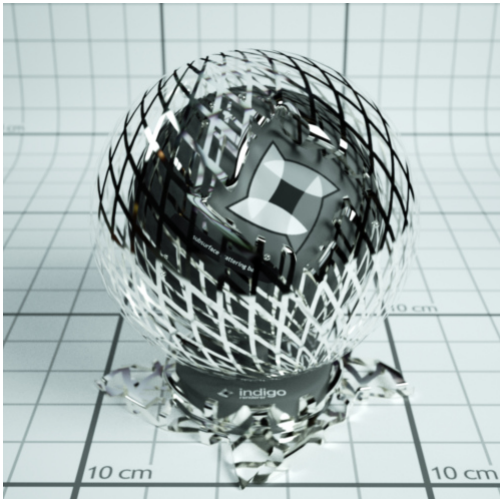
## Null

The null material is not a normal material type (like the exit portal material), but rather indicates that light should pass straight through it, unaffected in brightness and direction.

This on its own is not very helpful, however when combined with the [blend material](#) type it becomes very useful for making "cut-outs" such as leaf edges, which would otherwise highly complex geometry.



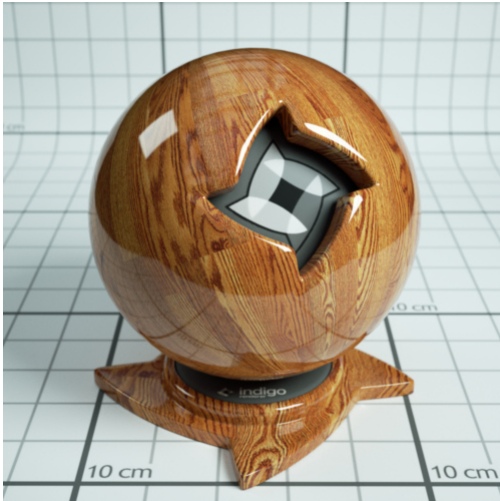
An example of the null material on its own.



An example of the null material blended with a Phong metal, using a grating texture as blend map.

## Texture Maps

Texture maps are a standard way of adding fine surface detail without adding more geometry. Indigo supports texture maps in many file formats, which are listed here.



An example Phong material with a wood texture applied.

### Supported texture formats:

JPEG	.JPG .JPEG	Greyscale, RGB and CMYK are supported.
PNG	.PNG	Greyscale, greyscale with alpha, RGB and RGBA are supported. 8 bits per channel and 16 bits per channel supported.
Truevision Targa	.TGA	Greyscale (8 bits per pixel) and RGB (24 bits per pixel) are supported.
Windows Bitmap	.BMP	Greyscale (8 bits per pixel) and RGB (24 bits per pixel) are supported. Compressed BMP files are not supported.
OpenEXR	.EXR	16 bits per channel and 32 bits per channel are supported.
TIFF	.TIF .TIFF	Greyscale, RGB, RGBA are supported. 8 bits per channel and 16 bits per channel supported.
GIF	.GIF	Supported. Gif animation is not supported.
RGBE	.HDR	Supported.

### Additional options:

UV set index	Index of the set of uv coordinates used for texture maps. Usually generated by your 3D modelling package.
--------------	---

Path	Path to the texture map on disk. The path must either be absolute, or relative to the scene's working directory.
Gamma (exponent)	Used for converting non-linear RGB values (e.g. sRGB) to linear intensity values. A typical value is 2.2, corresponding to the sRGB standard.

## ABC values / texture adjustments

Often you'll want to change a texture's brightness or contrast in the rendered image, without having to modify the texture map itself.

During rendering, Indigo modifies the source texture data according to a quadratic formula with 3 parameters, A, B and C:

$$y = ax^2 + bx + c$$

x is the input value from the texture map, and y is the output value.

As a quick example for how this equation works: If we use A = 0, B = 1, C = 0 then the value is completely unchanged; this is therefore also the default.

### A value – Quadratic

The A value scales the contribution of the quadratic ( $x^2$ ) term. This is typically not used, however it can be useful to adjust the contrast of a texture, with a value greater than 0, and/or a negative C value.

### B value - Scale/Multiplier



Texture Scale (B value) of 2.

The B value scales the contribution of the linear (x) term. This is typically used to adjust the overall brightness (for example to reduce the maximum albedo of a texture, using a value of 0.8 or so), and can also be useful to adjust the contrast of a texture, with a value greater than 1, and/or a negative C value.

### C value – Base/Constant



Texture constant (C value) of 0.2.

The C value is always added, regardless of the input texture amount; it therefore acts as a base or "floor" value. So for example if you have a completely black texture, and a C value of 0.5, it would appear as



50% grey.

## Internal Medium

A (transmission) medium defines the properties of the matter through which light travels, when it is refracted at a material interface.

For example, a green glass medium will specify moderate absorption in the red and blue parts of the spectrum, so as to leave behind mainly green light; clear blue water will specify a small amount of absorption in the red and green parts of the spectrum so as to leave behind mainly blue light. If the medium contains many small particles, such as milk, then it will also specify other properties such as the scattering coefficient, etc.

- The object has to be a closed volume. This means it cannot have any holes leading to the interior of the mesh.
- All mesh faces must be facing outwards. Check the face 'normals'.

**Precedence:** Precedence is used to determine which medium is considered to occupy a volume when two or more media occupy the volume. The medium with the highest precedence value is considered to occupy the medium, 'displacing' the other media. The predefined and default scene medium, 'air', has precedence 1.

### Basic

Typical values for glass and water lie in the range 0.003 – 0.01 (see [http://en.wikipedia.org/wiki/Cauchy%27s\\_equation](http://en.wikipedia.org/wiki/Cauchy%27s_equation) for some coefficients)

IOR	Index of refraction. Should be $\geq 1$ . Glass has an IOR (index of refraction) of about 1.5, water about 1.33. The IOR of plastic varies, 1.5 would be a reasonable guess.
Cauchy B Coeff	Sets the 'b' coefficient in Cauchy's equation, which is used in Indigo to govern dispersive refraction. Units are micrometers squared. Setting to 0 disables dispersion. Note: the render can be slower to converge when dispersion is enabled, because each ray refracted through a dispersive medium can represent just one wavelength. So only set <code>cauchy_b_coeff</code> to other than 0 if you really want to see dispersion.
Absorption Coefficient Spectrum	Controls the rate at which light is absorbed as it passes through the medium.
Subsurface Scattering	Use this element to make the medium scatter light as it passes through it.
Scattering Coefficient Spectrum	Chooses the phase function used for the scattering.

### Phase Function

The phase function controls in which direction light is scattered, when a scattering event occurs.

Uniform	Takes no parameters
Henyeey Greenstein	The Henyeey-Greenstein phase function can be forwards or backwards scattering, depending on the 'g' parameter.

### Dermis

Hemoglobin Fraction Controls the amount of hemoglobin present. Typical range: 0.001 – 0.1

### Epidermis

Medium for simulating the outer layer of skin.

Melanin Fraction	Typical range: 0 – 0.5
Melanin Type Blend	Controls the amount of eumelanin relative to pheomelanin in the tissue. Typical range: 0 – 1

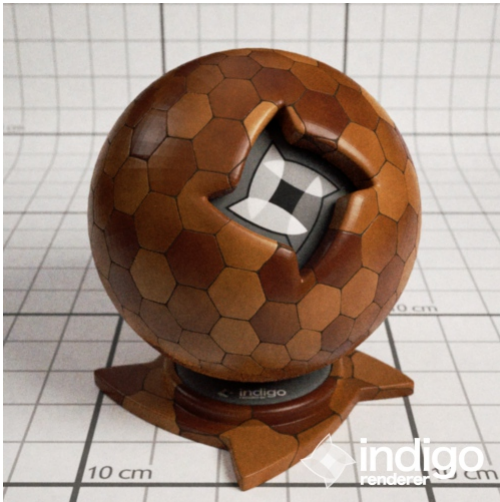
### Material Database

The online material database is located at <http://www.indigorenderer.com/materials/>

There you are able to browse and download any of the user uploaded materials, and upload your own.

Please note that you cannot upload textures that you do not have the right to redistribute.

## Indigo Shader Language



A fully procedural material made by [galinette](#).

ISL stands for Indigo Shader Language. It's a [functional language](#) that allows you to write shaders for every channel in Indigo materials. With shaders you are not tied to the restrictions of textures any more: they are procedurally computed for every point on the surface.

# Indigo Shader Language Beginner Tutorial

## Indigo Shader Language Beginner Tutorial

This tutorial covers:

- What's ISL? And why should I use it?
- What's a functional language?
- How to define functions
- Which channel expects what function parameters and return value?
- What data-types are there?
- Functional, huh? Does that mean I have to use functions instead of operators?
- How to use a shader in a material?
- Going further

### What's ISL? And why should I use it?

ISL stands for Indigo Shader Language. It's a functional language that allows you to write shaders for every channel in Indigo materials. With shaders you're not tied to the restrictions of textures any more because they are procedurally computed for every point on the surface (or in the volume, as of Indigo 2.4's ability to have volumetric shaders).

### What's a functional language?

In functional language there are only functions that are used to compute something, no variables no loops (unless you write them as a function). By functions it means functions in the mathematical sense, it calculates something and returns the result. Let's have a look at this shader function: `def doSomething(vec3 pos) real: noise(fbm(pos * 3.0, 8))` This is a function called 'doSomething', it has one parameter, a vector called 'pos' and it returns a real value. It uses two other functions, `fbm()` and `noise()`.

### Now what's happening here?

Like in a mathematical function you calculate the innermost parenthesis first. That means the vector 'pos' is multiplied by 3.0, then passed to the `fbm()` function which calculates a value of the type 'real', which is then passed to the function `noise()`, which calculates a 'real' value, which is the return value of the function 'def'. Pretty simple, isn't it?

## How to define functions

Now, let's see how to define a function. A ISL function definition always starts with the keyword 'def', needs at least a name and a return value type, and it can also have function parameters: `def name(parameters) return_value_type: [...actual function...]` Although you can give your functions an any name you want the main function in a channel always has to have the name 'eval'. Which takes us directly to the next topic: different channels expect different parameters and return values!

## Which channel expects what function parameters and return value?

There are three different channel types, wavelength dependent, wavelength independent and displacement. Wavelength dependent expects a `vec3` as a return value, wavelength independent expects a real and displacement expects real and cannot use the position in world-space (`vec3 pos`) as a function parameter. Here's a table that illustrates all that: C

Channel	Channel type	Eval function expected
Diffuse	Wavelength dependent	<code>def eval(vec3 pos) vec3:</code>
Emission	Wavelength dependent	<code>def eval(vec3 pos) vec3:</code>
Base Emission	Wavelength dependent	<code>def eval(vec3 pos) vec3:</code>
Specular Reflectivity	Wavelength dependent	<code>def eval(vec3 pos) vec3:</code>
Absorption Layer	Wavelength dependent	<code>def eval(vec3 pos) vec3:</code>
Exponent	Wavelength independent	<code>def eval(vec3 pos) real:</code>

Sigma	Wavelength independent	def eval(vec3 pos) real:
Bump	Displacement	def eval() real:
Displacement	Displacement	def eval() real:
Blend	Displacement	def eval() real:

## What data-types are there?

First of all, its very important to know that ISL does not convert values implicitly, so if a function expects an integer, you have to make sure you give pass an integer value instead of a real.

- **real – floating point number**

A real value always has to be written as a floating point number, for example 214.0.

- **int – whole numbers**

Only whole numbers, like 20 or -1545.

- **vec3 – 3 component vector**

There are two constructors for a vec3, `vec3(real x)` and `vec3(real x, real y, real z)`. The first one applies the number passed to any of the 3 components and the second one sets each component separately. You can access the three components separately with the functions `doti()`, `dotj()` and `dotk()`.

- **vec2 – 2 component vector**

Like vec3, only just 2 components.

- **bool**



A boolean value, true or false.

- **mat2x2 and mat3x3**

2x2 and 3x3 matrix, I'm not going to talk about these right now.

## Functional, huh? Does that mean I have to use functions instead of operators?

No, you don't have to use functions as operators, but you can, if you like to, are a hardcore mofo or just a little insane :) Operators are available for multiplication, division, addition and subtraction (\*, /, + and -, would you believe it?) for every data-type that supports them, but the order of operands is important sometimes, for example: `0.333 * vec3(5.2)` will not work since it expects the vec3 first. `Vec3(5.2) * 0.333` works. The equivalent functions are called `mul()`, `div()`, `add()` and `sub()`.

## How to use a shader in a material?

Your exporter most likely has an option to use ISL shaders, also, you can use ISL in the Indigo Material Editor it allows you to use ISL shaders. And if you're hardcore, you can edit the .igs file generated by you exporter and insert your shaders manually, here's how you do it: Open the .igs file and look for a material. I'll just assume we found a phong material:

```
<material>

<name>phong_mat</name>

<phong>

<ior>1.466</ior>

<exponent>

<constant>800</constant>

</exponent>

<diffuse_albedo>

<constant>

<rgb>

<rgb>0.588235 0.588235 0.588235</rgb>

<gamma>2.2</gamma>

</rgb>

</constant>

</diffuse_albedo>

</phong>

</material>
```

What you have to do is, is to replace the `<constant>...</constant>` XML elements (and anything in between) in the

diffuse\_albedo channel with this:

```
<shader>
```

```
<shader>
```

```
<![CDATA[
```

```
#paste your shader in here, oh by the way: every line starting with # is a comment
```

```
]]>
```

```
</shader>
```

```
</shader>
```

Then paste your shader in between '<![CDATA[' and ']]>'.

## Going further

For a complete list of all available functions, have a look at the ISL section in the 'Indigo Technical Reference.pdf' and the 'ISL\_stdlib.txt' in the Indigo folder. Also, more ISL tutorials are coming!

## Indigo Shader Language Tutorial

ISL stands for Indigo Shader Language, the language for creating procedural materials in Indigo. Since it's a functional language, it can be a bit tricky to create some patterns that are easy enough in an imperative language. For example, lets say you want to write a shader to make a polka-dot pattern. In an imperative language you might write some code like

```
def fillWithColour(background_colour):
    for(int x=0; x<W; ++x) {
        for(int y=0; y<H; ++y) {
            drawDot(x, y, dot_colour);
        }
    }
```

But in ISL, you can't do things this way. You have to write a function that returns the colour, and that depends only on the position and/or UV-coordinates of the current surface point being shaded, e, g:

```
def getColourForPoint(u, v):
    if( the point (u, v) is inside a dot ){
        return dot_colour
    } else {
        return background_colour
    }
```

This tutorial shows you how to use such a functional technique to create regularly repeating patterns, like polka-dots, with ISL.

Lets start by discussing the fract function.

As described in the Indigo Renderer Manual, fract takes a single real number, and returns a real number:

$$\text{fract}(x) = x - \text{floor}(x)$$

The useful thing about this function, is that it repeats regularly across the real number line, with period 1. We can use this function to create more complicated behaviour.

So lets say we want to create some stripes, such that they alternate in the U direction of the UV coordinates. Suppose we have a foreground and background colour.

Using the fract function above, we can assign the foreground colour when  $\text{fract}(x) < C$ , and the background colour when  $\text{fract}(x) \geq C$ , where  $C$  is some constant between 0 and 1. If  $C$  is 0.5, the stripes will have the same width as the background stripes.

Let's see how that looks in real ISL:

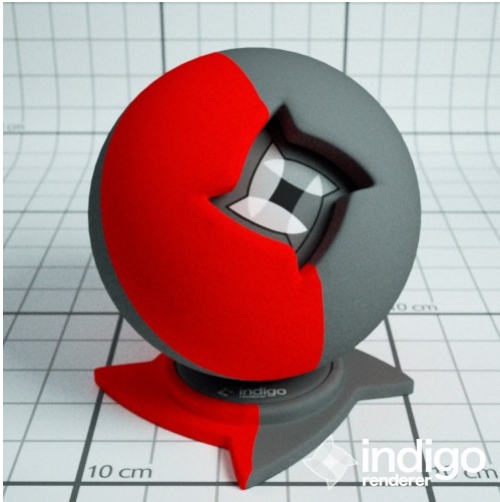
```
<material>
  <name>previewmaterial</name>
  <diffuse>
```

```

    <albedo>
      <shader><shader><![CDATA[
def eval(vec3 pos) vec3 :
  if(
    fract(doti(getTexCoords(0))) < 0.5,
    vec3(0.9, 0.0, 0.0), # Red
    vec3(0.2, 0.2, 0.2) # Dark Grey
  )
  ]]></shader></shader>
    </albedo>
  </diffuse>
</material>

```

And the resulting render:



The shader with `fract(x)`, so slowly repeating stripes.

In this example the foreground colour is red, and the background colour is dark grey. This example looks a bit weird because the stripes are large compared to the model. We can solve this problem by multiplying the UV coordinates by a number greater than one before we pass the value to `fract`, e.g we could use something like `fract(10 * x)`

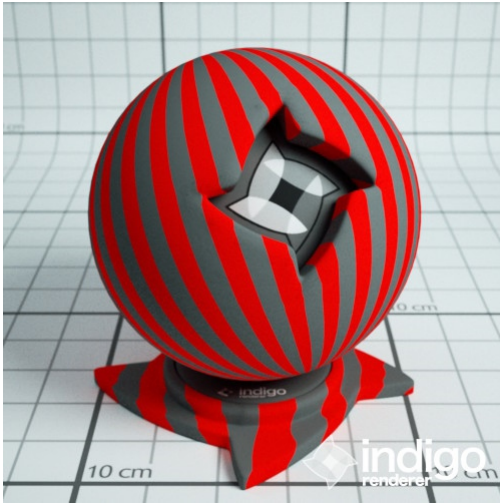
The ISL is then:

```

def eval(vec3 pos) vec3 :
  if(
    fract(doti(getTexCoords(0)) * 10.0) < 0.5,
    vec3(0.9, 0.0, 0.0), # Red
    vec3(0.2, 0.2, 0.2) # Dark Grey
  )

```

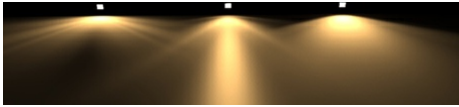
And the resulting render is:



The shader with `fract(10 * x)` so more repeating stripes.

So, at this point in the tutorial, we have more-or-less solved the problem of how to create regularly-repeating patterns, at least with respect to the U coordinate (e.g. in one direction).

# IES Lights



If an IES profile is selected, the distribution of emitted light is specified by the information in the IES profile. This is an easy way to get realistic emission profiles for a given light fixture, without creating a detailed model of the fixture.

Please note that only IES files with vertical angles below 90 degrees are currently supported.

See also the [IES tutorial](#) in the Techniques section of this manual.

Some freely available IES profiles are available to download from <http://www.indigorenderer.com/dist/ies-profiles.zip>

# Mesh Settings

There are some settings that control how a triangle mesh is rendered in Indigo - in particular the subdivision and displacement settings. Please read on for details on these.

## Instancing

Instancing is a way to replicate scene geometry or meshes without using much additional memory - each instanced object shares the original mesh data, however the material and placement in the scene can be different.

This is useful in many situations where there are multiple copies of the same mesh data (such as chairs around a table, or trees in a forest) to reduce the memory usage required for detailing an environment.

An excellent example is given below by forum user [Godzilla](#):



Image by [Godzilla](#)



## Subdivision



Subdivison off



Subdivison On

Subdivision divides each triangle in the mesh into 4 new triangles, with each subdivision level increasing the number of triangles exponentially. Therefore a large mesh of, say, 1 million triangles becomes 4 million triangles when subdivided only once. When subdivided twice it becomes 16 million triangles, and when subdivided three times it becomes 64 million triangles.

Below are available settings:

View Dependent	Subdivide only meshes visible by the camera
Max Subdivisions	How many times to subdivide the mesh. Polycount becomes exponentially larger with each subdivision level.
Curvature Threshold	Triangles will not be subdivided if their curvature is smaller than this threshold.
Pixel Threshold	Triangles will not be subdivided if they are smaller than this pixel threshold.

## Light Layers

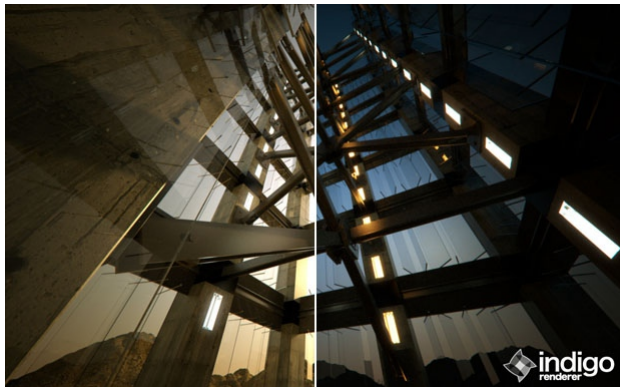
Light layers allow you to separate contributions from different lights onto different "layers". Each layer can then be manipulated separately, even after the render has completed.

You can change the brightness of each layer, or change the overall colour of each layer, or even turn each layer off completely.

### Enabling Light Layers

By default, all lights in an Indigo scene are assigned to Layer 0. This means that the HDR image will have only one layer – Layer 0. However, in the exporter for your 3D modelling program, you can change the layer that a light is assigned to. All contributions from that light will then be rendered onto that layer.

Each layer has a number of controls that you can manipulate in the Indigo GUI; please see the [corresponding section](#) in the Indigo interface section.



## Techniques

This is the tutorial and Indigo tricks section.

If you have any requests or suggestions please do not hesitate to send an email to the link provided below.



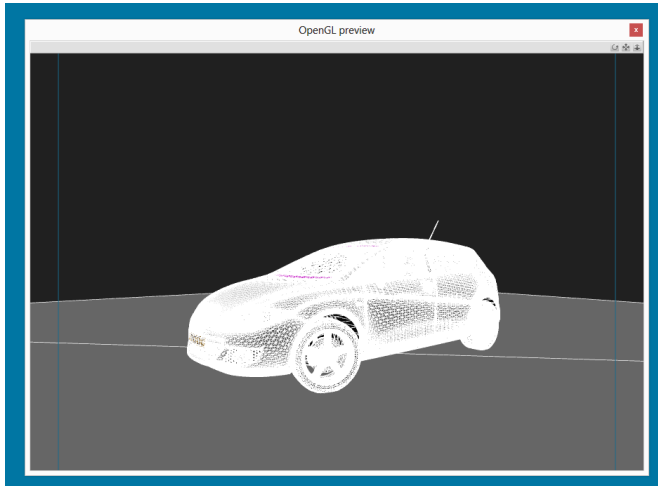
Vantage - Night by Oscar Johansson

## Compositing with shadow pass tutorial

### Requirements

This tutorial requires [Indigo 3.6.19](#) or newer.

For this tutorial you will need a car model, standing on a single quad:



You will also need a high dynamic-range environment map (in .hdr or .exr format), plus a backplate image. (which are typically in .jpg format)



Preview of background plate from [Moofe](#)



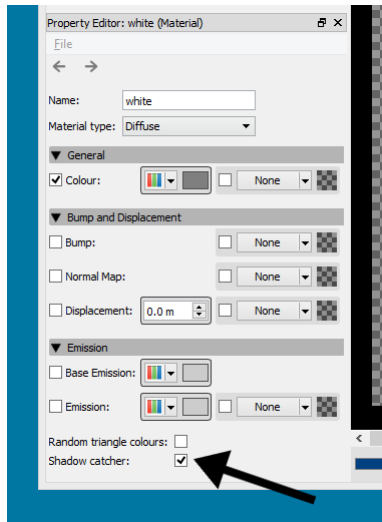
Preview of HDR environment map from Moofe

The environment type should be set to *Environment Map*, and should be using the high dynamic range environment map mentioned.

## Making the shadow pass

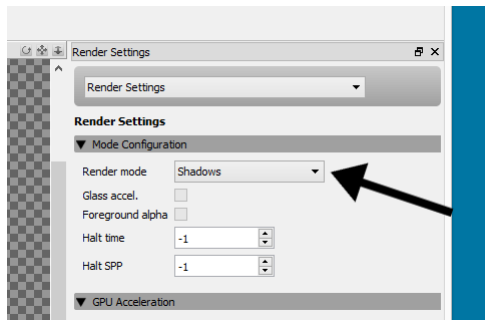
You will need to set the ground plane to be a *shadow catcher* material. To do this, first select the ground plane material with the *Pick Material* button. Then make the ground material be of *Diffuse* material type.

Then check the *Shadow catcher* checkbox at the bottom of the material options.

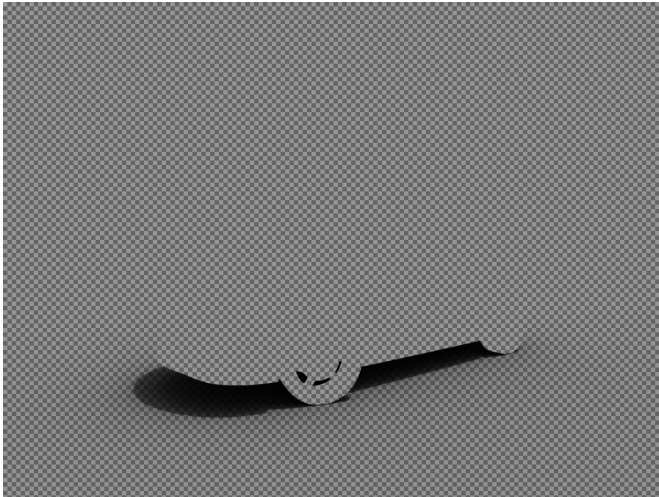


You may also be able to set the shadow catcher material option directly in your exporter.

Then change the render mode to *Shadows*:



This should produce a render like so:



For best results, you can make the car object invisible. You can do this by selecting the car object(s) with the *Pick Object* button, then checking the *Invisible to camera* checkbox, or by setting the Invisible to camera option for the object in your exporter.

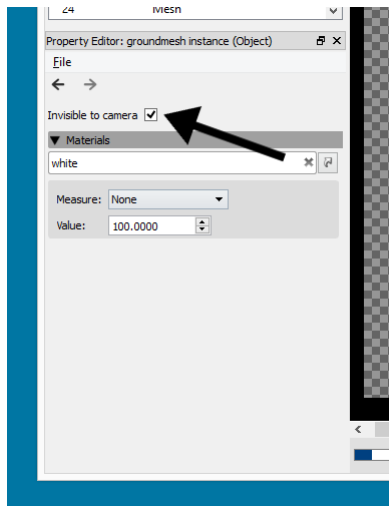
This should produce a result like this:



### Making the foreground car render

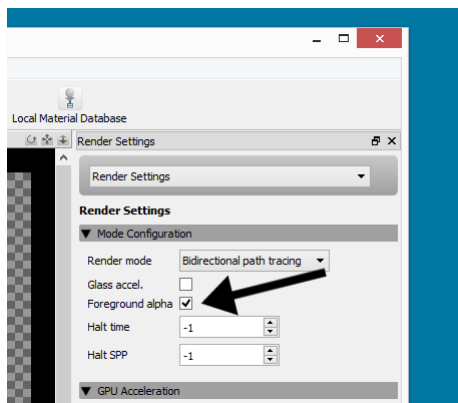
To make the foreground car render, the car object should be visible to the camera.

The ground object should be invisible to the camera - that is, it should have *Invisible to camera* checked: (Select the ground object with *Pick Object* to show these settings)



Now set the render mode to a normal rendering mode like *Bidirectional path tracing*.

Check the *Foreground alpha* checkbox. This will make only the car have non-zero alpha:



This should produce a render like this:



## Compositing the layers together

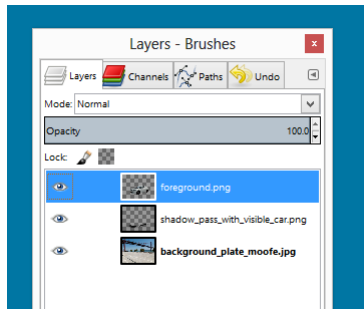
We now have 3 layers that we want to composite together:

- The background plate
- The shadow pass



- The foreground render

These layers can be easily composited together in a program such as Photoshop or GIMP:



The final result should look something like this:



Credits: Scene setup originally by hcpiter. Background plate and HDR are by Moofe.



### Compositing your scene into an environment



Scene with HDR environment map

Often it is not economical to model and create an environment for your scene to sit in, and it is a common technique to composite elements together to fake it.

There are three main techniques for doing this; the first and easiest is to use alpha compositing. The second method, which gives the best results, is to use an environment map, and the third is to use image planes (or "background cards").

## Compositing with alpha channel

The alpha channel specifies the amount of transparency present at each pixel in the rendered image, which is useful for compositing against background images in another image editing application, such as GIMP or Photoshop.

After modelling your scene, lighting it and framing it with the camera, find an image to use as a background that suits your scene. The camera angle, time of day or shadows, and the type of light (incandescent or fluorescent) are all important factors to keep in mind for a matching background image. The resolution of this image should ideally be equal to or greater than the size of the final rendered image.

### 1. Step 1: Render with alpha channel

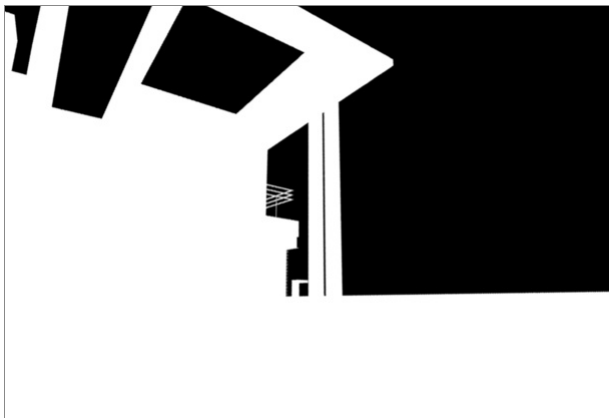
To enable rendering with alpha channel output, either click the "Foreground alpha" checkbox in the Indigo render mode settings, or enable it from your exporter package:

SketchUp   SkIndigo > Render Settings > Advanced > Tracing Method

Cinema 4D   Indigo for Cinema 4D Render Settings > Export Settings > Background alpha

Blender   Blendigo > Render Settings > Alpha render

3ds Max   Maxigo > Export > Export Scene > Alpha Mode



### 2. Step 2: Insert background image in Photoshop

You can now composite the two using an image editing tool such as GIMP or Photoshop; we'll use Photoshop here for illustration.

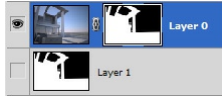
1. Put the both images together



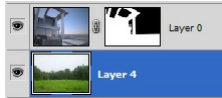
2. Create a mask for the real render by selecting the layer and clicking the add mask button. Select the alpha layer and copy it.



3. Win:Alt; Mac:Option + click the white mask to select it, and then paste the alpha render onto it.



4. Put an image under it and you're done!



Final composite

Results: Notice the Sun & Sky is still present in the reflections of the windows and looks quite out-of-place.

## HDR Environment Map

An HDR environment map is an image that fully wraps around your scene to create the effect of having a background, and also emits light. It is important that you use an HDR environment map that has good light, especially if it has the sun in it – the sun is many thousand times brighter than anything else and will only cast good shadows if it has been captured right. This technique gives the best results because all the reflections from the materials will accurately respond to the 'environment' as if it were real. But it is difficult to find a good HDR environment map for free – and can be difficult to make your own. I am using an HDR environment map found from [openfootage.net](http://openfootage.net)

1. Download an .EXR or .float HDR map. Indigo does not support .HDR files, but there are good converters out there. Try [Picturenaut](http://Picturenaut.com).



Italian field from openfootage

2. In your render settings, change the environment to Environment Map and set your HDR environment map.
3. Render! If the environment map is too dark, or not positioned correctly, you can change the gain for brightness, and rotation to change its position.



Final render

Results: The scene is effected by the light of the environment map and you can just see reflections of trees from the map in the windows.

## Image Planes

This technique is a bit of a cheat really. It involves taking a picture of a background that you would otherwise composite, and texturing them onto a large flat plane. The trick is to get a large enough image that it fills the whole background of the render.

1. Create a massive plane that sits directly behind your model, and fills the camera's view. Texture it with your background image and resize it so it fits nicely.
2. Copy this front image plane and mirror it behind the camera to create a back-plane.



Sketchup scene with massive image planes arranged

3. To create a convincing background, light must come from the image planes, or they will catch shadows and look fake. Add emission to the textured material, as an emission texture. This will emit light from the surface based on the texture itself. In SketchUp, simply change Emission to SketchUp. I've also set the emission color to white and power to 200.
4. Render with sun & sky for good shadows, or you could even just use background colour.



Render with only a front image plane



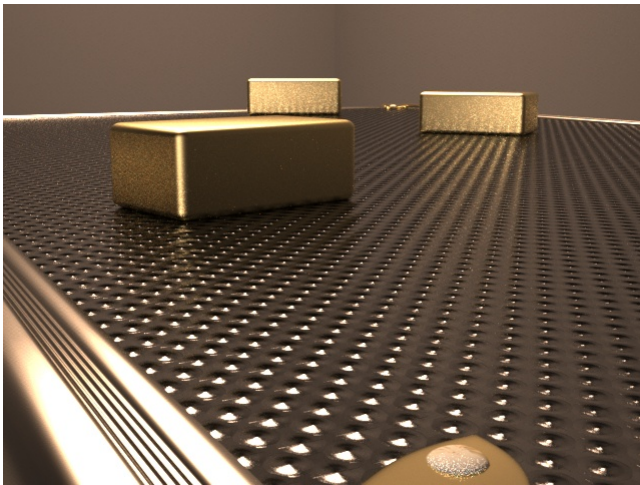
Final render

Result: (excuse the tonemapping) The bottom has reflections in the windows and a mostly convincing background. It looks alright.

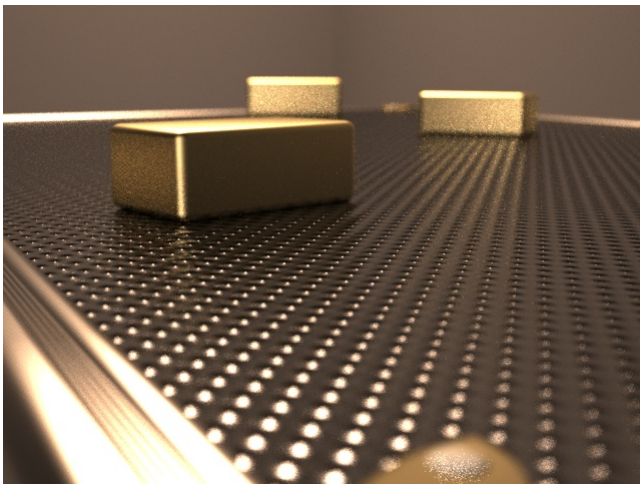
## Depth of Field

Depth of field is a phenomena that causes certain parts of an image to blur based on its distance from the camera. It is exploited as a photographic technique for a variety of reasons. A shallow depth of field can be used to accentuate certain parts of a scene, give more detailed information about a subject by exposing just a portion of it, or to remove a distracting background. A wide depth of field puts all elements in focus.

An image is always sharpest at the cameras focal distance, but the depth of field is controlled by the camera's aperture, measured in f-stop. The smaller the f-stop number, the smaller the depth of field; the larger the f-stop, the larger the depth of field.

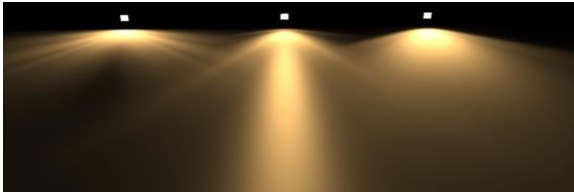


Wide depth of field, f/22



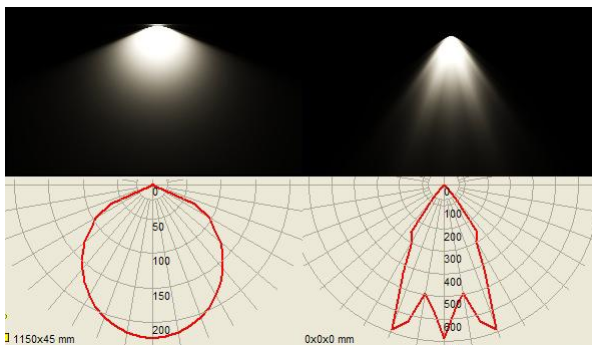
Shallow depth of field, f/2

## IES Lights



IES stands for the Illuminating Engineering Society, which has defined a file format for describing the distribution of light from a light source.

Using only a small, simple emitter such as a single quad, an IES profile will shape the distribution of light emitted from it to match that of a much more complicated light fixture, such as in the examples below:



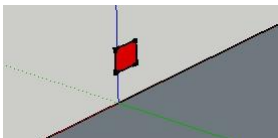
Digital Photometric data. With the basic Indigo profile Left, and an IES profile Right.

While Indigo is capable of creating real refractions of an accurately modelled light fixture to create this effect, it is far easier to use an IES profile, and the result is much the same. Many manufactures provide IES files for their lights, and it is a great way to add realism to your scene.

You can use an IES viewer to preview these profiles. There is a very good viewing program made by Andrey Legotin that can be found here: <http://www.photometricviewer.com/>

## Setting up the scene

Firstly, we will set up a simple scene to show off the light effects. For this I have made a flat ground plane, to catch the light, and a small, single plane mesh above and perpendicular to it.

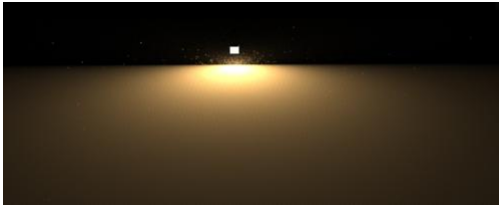


A single plane mesh raised off the ground

The plane will be our light-source, so select the it and assign it an emitting material in your Indigo plugin. Since we are working with artificial lighting, go to your environment settings and disable the sun and set a black background.

Hit render and you will get the standard Indigo light render like so:

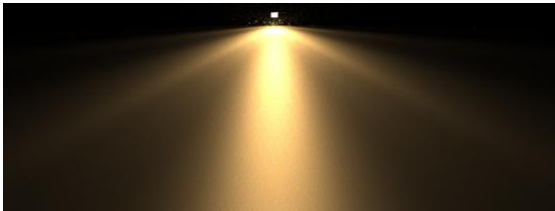




Render with no IES profile

### Adding an IES profile

Open the material editor of your Indigo Plugin, under Emitter Attributes you will find the IES path. Download the zip of IES files here <http://www.indigorenderer.com/dist/ies-profiles.zip> and link to one. Hit render to view the new IES profile in your scene.



Render with IES profile

Note that Indigo does not support IES profiles with a vertical angle of more than 90degrees and will give an error if attempting to do so.

## Optimizing your scene

There are many things you can do to make Indigo render your scene faster, and cleaner, often without compromising quality.

### Scene Settings

The first thing you can do is make sure the scene you are rendering is suitable for the power your computer offers.

### Light Layers

If there are many light layers used, it will consume a lot of memory. If this exhausts all physical memory the rendering will become extremely slow due to disk access. For more information on light layers, see [this section](#) of the manual.

### Render resolution

A larger screen resolution, and larger Super Sample factor, will take longer to render, simple because there are more pixels to cover. See [Imaging Settings](#)

### System Requirements

Check the [recommended system requirements](#) to see how your computer compares.

### Understand your Render mode

Indigo has several render modes, which control how it chooses where to render, and how. Which mode is appropriate to use varies from scene to scene. Scenes with a lot of specular materials, for example, should be rendered with MLT enabled.

See [Render Mode](#).

### Scene design

Some things about the way the scene is made can slow a render down, and likewise speed it up.

### Over-saturated colours

If a colour (this includes any colours in texture maps) is over-saturated, it creates an unrealistic conservation of energy as light bounces off it. This can create strange artifacts known as 'fire-flies', along with other problems.

Make sure that no colour is more than 80% saturation, which translates to an RGB no higher than that of 204,204,204.

## Glass and liquids

Glass and liquids are one of the things that make Indigo renders stand out. However they can be computationally intensive, and therefore it pays to be aware of how they can affect render times.

### Try MLT Bi-Directional path-tracing

MLT is good for clearing glass and liquids, and combined with bi-dir it can help those otherwise hard-to-render areas

### Glass around a light source

While it is important to model your scenes as they would appear in the real world to realistically render with Indigo, modeling a light bulb as glass around a light source can greatly lengthen render times. This is because every single ray of light that is emitted has to pass through the glass and reflect & refract its way through it. Effectively this means that all light in the scene is now a caustic. Simply giving the light-bulb's outer shape an emitting material will give the same effect as a real light-bulb, without the extra calculations.

### Glossy-Transparent

This material is especially difficult to reproduce efficiently, because of the complex effects it creates. It is strongly suggested that this material not be used to transmit the sole light into a room (as a sky-light for example) See [Glossy Transparent](#)

### Glass Acceleration

This feature allows glass sheets/panes to be rendered extremely effectively. If the scene has sheets or panes of glass, enabling this will reduce your render times. See [Render Settings](#)

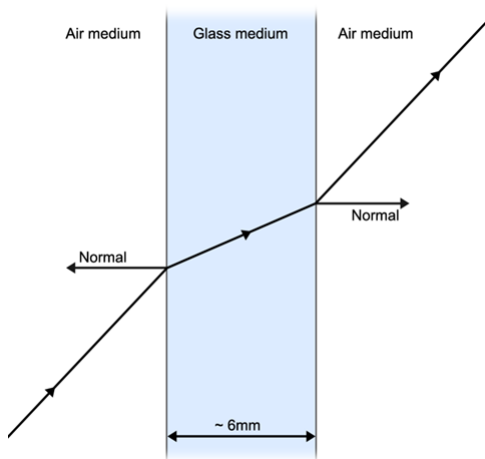
## Physically correct modelling of glass

Glass is realistically simulated in Indigo, as a [\(transmission\) medium](#), which requires a little more care when modelling scenes compared to "traditional" renderers. In this section we will focus on glass, however the principles covered here extend to other media (such as water, fog, etc).

We begin with a concise statement of the requirements, and then go into some details. The executive summary is:

When modelling glass in a scene, it cannot be represented as a single surface; there must be at least two (for entering and exiting), and the glass medium occupies the space between them. Furthermore, the surface normals in all cases must point outwards.

Let's start with the first part: glass must be represented as a volume. The following diagram illustrates how light interacts with a glass pane:

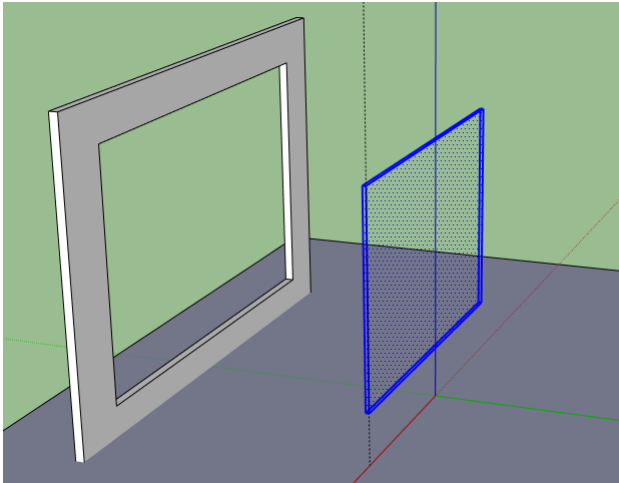


As light travels through the glass medium, [its brightness is diminished](#); this is what gives glass its colour, and it is physically due to the thickness of the glass (and the medium properties), not a "glass colour" modifier at the surface.

If either of the glass faces are not present, the glass volume becomes infinite (since light cannot exit the medium). For example, rendering a house with single-surface glass panes will result in the interior being essentially black, as the light will have lost most of its energy travelling through several metres of glass.

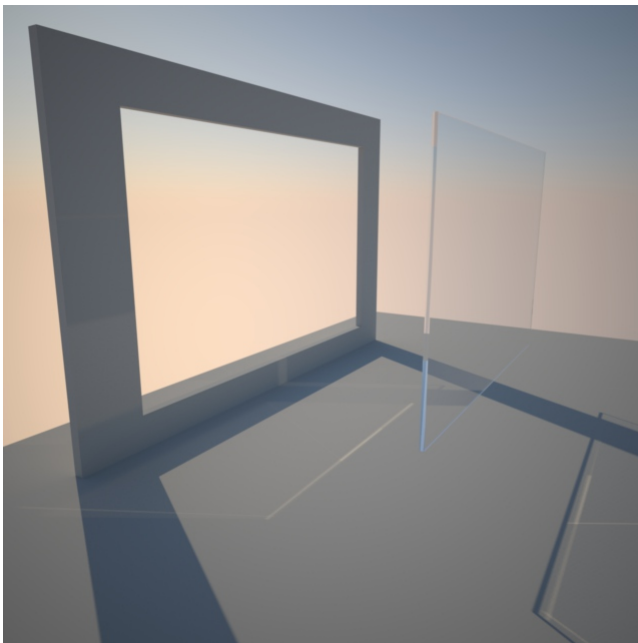
Now that we have described the problem with single-surface glass, we show how to correctly model glass panes. SketchUp is used here, however the same principles apply to all 3D modelling packages.

The simplest way to correctly model a glass pane, is to take a thin box (e.g. a cube squashed in one dimension) and apply an Indigo glass material to it:



SkIndigo has a default glass material, however if you wish to make your own then you should use the Specular material type with IOR 1.5 and transparency enabled.

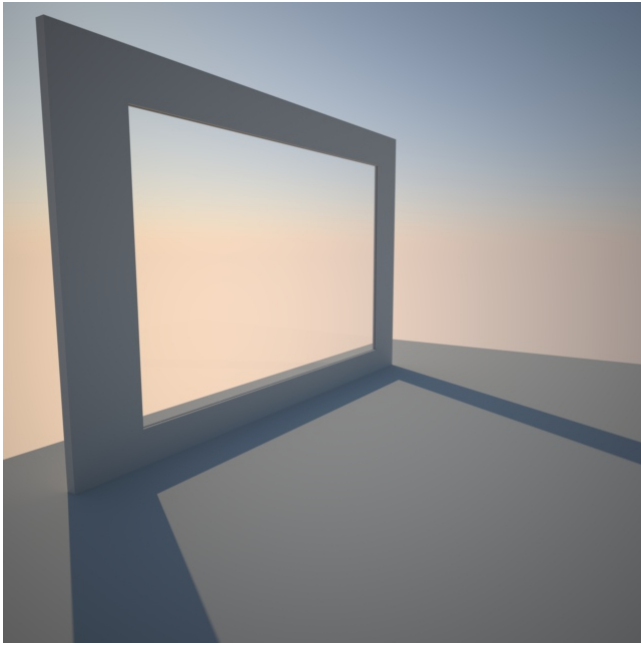
Here is the result rendered in Indigo:



The next step is to insert the glass box into the window frame. Note that we make the glass pane bigger than the opening in the window, allowing the glass faces to intersect the frame, so that the medium is perfectly contained without tiny gaps.

If you want to be more efficient about this, you can remove all faces except for the front and back - since these should be contained entirely inside the solid wall, they cannot affect the rendering and can be safely removed.

With the glass inserted to the frame, we obtain the desired result:



## Indigo for SketchUp

This is the manual for Indigo for SketchUp - the Indigo exporter for SketchUp.



Micro living concepts by sking

# Installation

## 1. Download and install Indigo

Download Indigo RT or Indigo Renderer for your system and install to the default location on your system.

You can download Indigo from here: <http://www.indigorenderer.com/download/>

## 2. Download and install Indigo for SketchUp (SkIndigo)

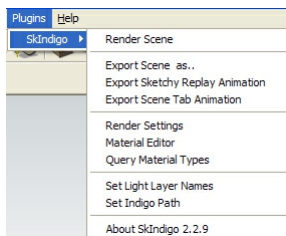
On Windows, the Indigo installer comes bundled with the latest version of SkIndigo, the installer for which should launch from the main Indigo installer.

Mac users must download the RBZ file and install it manually (since there is no installer for those platforms); the latest version of SkIndigo can be downloaded from <http://www.indigorenderer.com/documentation/sketchup>.

Please see <https://www.indigorenderer.com/documentation/manual/indigo-sketchup/inst...> for instructions on how to install the RBZ file.

## 3. Check your SkIndigo Installation

Restart SketchUp after installing SkIndigo and you should see see "SkIndigo" become available under the Plugins menu.



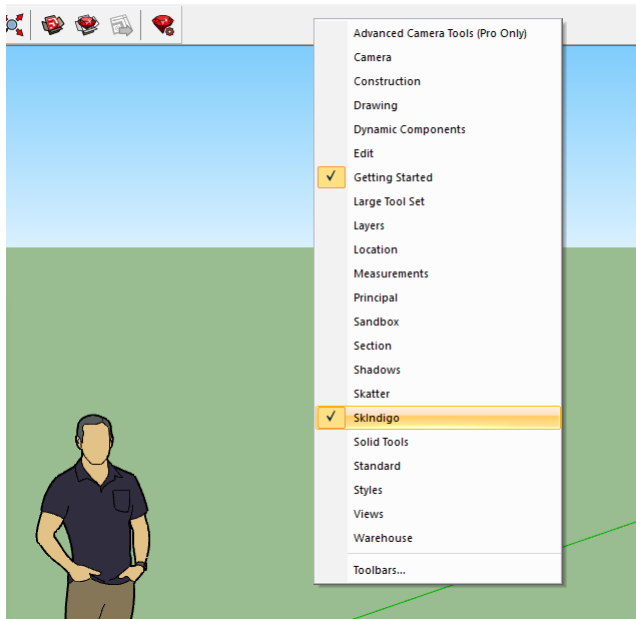
The SkIndigo Plugins menu

You are now ready to use SkIndigo.

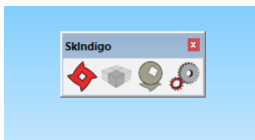
## 4. Enabling the SkIndigo toolbar

The SkIndigo toolbar should be enabled by default, however if it is closed, you can re-enable it by right clicking on the toolbar area, and ticking SkIndigo from the context menu.





The SkIndigo toolbar should now be visible:



## 5. Getting help with SkIndigo Installation

If you have any issues installing SkIndigo, please email us at [support@indigorenderer.com](mailto:support@indigorenderer.com).

## Installing SkIndigo from a RBZ file

The Indigo for SketchUp (SkIndigo) extension is available in the standard .rbz format for SketchUp extensions. These steps will describe how to install the extension.

### 1. Download and install Indigo

Make sure you have Indigo Renderer or Indigo RT installed first.

Download Indigo for your system and install Indigo to the default location on your system.

You can download Indigo from here:

<http://www.indigorenderer.com/download/>

### 2. Download the SkIndigo .rbz file

If you haven't already, download the SkIndigo rbz file from the [SkIndigo page](#)

### 3. Install the .rbz File

Start SketchUp.

For SketchUp 2017 or later, select Windows > Extension Manager.

For SketchUp 2016 or earlier,

On Mac, select SketchUp > Preferences.

On Windows, select Windows > Preferences.

Select the Extensions item on the left of the Preferences dialog.

Click on the Install Extension... button.

Select the SkIndigo-3.x.x.rbz file you just downloaded.

Click the Open button.

Click the Yes button in the 'Do you trust this Extension' dialog.

Restart SketchUp.

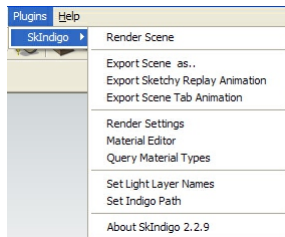
### 4. Check your installation

Once you have restarted SketchUp, SkIndigo should be installed and available to use.

You can check it by selecting Plugins > SkIndigo > Render Scene

## The Plugins Menu

The Plugins Menu is the primary place to access SkIndigo functionality, open dialogues and start rendering.



Render Scene	This exports the scene to an Indigo Scene File and launches Indigo to start Rendering.
Export Scene as...	Exports the current scene as an Indigo Scene File and prompts you to choose a location to save to on your computer. This is handy if you want to send a scene file to someone else, or upload it to a render-farm.
Export Sketchy Replay Animation	Exports an animation generated by Sketch Replay. You should get the latest version of SketchyPhysics (a plugin by Chris Phillips) to use this feature. For every object that you wish to animate, you must right click and 'Enable Instancing' for that Group or Component.
Export Scene Tab Animation	Exports each scene tab as a separate frame in an animation. Creates Indigo Scene files for every frame and saves a batch file that progresses through them. Must have Halt (Render settings > Advanced) set to stop the rendering frame after a certain amount of seconds, or samples per pixel has been reached.
Render Settings	Opens the Render Settings window for configuring the Indigo export.
Material Editor	Opens the Material Editor window for creating and modifying Indigo materials inside SketchUp.
Query Material Types	Pops up a dialogue with all of the Indigo Materials in the current scene and the types of each material. Also useful for checking which material are emitters.
Set Light Layer Names	Specify names for the Light Layers. Useful for later reference.
Set Indigo Path	If SkIndigo does not find Indigo in the default folder, then you can specify its location here.

## Toolbar

After SkIndigo has been installed, a small toolbar is added to the SketchUp user interface for easy access to commonly used export functionality from the menu:



Render with Indigo



Export the SketchUp scene to Indigo format and start rendering it in Indigo.

Render Selected  
Entities



Export and start rendering only the currently selected entities in your SketchUp scene.

Material Editor



Open the SkIndigo material editor.

Render Settings



Open the SkIndigo render settings dialog.

### Right-Click Menu

SkIndigo adds some features to the context-sensitive right-click menu in SketchUp.

## Instancing

Instancing is a feature which allows many instances (or copies) of a mesh to be represented in the scene, while only actually storing the mesh once (thereby saving a lot of memory compared to straight-forward duplication).

There are two ways this can be achieved in SkIndigo:

### 1. Proxy Instancing

A proxy is a reference object that references a more complex object with a usually simpler copy, in order to keep the overall scene geometry low, while allowing the render geometry to stay high.

1. Create the object you wish to copy, create a component out of it (right click, select 'Make Component') and name it "objectname"
2. Create a new object to use as a proxy (a cube will do) and create a component out of it named "objectname\_dummy". It is important that it has the name of the original component followed by "\_dummy"
3. Copy and manipulate these dummies around the scene and on render they will be replaced by the original component.

### 2. Component Instancing

Copies of components in SketchUp will export using Indigo instances.

1. Create the object you wish to copy, create a component out of it (right click, select 'Make Component'), with any name
2. Copy and paste the component object.

Component copies will be exported as Indigo instances - each component copy will only use a small amount of memory.

## UV Mapping

UV mapping is the process of modifying the texture map to fit the model. SkIndigo supports 4 UV maps per mesh face, which means you can have the texture map aligned one way, the bump a different way, and the clip map another way on every surface in your scene.



SketchUp has a basic UV positioning function that is used to manipulate the textures. It is also very important that the faces are facing the correct direction, if the UV mapping function is not listed in the Right-click SkIndigo menu, then try reversing the faces.

1. Here is an example run-through on how to set a textured object with a differently positioned bump map
2. Set the desired bump map as the albedo for positioning purposes.
3. Set the bump map position with the SketchUp tools (Rightclick > Texture > Position)
4. Rightclick > SkIndigo UV Mapping > Save UVs to set 2
5. Change the albedo to the actual albedo and the the bump to the bump.
6. Position the Albedo with the texture tools and save it as UV set 1.

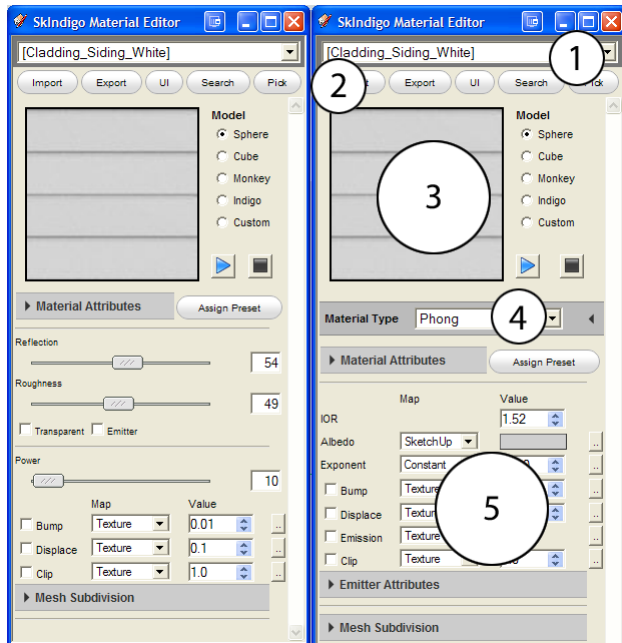


Selecting the UV set in the texture editor

7. In the SkIndigo Material editor, open the Texture Editor for the albedo and change the UV Set to 1. Set the bump map's UV Set to 2.
8. Now it will use different texture positions for the albedo and bump map. Render to test.

## Material Editor

This is where you can add the unique Indigo material settings to your objects so they behave realistically. You can find it via: Plugins -> SkIndigo Material Editor. See [Indigo Materials](#) for full information.



To work with an Indigo material, simply open up the SkIndigo Material Editor and apply any SketchUp material to an object. The painted material will be selected in the SkIndigo Material Editor. There are other ways to select a material for the SkIndigo Material Editor:

- Use "pick" from the SkIndigo Material Editor
- Select a face and in the right-click context-menu select the name of the material

### 1. List of Materials

Here is where all the textures applied in your SketchUp scene are listed, choose the desired one to start creating a material out of it.

### 2. Main Menu

**Import** Import a Indigo Material (.IGM). Use to bring in materials from the Material Editor.

**Export** Export this material as an IGM or PIGM file which can be opened in the Indigo Material Editor or uploaded to the Indigo Material Database.

**UI** This switches between the simple mode of SkIndigo, and the normal mode (See above). The simple view takes your settings and converts them as best as it can to normal settings.

**Search** Search the online database and download materials

**Pick** Use a picker tool to select materials



### 3. Preview Pane

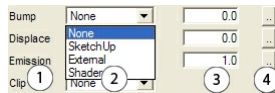
Shows the current material, the Preview function replaces this with a rendered scene with the material placed on the chosen model (to the right). Make sure you stop the preview when you are finished previewing the material as this can greatly slow down your computer.

### 4. Material Type

Select the Material Types from this drop-down list. Each has its own attributes that are listed below. Assign Preset: Choose a preset material from the list provided. This will replace any of the current material settings. See [Material Types](#).

### 5. Material Attributes

Here is where the material attributes live. They allow you to add more details to your material. The list will change depending on the Material Type selected. See [Material Attributes](#).



#### 1. List of Material Attributes

#### 2. Map

Where the information is sourced from. Depending on which map type is selected, the "..." on the right will open different editors.

Constant A constant value.

SketchUp Use the SketchUp texture or color.

Texture Use an external (non-SketchUp) texture map.

Shader Define a material using the Indigo Shader Language (ISL)

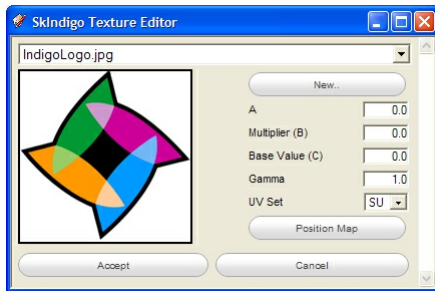
#### 3. Values

Changes the common value of this attribute. Usually a multiplier.

#### 4. Map Editor

There are several editors that will open depending on the map set.

##### ■ SkIndigo Texture Editor



**Position Map:** First, select any number of faces in your model. Clicking this button will apply this texture to the selected faces so you can then position the texture using the SketchUp texture positioning tools. Once you have positioned the texture, you can save the UV set using the right-click context menu. Be sure to paint the desired material back to the selected faces before rendering.

See [Texture Maps](#)

### 6. Emitter Attributes

See [Base Emission](#) and [Emission](#)

### 7. Media Attributes

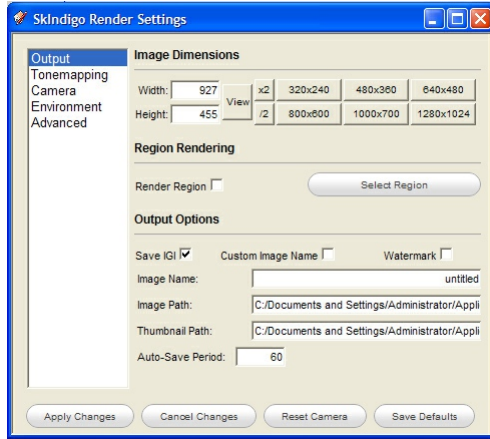
See [Internal Medium](#)

### 8. Mesh Subdivision

See [Subdivision](#)

## Render Settings

The Render Settings Window configures the scene for export to Indigo. Find it at Plugins > SkIndigo > Render Settings



**Apply to model** Apply the settings to the current scene.

**Apply to Scene** When you select the scene, the settings will be loaded and used for rendering.

**Reset Camera** Resets any changes to the camera.

**Save Defaults** Saves all current render settings as default starting settings.

### 1. Output

Image Dimensions	Configures the width and height of the render to be created. Keep in mind that the free version of Indigo can only render images up to 1000x700 pixels in size; you will need to order a licence to use higher resolutions.
Region Rendering	Region rendering allows you to render only a small part of the scene. This is similar to moving the camera, but is useful when you need to focus the render on a part of the full image.
Save to .IGI	Saves an IGI file for each rendered scene. On by default.
Custom Image Name	Enable this to use a custom name for the PNG image that is automatically saved when rendering.
Watermark	Enable this to add the Indigo logo to the rendered image. The free version of Indigo will always do this, regardless of this setting.
Image Name	To use this image name, the Custom Image Name checkbox must be checked.
Image Path	Renders will be auto-saved to this directory.
Thumbnail Path	Material preview images will be saved to this directory.
Auto-Save Period	Enter the time interval in seconds that the rendered image will be saved.

## 2. **Camera**

See [Camera](#)

## 3. **Tonemapping**

See [Tonemapping](#)

## 4. **Environment**

See [Environment](#)

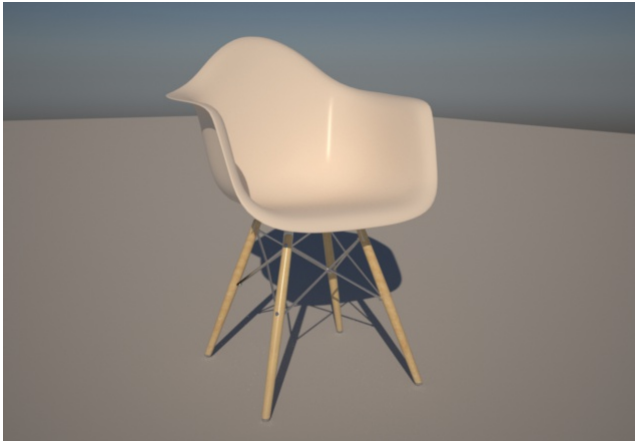
## 5. **Advanced**

See [Indigo Render Settings](#)

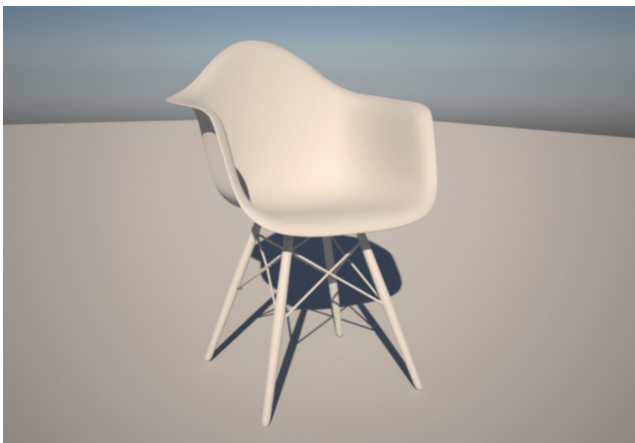
### Clay Mode

If the SketchUp Face Style is set to Monochrome, SkIndigo will export the scene in *Clay mode*. This is where all the materials are replaced with a grey diffuse material. This can be helpful for checking geometry and lighting without being distracted by materials.

To enable clay mode, Go to the *View* menu in SketchUp, then select *Face Style > MonoChrome*



Clay mode disabled



Clay mode enabled

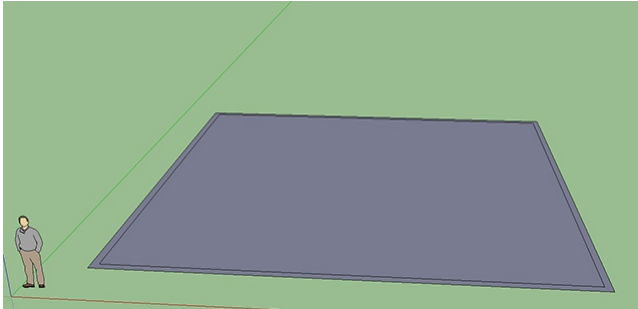
### SkIndigo Tutorial

This tutorial goes over generating a simple scene and lighting it a few different ways. The scene will be a typical German apartment with white walls, wood flooring and some furniture from Ikea. We won't be modelling the garage with an Audi, dog called Schatzi and traditional lederhosen.

This tutorial was done for SketchUp 2013 free version; your SketchUp may look slightly different, and basic familiarity with SketchUp (e.g. navigation and selection) is assumed.

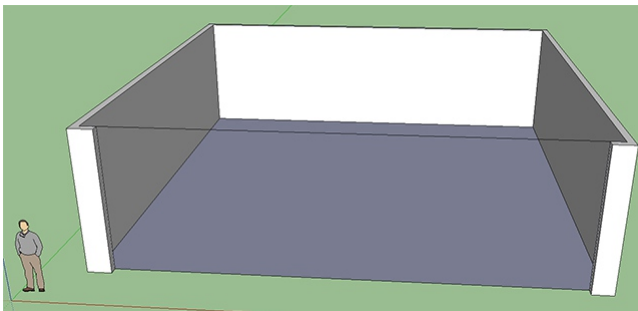
## Step 1. Create the basic room

First up we will model the room. Start by drawing a rectangle on the ground and another rectangle just inside it. Your rectangle should be 12 metres by 12 metres, look at the dimensions box in the bottom right of your window to see the size as you drag the box out.



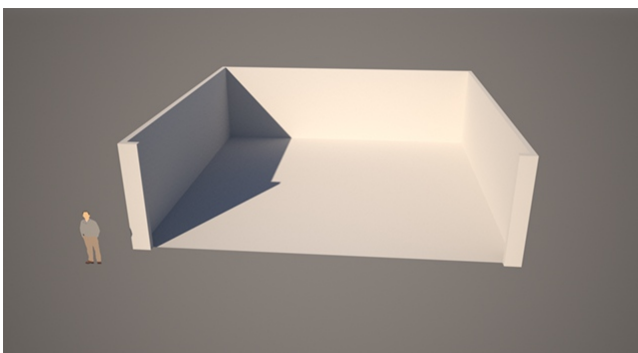
The ground plan of our room

Use the push/pull tool to make the walls 3.5 metres high. We won't put a roof on the box just yet so that we can see inside it. The next step is to put a big floor-to-ceiling window at the front of our box: use the rectangle tool to draw a rectangle on the front of the box, then use the push / pull tool to push the new rectangle inwards until the wall is paper thin.



Our room with a paper thin front wall.

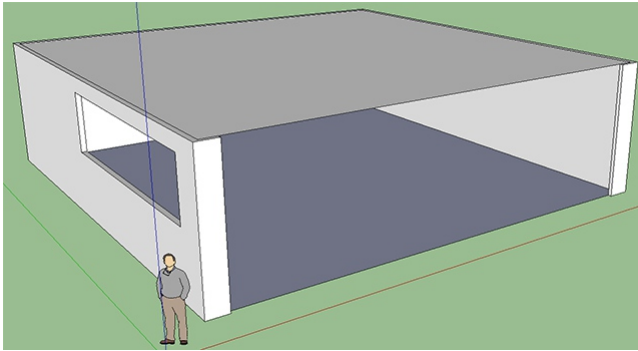
Now click on the paper thin wall and delete it. Don't delete the line at the top of your room, we'll need that in a second. Now press Plugins → SkIndigo → Render Scene and you will get a render like this:



Derrick standing outside his new house, in free space.

## Step 2. Add window and roof

For the next step, we will add a roof and a window and set a wooden texture on the floor and a glass window. Start by using the rectangle tool to enclose the roof. Then draw a rectangle on the wall of the left hand side and use the push / pull tool to push the rectangle through to the inside. Delete the paper thin wall that is left and you should have a window hole like so:



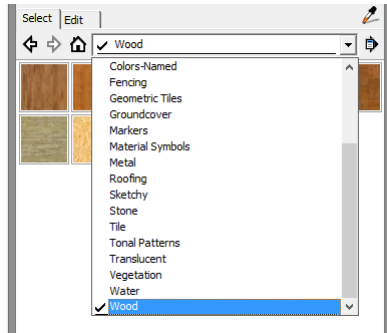
Our box room with a window and a roof.

As a final step, use the Move tool to put Derrick somewhere side the house.



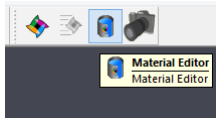
## Step 3. Paint a wood texture on the floor

Next, press the Paint Bucket tool and the SketchUp materials dialog will open. Select wood from the selection box:

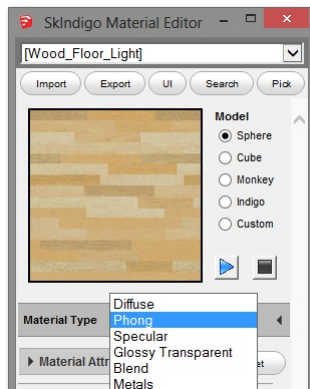


Now choose wood\_floor\_light as a texture. Then click on the floor. Your floor will now be textured, however to get a more realistic appearance for a varnished floor, we need to change the material type to Phong.

To do this, we first open up the SkIndigo Material Editor from the SkIndigo toolbar:



With the material open in the SkIndigo Material Editor, change the material type to Phong:



Zoom in the camera a little and hit render – your house should look like this:



Derrick in his new house with a roof, window and a floor.

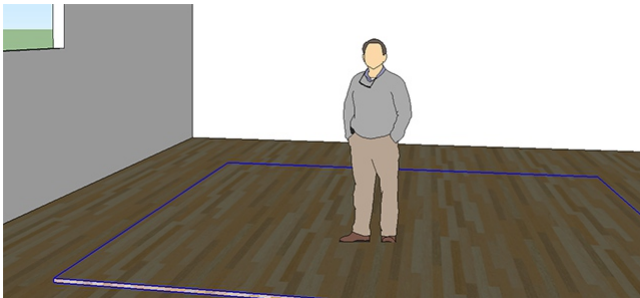
Now right click on the floor and select Texture → Position. Rotate the wood texture 90°. You can also scale the wood texture if you want.

## Step 4. Add some carpet

Carpet is a tricky thing to model because it has so many individual fibres. The best way of creating a carpet in SkIndigo is to use what is called a displacement map.

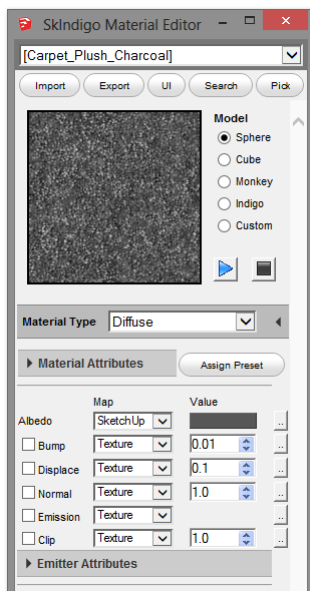
Start by drawing a rectangle on the floor and using push/pull to make it into a box of 3cm height. Then use the select tool Edit → Make Group.

When we add a displacement map, it will make our "carpet box" very bumpy, which means the edges of the box won't line up. To prevent gaps appearing, right click on the carpet and select Soften / Smooth Edges, choose a value of 90 degrees between normals and press enter.



Our carpet box ready to texture map.

Next, using the paintbrush tool, select the Carpets and Textiles set and the Carpet\_Plush\_Charcoal texture. Apply it to the carpet box. Right click on the carpet box and choose SkIndigo → Edit [Carpet\_Plush\_Charcoal]. The SkIndigo Material Editor will open:



SketchUp material editor

Note that the Albedo channel (which means the color of the material) is already set to the SketchUp carpet texture.

Now, change the displacement map to "SketchUp" which will tell SkIndigo to use the current SketchUp texture as the displacement map. Then change the value to 0.05 which will give a maximum displacement of 0.05 meters where the map pixel value is pure white. Be sure to enable displacement mapping by clicking on the checkbox.

Press Plugins → Skindigo → Render Scene, you may note that the carpet looks all triangulated and bumpy. You may need to increase the "detail" of the carpet by adding more subdivisions. Right-click the group and, from the Edit Active Mesh dialog, increase Max Subdivisions to at least 9. Also, uncheck the box for "View Dependent". This setting will decrease the amount of subdivision for geometry that is farther away from the camera, but we'll leave this optimisation for now.

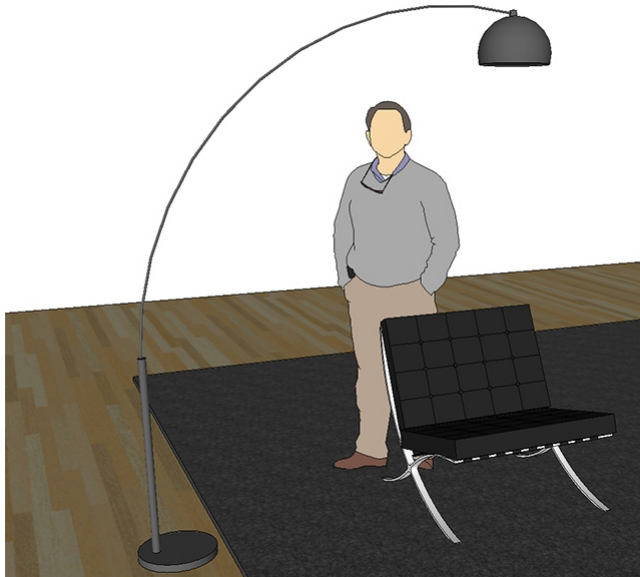
Render the scene again.



Derrick admires his nice grey carpet.

## Step 5. Add a chair and lamp

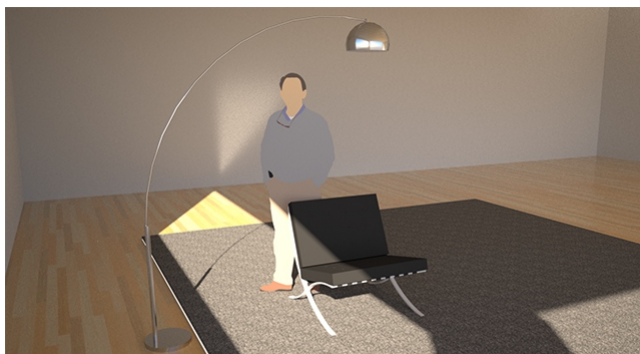
Use Windows → Components menu to show the components window. Search for Barcelona chair and insert one into the scene. Then search for Kare 5701 (a lamp) and insert it into the scene too.



Scene with a lamp and chair added.

In the Material Window click New Material to create a new material. Name this material "Chrome" and apply it to all of the surfaces of the lamp (Use SkIndigo → Edit [Chrome]). Then, in the SkIndigo Material Editor, change the material type to Metals, and select the Chrome preset from the dropdown box.

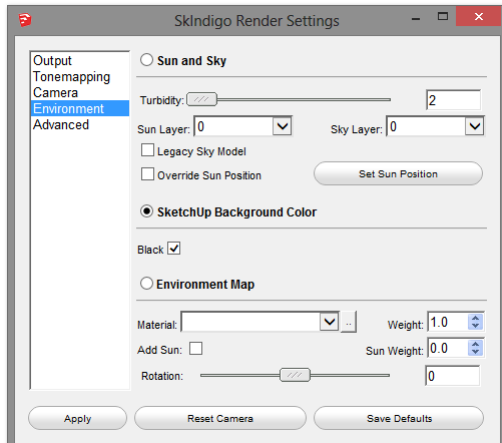
If you now render the scene, it should look like this:



Lamp has a shiny material applied.

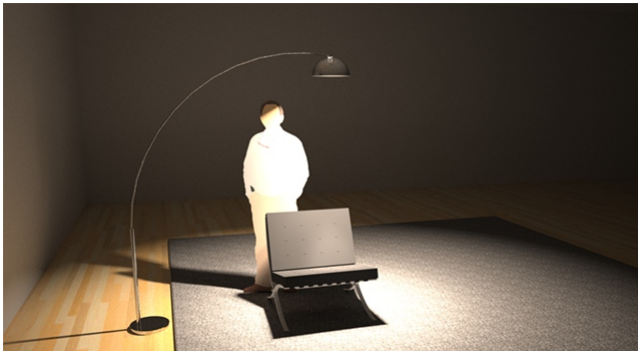
## Step 6. Adding nighttime lighting

Now we will try adding a lightbulb inside the lamp and taking a night scene. Start by turning off the sun by going to Plugins → SkIndigo → Render Settings, then Environment. Select SketchUp background color and make sure Black is selected like so:



Now we need to add a light inside the lamp. Double click the lamp to edit it, then look inside the lampshade and create or select the lightbulb inside it (your lamp may look slightly different, you may have to create the lightbulb yourself).

Create a new material called Lightbulb, then right click on the bulb SkIndigo → Edit [Lightbulb] material, then set the Albedo to be a constant black (since we don't want the lightbulb itself to reflect light), and set the Emission layer to 0 in the Emission section of the material editor.



Our house illuminated from a single 5000K emitter.

### Step 7. Finetuning

The trick to getting really realistic renders from SketchUp + Indigo is to spend time tweaking your materials until they look just right. In this example we used models from the Google Warehouse that are relatively low in polygon count, so don't look ultra realistic, but by carefully editing the materials used on the models you can make the scene look better and better.

One of the advantages of Indigo is that if you set a 100 Watt lightbulb in a lamp, you can see how the light will fall off around the room, useful for doing lighting analysis – will you need more light fittings in the corner of the room?

To increase the realism of this scene, you could:

- Increase the Mesh Subdivision of the carpet to 10. This will make the carpet seem finer grained and more "fluffy".
- Add a bump map to the floor material, of height 0.1 centimetres, to simulate the grain of the wood.
- Reduce the "exponent" of the chrome material to make the floor lamp less reflective.

As you can see there are many options that you can tweak to get the best possible results out of Indigo.

Creating ultra realistic scenes that look like something from the real world is usually achieved by recreating all of the models in the scene with accurate geometry, and then spending 30-40% of your time modifying materials in the scene to ensure that they are as realistic as you would like them to be.

We hope you have enjoyed this brief introduction to SkIndigo.

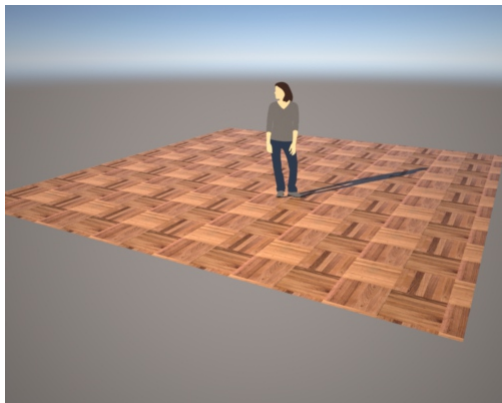
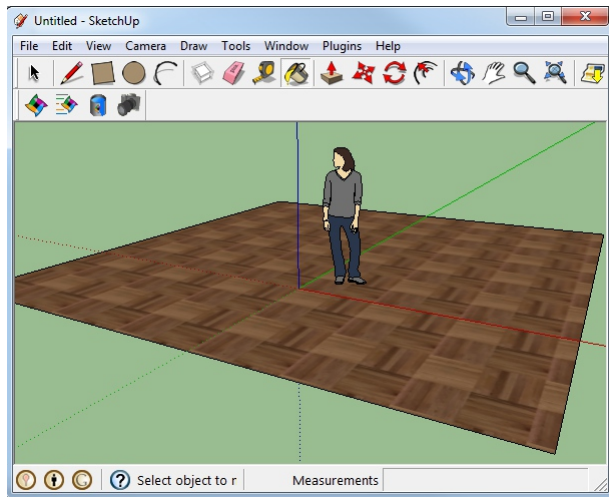
## Materials



## Texture Scaling

Often you will need to change how rapidly a texture or material is repeated over a surface. This is referred to as texture scaling, or sometimes as UV-scaling.

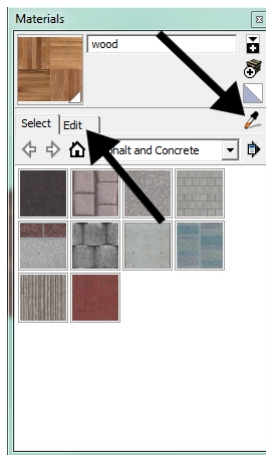
Suppose you have a material where the texture is too 'stretched-out'.



We can fix this by changing the texture scaling.

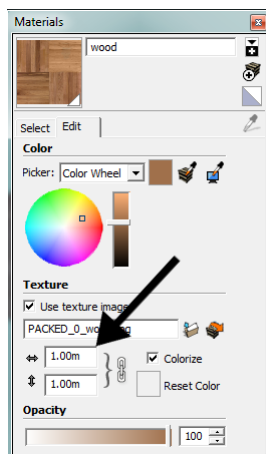
In SketchUp, show the material dialog with Window -> Materials.

Select the material you wish to scale with the picker tool. In our case we will select the wood material on the ground object.

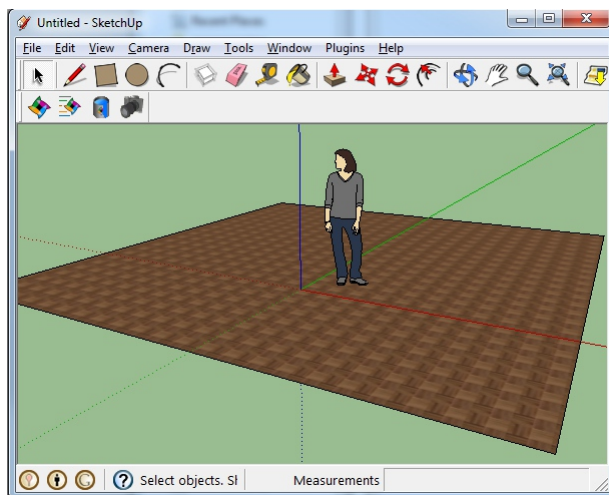


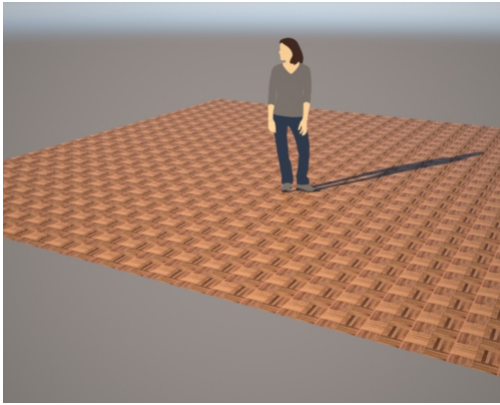
Click on the 'Edit' tab.

In the 'Texture' section, change the horizontal width to a smaller value, such as 0.3 m.



The texture should now repeat more rapidly over the surface, in SketchUp and in Indigo as well, when the scene is exported again.





### Procedural Materials

The same technique can be used for procedural materials, which may not use a texture map. In this case, you can use a 'dummy' texture, by clicking 'Use texture image' in the Materials edit tab, and then selecting a dummy texture. The dummy texture will be displayed in the SketchUp viewport, but will not affect the Indigo render.

## SkIndigo on Windows Tutorial

This tutorial will cover getting Indigo running with Google SketchUp on your computer running Microsoft Windows. We will use the SkIndigo exporter to export scenes from SketchUp to Indigo.

You can use either the free or the commercially licensed Indigo version to follow this tutorial; the free version will add a watermark to the final renders and limit the resolution to 0.7 megapixels.

### Step 1: Install SketchUp

If you already have SketchUp installed, you can skip this step.

Download and install SketchUp from here: <http://www.sketchup.com/intl/en/download/>

### Step 2: Download Indigo Renderer

The latest version of Indigo Renderer can be downloaded from this page:

<http://www.indigorenderer.com/download-indigo-renderer>

If you have a 32-bit operating system, or you are not sure, download Indigo Renderer for Windows 32-bit.

If you have a 64-bit operating system, download Indigo Renderer for Windows 64-bit.

### Step 3: Install Indigo Renderer

Once you have downloaded the Indigo installer program in Step 2, run the installer program.

If the installer asks you 'Do you want to allow the following program to make changes to this computer,' select 'Yes.' Please carefully read the licence agreement, then click 'I Agree.'

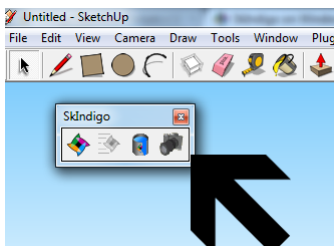
On the 'Choose Components' page, leave all components selected, and press 'Next >'

On the 'Choose Install Location' page, leave the Destination Folder as it is, and press 'Install.'

Press 'Finish'. Indigo will open after installation; close it for now, since we'll be using it via SkIndigo.

### Step 4: Open Sketchup

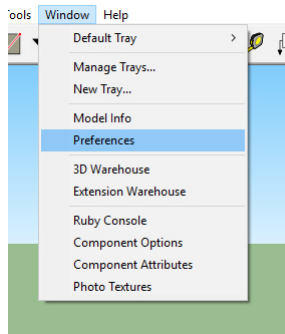
You should now see the new SkIndigo tool bar:



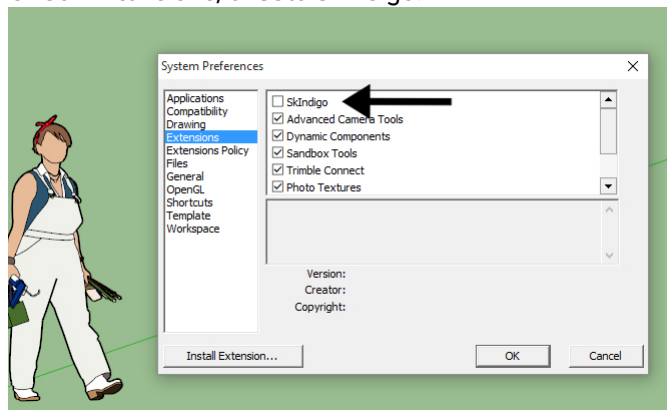
## Step 5: Enable extension in SketchUp 2016

In SketchUp 2016 you may need to enable the extension in the preferences.

Select Window > Preferences from the main menu.

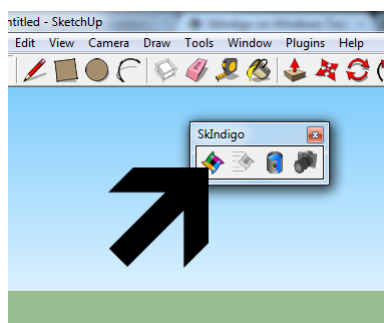


Under Extensions, enable SkIndigo.

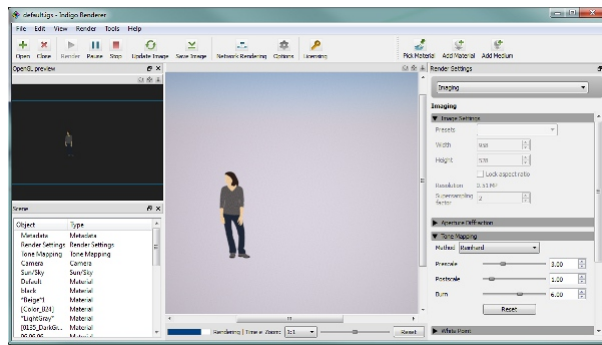


## Step 6: Render with Indigo

Press the 'Render with Indigo' button in the SkIndigo tool bar:



If everything has been installed successfully, Indigo should launch, and start rendering the default scene immediately:

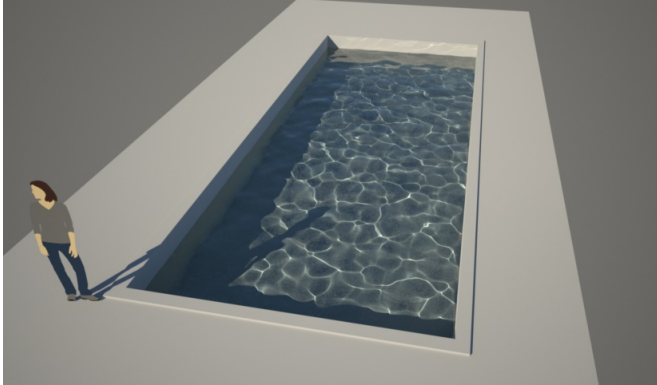


## Tutorials

# Modelling a pool in SketchUp

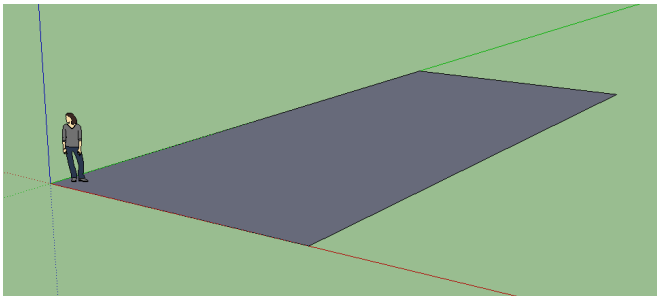
## Introduction

In this tutorial I will show how to model a simple swimming pool in SketchUp and Indigo, with nice caustics at the bottom of the pool. This is what we will be making:

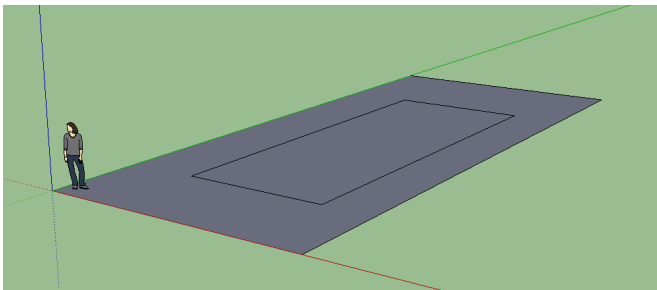


## Model the pool ground and walls

Create a quad like so:

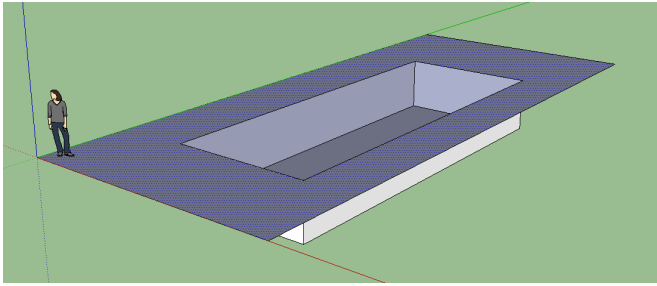


Create a rectangle in the middle of the quad:



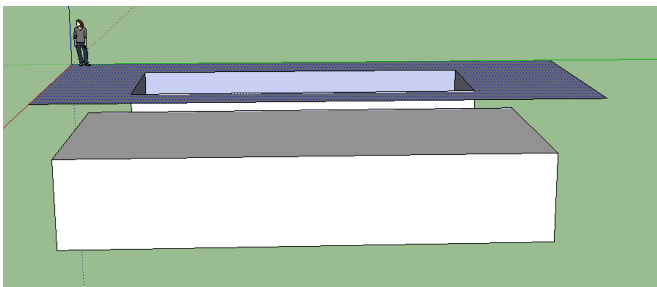
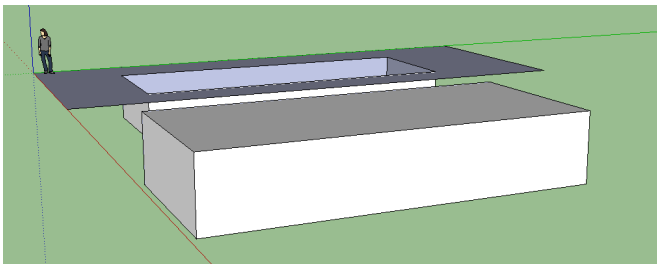
Using the Push/Pull tool, push the middle down to create the pool recess:





## Create the water volume

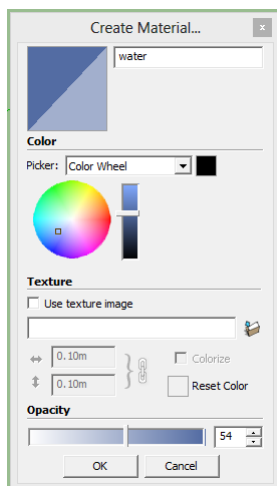
Create a cuboidal volume for the pool water. It needs to be somewhat larger than the pool recess along all axes.



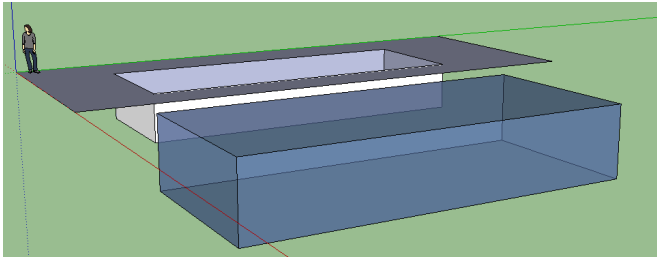
## Create the water material

Create a new SketchUp material called 'water' or similar.

Set the opacity to something around 50% to make sure the water material is transparent.

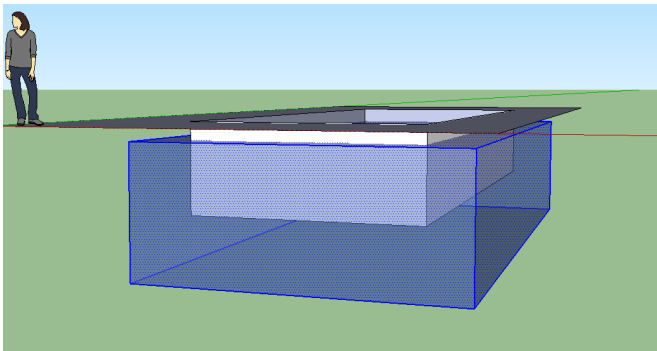


Apply the material to all faces of the water volume.



### Move the water volume into place

Move the water volume into place with the move tool. The sides of the water volume should extend past the pool recess like so:

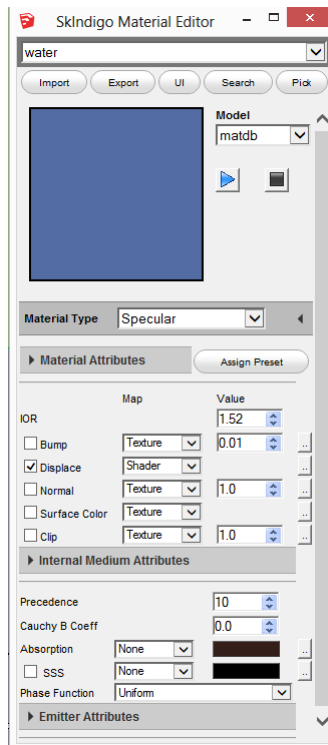


### Finishing the water material

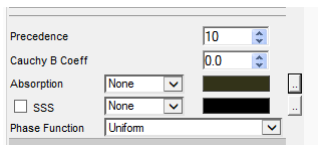
We need to set a couple of settings in the water material - some absorption to give the water a blue/green tint, and some displacement, to create the water ripples.

Open the SkIndigo material editor, and make sure the 'water' material is selected.

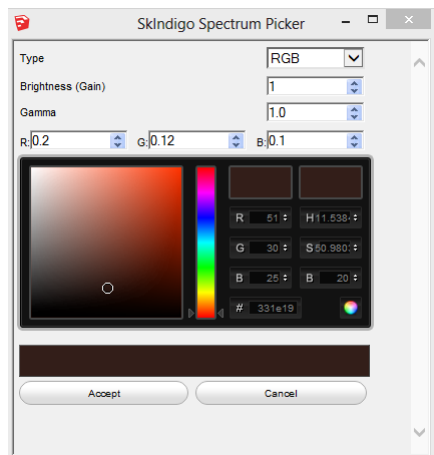
Change the material type to 'specular':



Change the Absorption type to 'none'.



Click the dotted button to the right of 'Absorption'.  
Set the RGB absorption to R: 0.2, G: 0.12, B: 0.1:



Now we need to create the displacement shader.

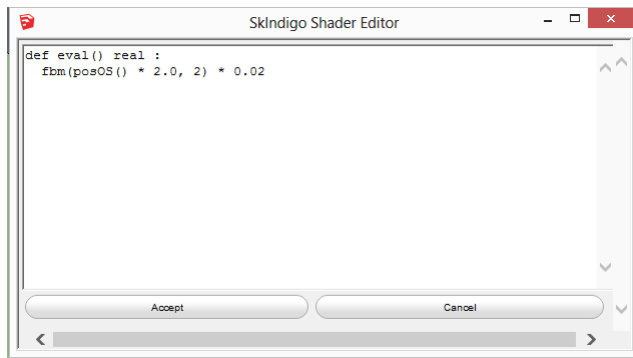
Check the 'Displace' checkbox in the SkIndigo material editor, and set the type to 'Shader'.

Now click on the dotted button to the right of 'Displace'.

Paste the following shader code in to the shader editor:

```
def eval() real :
fbm(posOS() * 2.0, 2) * 0.02
```

Like so:



What this shader does is displace the water surface by up to 2 cm (0.02 m), based on some pseudo-random noise (fbm) based on the object-space position (posOS).

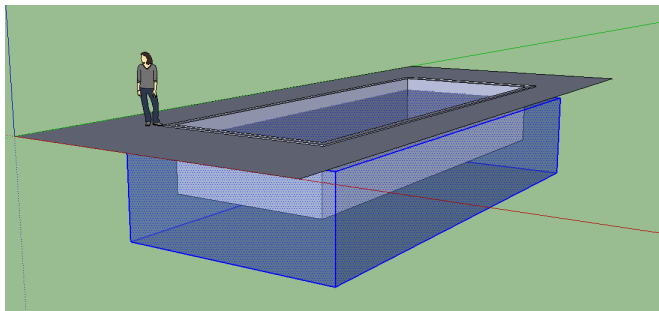
If you want to make the ripples higher, you can increase the 0.02 number.

If you want to make the ripples more closely spaced, you can increase the 2.0 number.

## Setting subdivision on the water volume

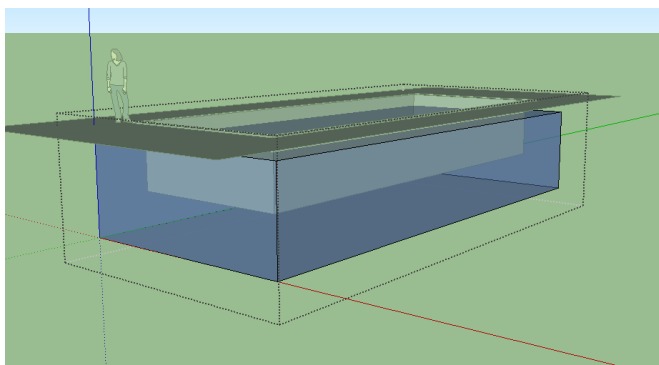
We want to make sure that *only* the water volume mesh gets subdivided.

To do this, first make sure the entire water volume object is fully selected by triple-clicking on it:



Now right click on it, and choose 'Make Group'.

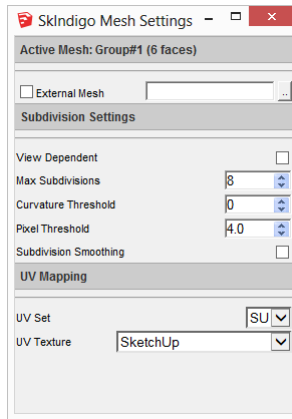
Now double-click on the group to select it. This should grey everything else out:



Now right click on the pool volume and choose 'Edit Active Mesh' from near the bottom of the menu. It should say something like 'Group#1 (6 faces)' at the top of the SkIndigo Mesh Settings dialog.

Set 'Max Subdivisions' to 8. This will turn each original quad face into  $4^8 = 65536$  subdivided faces.

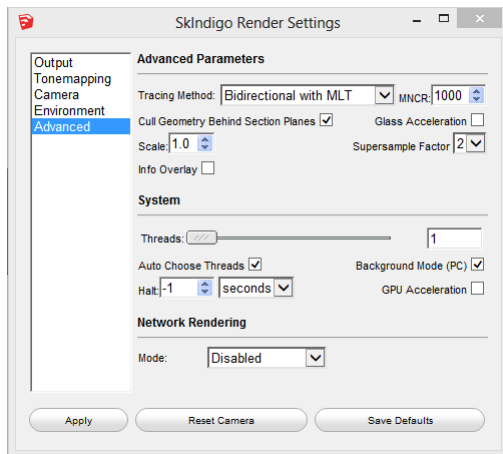
Also un-check the 'View Dependent' checkbox, and set 'Curvature Threshold' to zero:



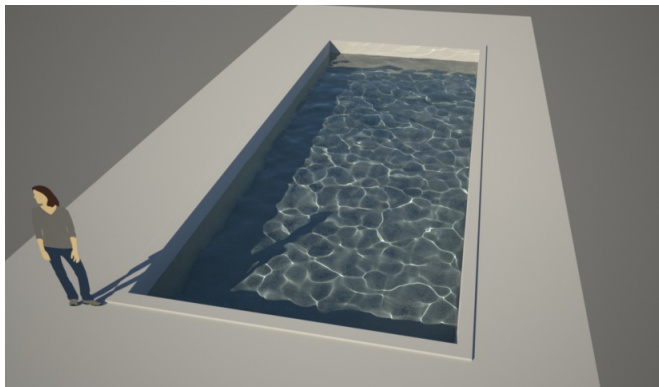
## Make the final render

First off, we have to change the render mode to Bidirectional with MLT:

This is very important as it's the most efficient mode for rendering these kind of caustics.



Position the camera above the pool, then press the 'Render in Indigo' button. After a while, you should get an image like this:



You may have to be a little patient waiting for the render - even with bidirectional MLT, it can still take a while.

You can [download the SketchUp scene \(.skp\) file here](#).

## Setting up interior lights mixed with sun+sky lighting

We will start with a kitchen scene, downloaded from the SketchUp 3d Warehouse, with some walls added:



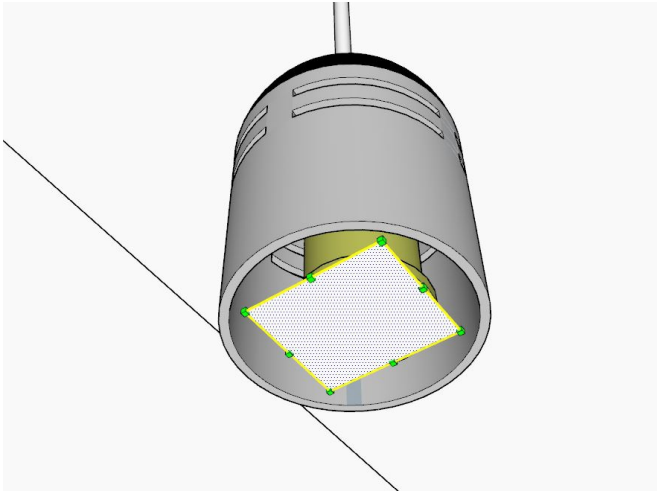
By default this will render in Indigo with Sun+sky illumination, resulting in a render like this:



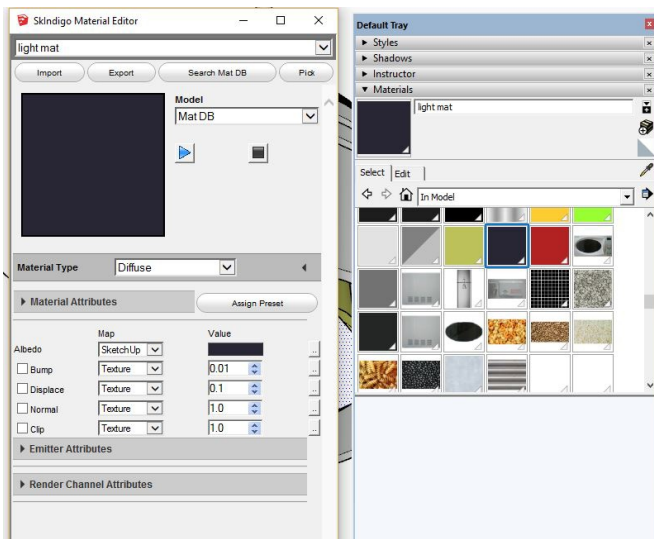
We now want to set up the lights hanging above the bench so they emit enough light to be visible.

First off, we want to add a quad in each light fixture which will emit the light.

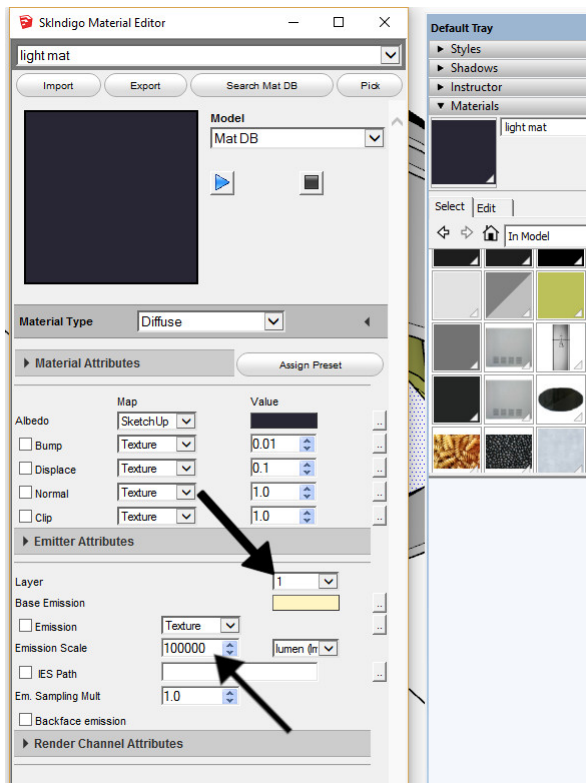
A single quad results in the most efficient rendering. The quad should not be too small (otherwise it will make specular reflections noisier). It is also important that the front side of the quad is pointing down - as in Indigo by default, light is emitted from the front side of a surface only.



The next step is to create a new SketchUp material. I have called mine 'light mat'. Since I have the SkIndigo material editor open (you can open the SkIndigo material editor from the SkIndigo toolbar) it shows and allows me to edit the 'light mat'. You will also want to assign the your new material to the new quads you have created.

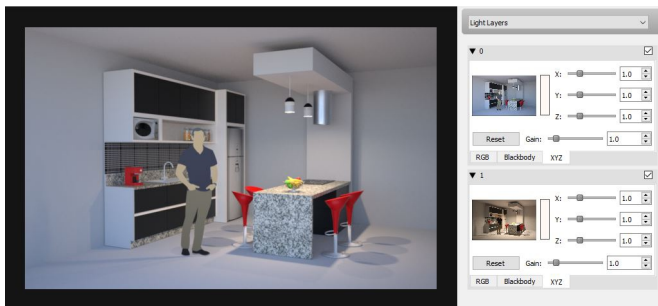


The next step is to edit your new material so it emits light. You can do this by expanding the Emitter Attributes section, then setting the Layer to something like 1, and setting the Emission Scale to something like 100000 lm. Note that the emission scale needs to be very high for the light to be easily visible in daylight.



And that's all there is to it!

Since we put the interior lights on light layer 1, we can use the light layer controls in Indigo to view the contribution of the interior lights separately from the sun+sky lighting (see the layer thumbnails on the right):



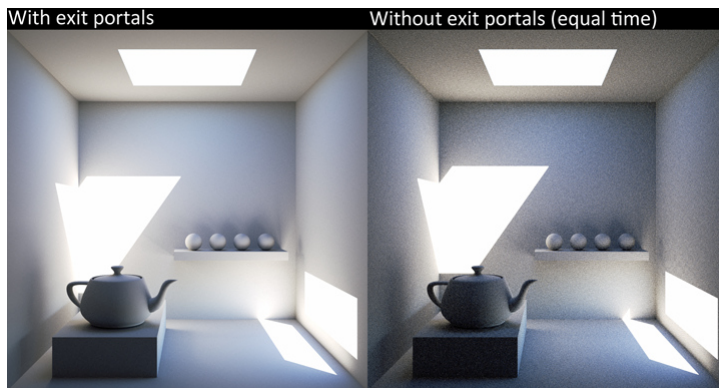


## Using exit portals in SketchUp

### Introduction

Exit portals (often abbreviated as EP) are a way for artists to assist the rendering engine in difficult lighting situations, by specifying a "portal" through which light will travel. These are very useful for interior renders, where there is a relatively small opening letting in a lot of light (such as a window) that is not easily accessed - in such scenes, there is often a dramatic increase in convergence speed (since light-carrying paths are much more easily constructed).

In the example below both images rendered for 5 minutes, the only difference being the presence or absence of exit portals (click to enlarge):



### Requirements and limitations

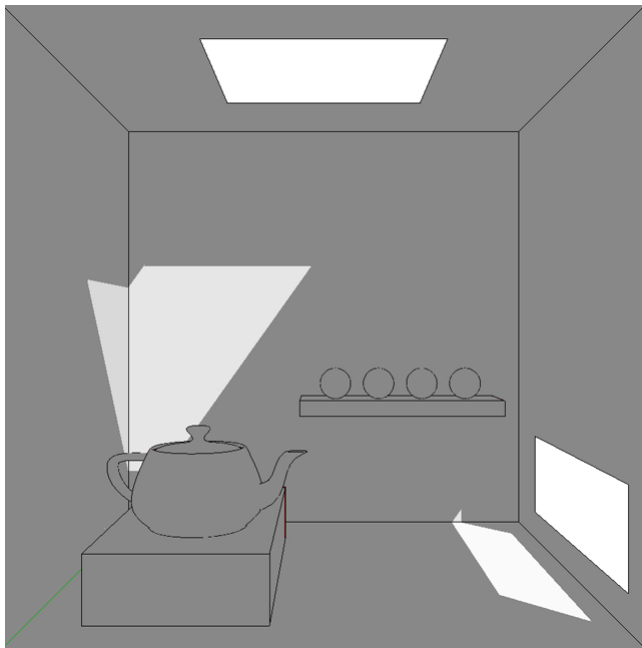
There are several requirements for exit portals to work effectively:

- All openings through which light can "escape" (directly to the background or environment) must be covered with exit portals.
- The exit portal normals must point inwards.
- Bi-directional path tracing mode must be used.

Finally, note that nothing gets rendered behind an exit portal: when a light ray strikes the portal it is assumed to immediately "exit" or escape to the background / environment, without considering any other objects in the scene.

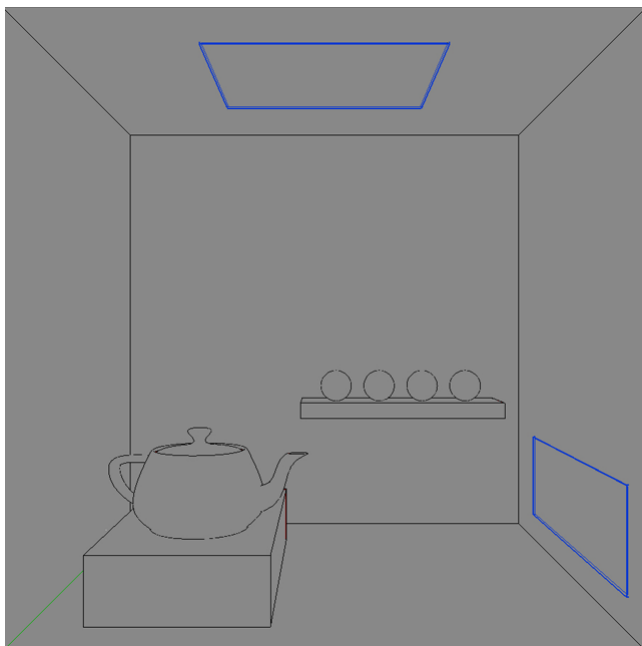
### Modelling with exit portals

We now show how to convert a simply modelled room with an opening into a scene using exit portals in SketchUp. The scene below shows an interior space, illuminated only by two openings which allow sunlight in:

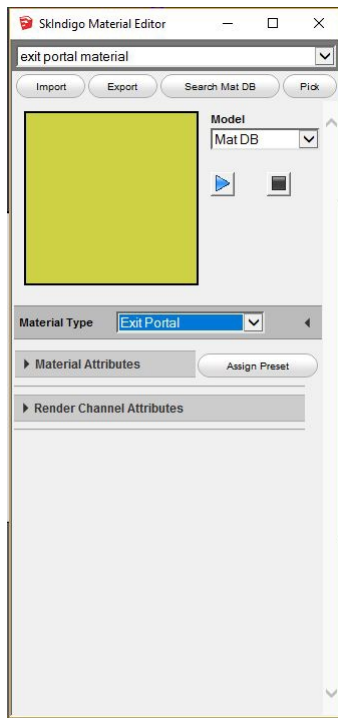


Next we create quads to cover the openings, corner to corner. There should be no gaps, and to really make sure this is the case it could be made slightly bigger than the opening and fitted inside to slightly intersect the surrounding geometry.

The next step is crucial to stop these quads being part of the surrounding geometry (and therefore sharing its material settings): select the quads to become exit portals, right click on them, and make them a group. You should see them marked as a blue group:



To turn them into exit portals, simply use the exit portal material type for the object.



With the exit portal created, we can now export to Indigo and enjoy the much faster convergence.

If Indigo reports an error such as "Error: Scene parsing error: Model (UID 42) has a different number of materials than geometry material references", it means that the exit portal quad was merged with the surrounding objects. This won't work because we need only the quad to be flagged as an exit portal.

Make sure that the front face of your exit portal object is pointing inwards into the scene!  
You may need to flip the group along an axis to achieve this.

Thanks to [Filippo Scarso](#) / [Pibuz](#) for the example scene and helpful suggestions.

## Using orthographic cameras with SketchUp

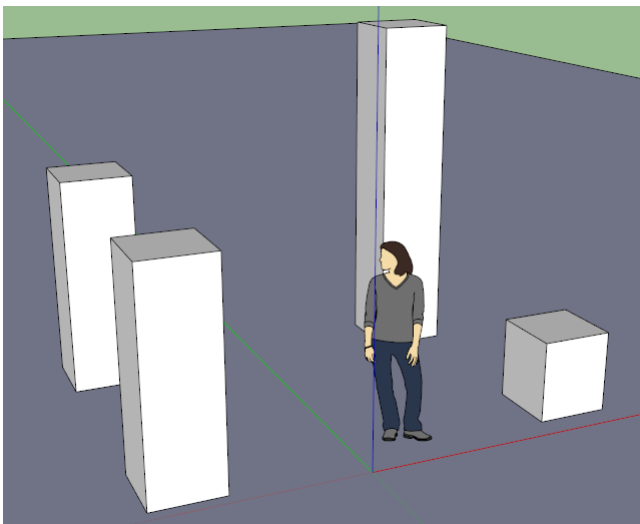
### Introduction

Orthographic cameras remove the perspective effects normally seen in a 3D rendered image. This is sometimes desirable for technical illustration or architectural visualisation purposes.

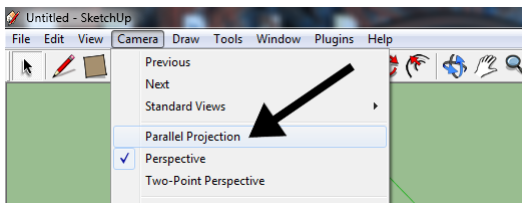
In this tutorial we'll cover the steps required to render with orthographic cameras in SketchUp, as well as cover some potential pitfalls with their use.

### Switching from perspective to parallel projection

We start with the normal (perspective projection) camera, which will give the normal depth cues such as distant objects being smaller, in a simple scene with some boxes:

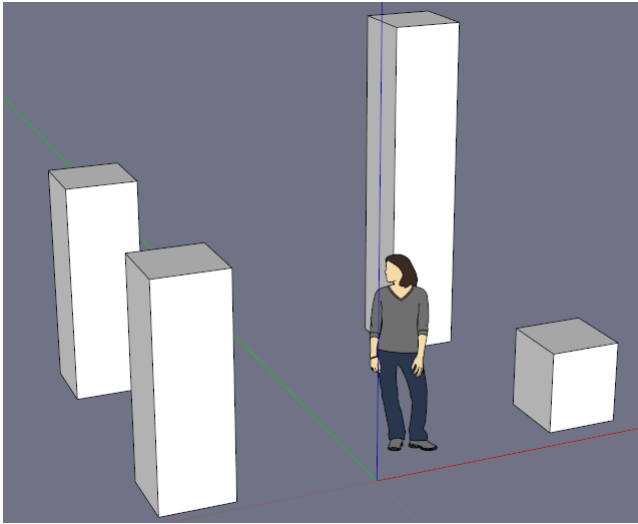


We now select the Parallel Projection option from the Camera menu, which will enable the orthographic camera mode:



With this applied, we can immediately see the viewport change to remove all perspective effects.

The zoom level may need adjusting to keep the same part of the scene in view; bear in mind that this is actually changing the camera's sensor size, so if you're rendering a building it will need a building-sized camera.



## Pitfalls

Orthographic cameras require Indigo 3.4.4 or newer; earlier versions will report an error when attempting to render scenes which have a non-perspective camera defined.

The orthographic camera must be at least as large as the objects in view, since they are directly projected (parallel rays) without any perspective effects which might otherwise allow large objects in the distance to be viewed with a much smaller sensor.

One potential pitfall with large camera sensors is that they may intersect other scene geometry (e.g. go through walls), especially different media (e.g. go through glass or water), causing problems.

The former will result in open spaces between surfaces, which may or may not receive light, while the latter can be more problematic; Indigo currently (as of 3.4.8) assumes that the camera is in a single medium, which is usually a reasonable assumption, however with the large sensors sometimes used in orthographic rendering, this assumption is more likely to break down.

## Using section planes in SketchUp

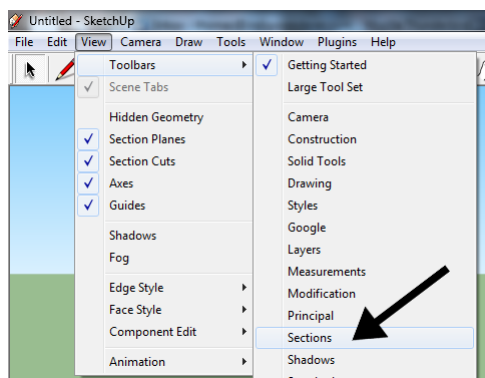
### Introduction

Section planes allow you to create "cut-away" renders of scenes without having to change the (potentially complex) underlying geometry, using oriented planes to slice away obstructing sections from view. This is related to [cut-away diagrams](#) classically used in technical illustration.

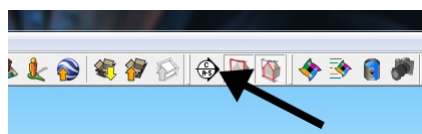
In this tutorial we'll cover using SketchUp's section planes with Indigo version 3.4 or newer.

### Adding and enabling sections

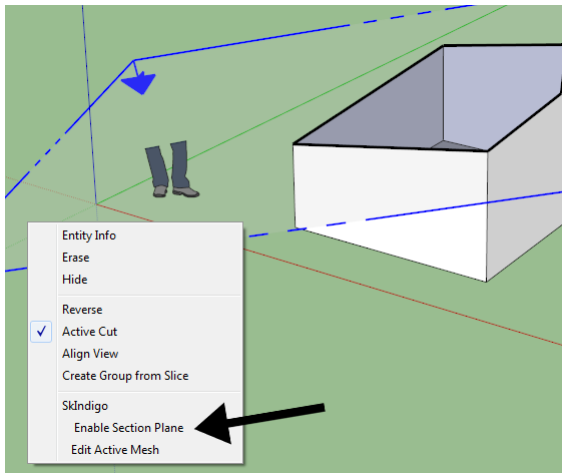
The first thing we'll need to do is enable the Sections toolbar if it is not already enabled; this can be done via the "View" menu, under "Toolbars" -> "Sections":



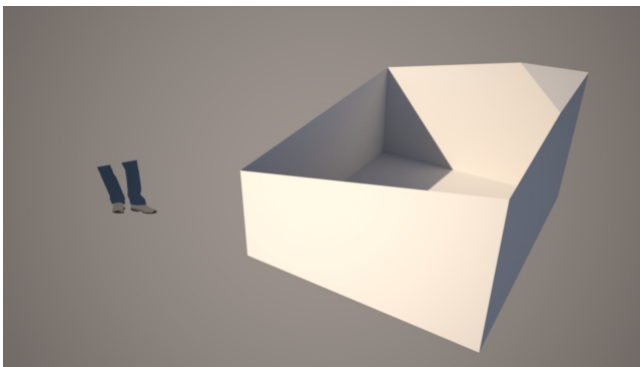
With these controls available, we can now add a section plane using the first tool on the toolbar (Section Plane):



Once the plane has been placed and oriented as desired, we must right-click on it and select "Enable Section Plane" from the SkIndigo sub-menu:



If we now render the scene with Indigo, we see that the section to the front of the section plane is rendered, and the rest has been clipped away:



Multiple section planes can be used together in this manner, to cut away whichever parts of the scene are obstructing an important interior view. There is no performance penalty for using one or many section planes, as they are processed at load-time; however, if a section plane intersects a complex mesh, it may require a considerable amount of extra memory to process during loading.

## Pitfalls

If the section plane intersects any media in the scene, there is a chance that the medium definition will become inconsistent.

For example, if a glass pane is sliced by a section plane such that one of the sides is removed, the geometry containing the medium will be open, which can cause rendering problems.

## Using the material database in SketchUp

This tutorial will cover how to download and use materials from the [online material database](http://www.indigorenderer.com/materials/) using the SkIndigo exporter for SketchUp.

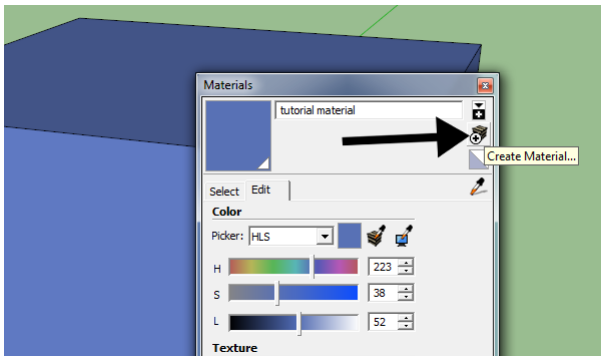
The Indigo Material Database can be found at <http://www.indigorenderer.com/materials/>

There are two ways to use materials from the online database in SketchUp/SkIndigo: loading directly into SketchUp, and externally linking to downloaded materials. We'll cover both options in sequence, starting with the simpler direct import method.

### Importing directly into SketchUp

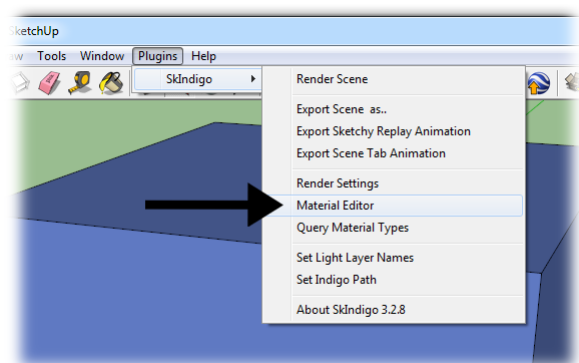
#### 1. Creating a new SketchUp material

Let's start by opening SketchUp and making a simple object to apply the material to (in this case a cube). We'll also want to create a new SketchUp material (via Window menu -> Materials -> Create Material) to hold our Indigo material, and apply it to the newly created cube.



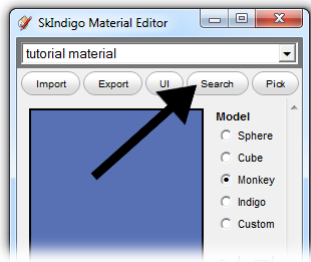
#### 2. Opening the SkIndigo online material browser

Having created a simple object and applied a new material to it, from the "Plugins" menu, under the "SkIndigo" sub-menu, select "Material Editor":



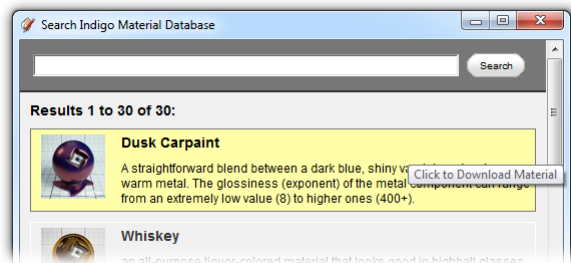


This opens the SkIndigo material editor window. At the top of the material editor window is a "Search" button, click this to open the material database window in SkIndigo:



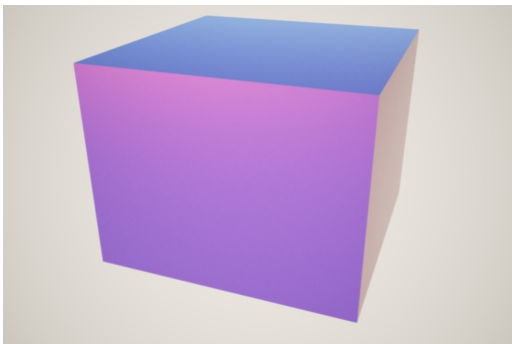
### 3. Downloading and applying a material

The material database browser window will open, allowing you to search for materials by keywords or name. By default the most recently materials submitted will appear at the top:



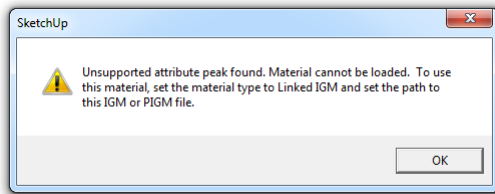
If we choose a material and double click it, SkIndigo will ask if you'd like to load it into the currently selected material; click "Yes", otherwise it will want to save the downloaded material to disk.

The downloaded material should now be loaded into the SketchUp material and applied to your object, ready for rendering with Indigo:



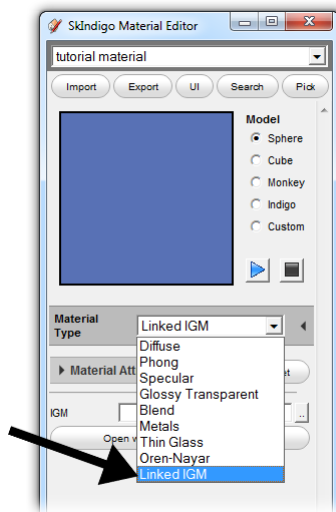
### Alternative method: Linked IGM

It is however possible, that the material cannot be represented properly within SketchUp, since it is not a physically-based renderer. In such situations, SketchUp will present a warning dialog suggesting it be used as a "linked" material:



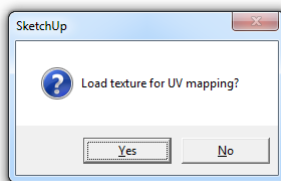
## 1. Linking a downloaded material

To do this, we select the material in the SkIndigo material editor, and set its Material Type to "Linked IGM":



Click the ".." button next to the "IGM" field, and select a downloaded IGM or PIGM file. This will link the downloaded material to the SketchUp material so that when Indigo renders the scene, it will use the linked material.

Since the material cannot be represented correctly in SketchUp, SkIndigo will prompt you for a texture to use for the material in the SketchUp viewport. This is so you can easily identify the linked material, however it is optional:



## Indigo for Cinema 4D

This manual is for Indigo for Cinema 4D 3.2.10 or later.



The Cosmonaut by aleksandra

# Installation

This sections describes how to Install Indigo for Cinema 4D on your Computer.

## Installing Indigo for C4D on Windows

This tutorial will cover getting Indigo running with Indigo for Cinema 4D on your computer.

This tutorial is for Windows users.

If you have not purchased an Indigo licence, you can still follow this tutorial. Indigo will run in trial mode, which will apply some watermarks to Indigo renders.

### Step 1: Check your version of Cinema 4D

Indigo for Cinema 4D is compatible with 64-bit versions of releases 12 to 23 of Cinema 4D.

### Step 2: Download Indigo

From the page <http://www.indigorenderer.com/download-indigo-renderer>, download the latest version of Indigo for Windows.

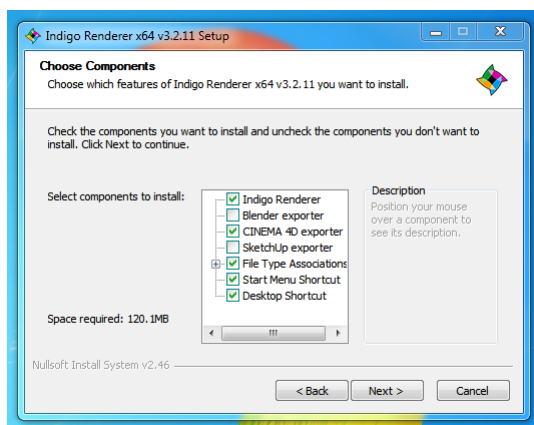
### Step 3: Install Indigo

Once you have downloaded the Indigo installer program in Step 2, run the installer program.

If the installer asks you "Do you want to allow the following program to make changes to this computer", select "Yes".

Then accept the licence agreement.

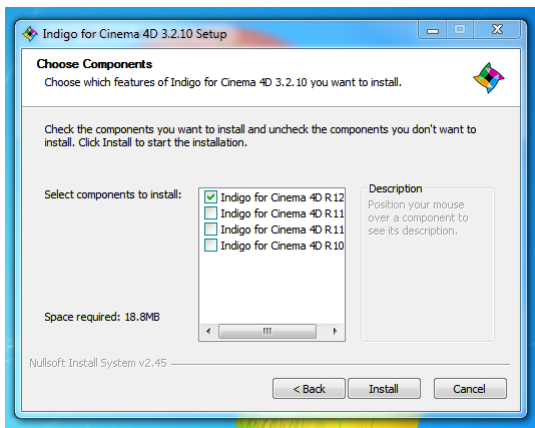
On the 'Choose components' page, select the C4D Exporter, and press "Next".



On the "Choose Install location" page, leave the Destination Folder as it is, and press "Install".

### Step 4: Install Indigo for Cinema 4D

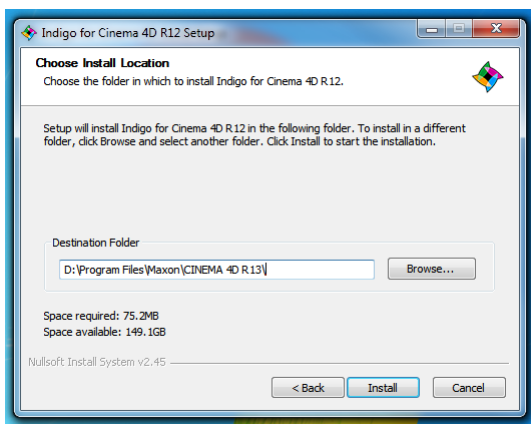
An installer will launch where you can select the C4D versions you want to install Indigo for C4D for on the "Choose components" page.



After selecting, press "Next".

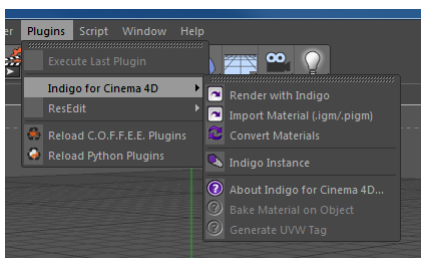
For every selected version, another installer will launch.

The installers will try to automatically detect your C4D installation directory. If the directory could not be automatically detected, set it to the main C4D directory.



### Step 5: Locate Indigo Plugin menu

Once installed you can find Indigo for Cinema 4D in the "Plugins" menu inside Cinema 4D:



## Installing Indigo for C4D on Mac OS X

This tutorial will cover getting Indigo running with Indigo for Cinema 4D on your computer.

This tutorial is for Mac OS X users.

If you have not purchased an Indigo licence, you can still follow this tutorial. Indigo will run in trial mode, which will apply some watermarks to Indigo renders.

### 1. Check your version of Cinema 4D

The Mac version of Indigo for Cinema 4D is compatible with C4D releases 12 to 19. There are both 32 and 64-bit versions of Indigo for Cinema 4D available.

### 2. Download and install Indigo

Download Indigo for your system and install it to the default location. Instructions for doing this are in the Indigo Manual. You can download Indigo from:

<http://www.indigorenderer.com/download/>

### 3. Download and install Indigo for Cinema 4D

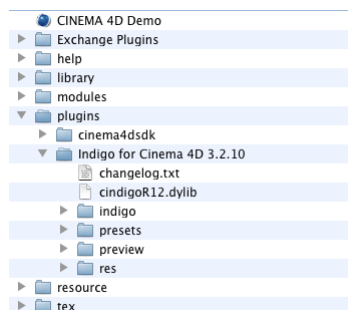
Download the version of Indigo for Cinema 4D for your system from:

<http://www.indigorenderer.com/cinema4d>

Extract the downloaded archive to you Cinema 4D's "plugin" folder, for example:

/Applications/MAXON/Cinema 4D/plugins

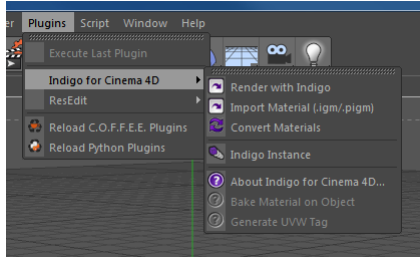
After extracting, your folder structure should look like this (an 'Indigo for Cinema 4D' folder inside the plugins folder, that contains all the Indigo for Cinema 4D files and folders):



If you have any issues installing Indigo for Cinema 4D, please email us at [support@indigorender.com](mailto:support@indigorender.com)

### 4. Restart Cinema 4D

After installing Indigo for Cinema 4D should become available under the Plugins menu.



The Indigo for Cinema 4D plugins menu.

You are now ready to use Indigo for Cinema 4D.



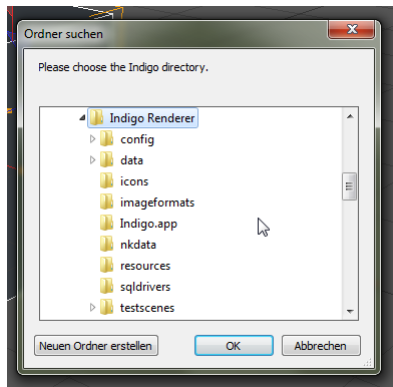
## Selecting Indigo for rendering externally

This section is about how to select the Indigo installation to use for rendering externally with Indigo Renderer or Indigo RT.

### Automatic detection of Indigo installation on Windows

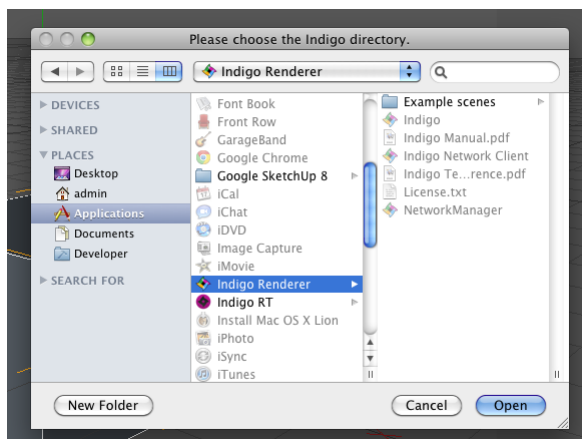
On Windows the Indigo installation directory should be automatically detected.

If the installation directory can not be detected Indigo for C4D will show a dialog on export asking to specify the location of indigo.exe.



### Selecting the Indigo installation on OS X

On OS X the installation directory can not be automatically detected. When starting the first export process, a dialog will show up asking to select the directory the Indigo.app is located in.

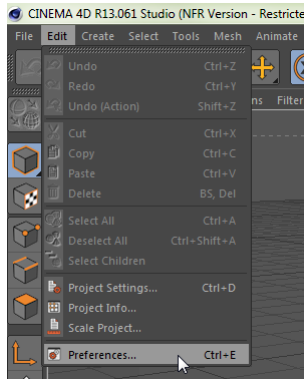


### Selecting Indigo installation in Presets in C4D R12 and newer

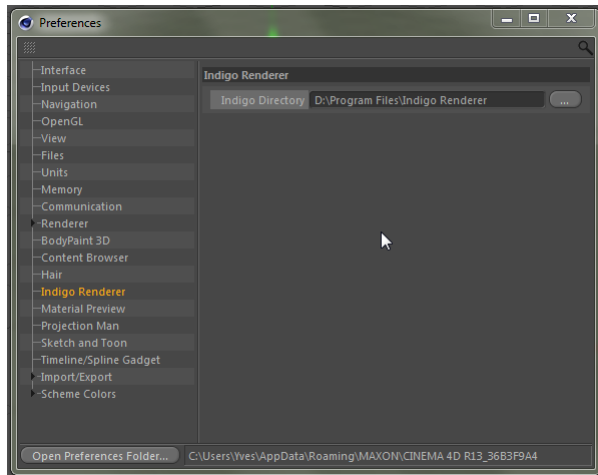
*This part applies to Indigo for C4D 3.4.8 and newer only.*

In C4D R12 and newer there is a Indigo Renderer page in the Presets window where the path to the Indigo installation can be changed.

The presets can be accessed from the main menu under "Edit" -> "Preferences" (or with the shortcut CTRL + E).



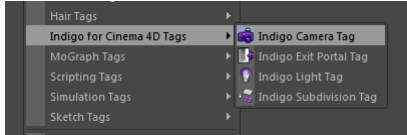
On windows this path will default to the automatically detected Indigo installation directory.



## Interface

## Indigo for Cinema 4D Tags

There are several special tags you can add to you Cinema 4D objects. These tags extend the functionality of certain Cinema 4D objects to include Indigo-specific functions. See the Indigo manual for more information.



### Indigo for Cinema 4D Tags

You can access this menu via: Object Manager menu > Tags > Indigo for Cinema 4D

## Indigo Camera Tag

Add this to a Cinema 4D camera object to set Indigo specific settings. Use in conjunction with the Cinema4D's camera settings such as focal length and aperture.

See [Camera](#)

## Indigo Exit Portal Tag

Adding this tag to an object makes this object export as an exit portal. The normals of the object should be pointing in the direction you want the light to enter the room.

Objects with this tag will only be exported as an exit portal and any materials on it will be ignored.

See [Exit Portals](#)

## Indigo Light Tag

Add this tag to the lights Omni lights, Area lights, or Spot lights to set Indigo specific light options. The options are the same as adding emissions to a material. Because there are no point lights in Indigo, on export these special Cinema4D entities are translated into a mesh and act exactly as if you made a light bulb mesh and applied an emitting material to it.

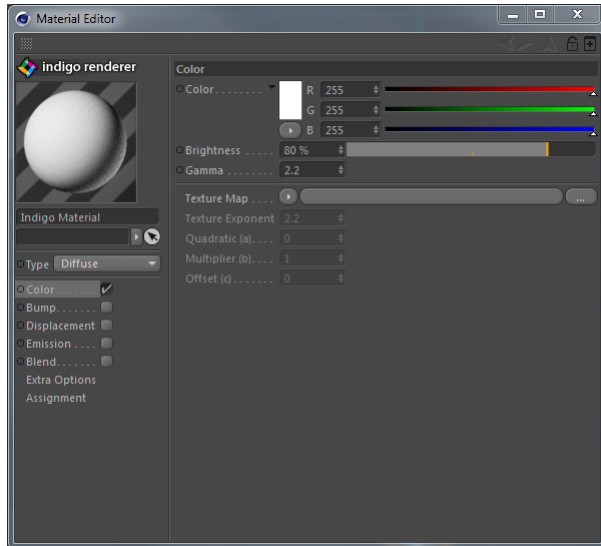
See [Emission](#), [Base Emission](#)

## Indigo Subdivision Tag

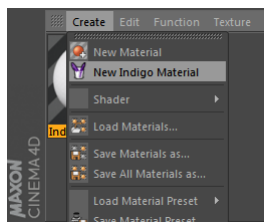
See [Subdivision](#)

## Material Settings

Here you can edit the attributes of an Indigo material. It can be found by selecting an Indigo material and looking in the Attribute Manager, or double-clicking on the Indigo material.



To create a new Indigo material, navigate Material Manager > File > New Indigo Material.



Create new Indigo material

## Material Types

See [Material Types](#)

## Material Channels and Attributes

See [Material Attributes](#)

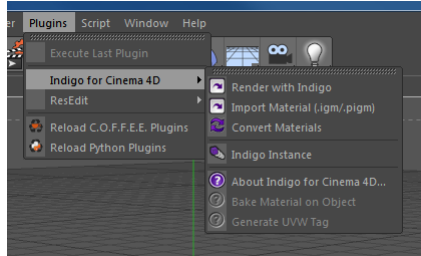
## Extra Options

Texture Preview Size	This changes the quality of the texture in the Cinema4D view-port, it does not affect the rendered scene at all.
----------------------	--

Shader Baking resolution:	The resolution to bake a shader at on export.
Export Material	Export the material to be used externally.
Assignment	Any objects that have this material applied are listed here.

## Plugins Menu

The Plugins Menu is mainly used to create Indigo objects, convert materials and start rendering.



Indigo for Cinema 4D Plugins Menu

### Render with Indigo

This exports the current scene to be rendered with Indigo. This works identically to the Cinema 4D command, except it exports using Indigo materials and launches Indigo to render the scene.

### Import Material .igm/.pigm

This loads an .igm (Indigo Material) or .pigm (Packed Indigo Material) file from your computer and adds it to your scene so it can be applied to an object.

### Convert Materials

This will load all of the Cinema 4D materials and convert them automatically to Indigo materials. Indigo for Cinema 4D will convert all C4D materials automatically on export to Indigo, however using this function will give you more control over the material settings.

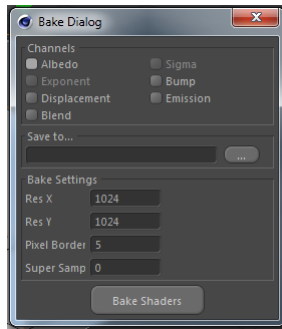
### Indigo Instance

Creates an instance that can be used to make multiple copies of an object in a memory-efficient and CPU-efficient way. Indigo instances are faster to render than Cinema 4D instances (which aren't real instances but are copies).

### Bake Material on Object

Indigo for Cinema 4D bakes all C4D shaders to textures on export, but this feature allows you to bake unique 3D shaders to an object's UV map. It works similar to C4D's "bake on object" command. A decent UV map is required. To use this function follow these steps:

Select the texture tag of a polygon object that has an Indigo material applied. Click Plugins > Indigo for Cinema 4D > Bake Material on Object.



### Bake Material on Object dialog box

Select the channels you want to bake and specify an output path. Pixel Border adds a blank border to the output texture file. Super Sample is an anti-aliasing technique.

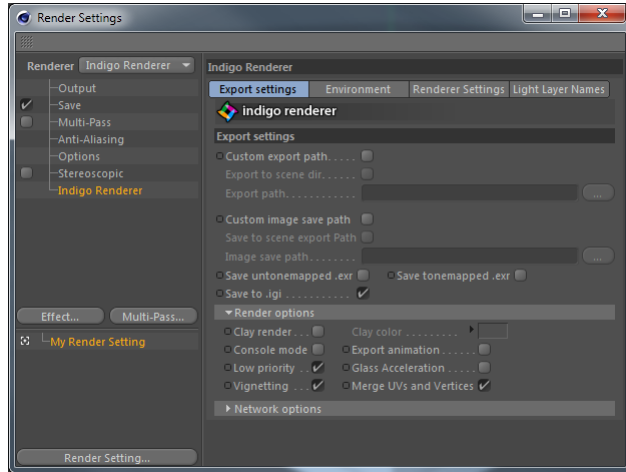
### Generate UVW Tag

C4D does not fully support UVW map generation for custom materials. This Indigo for Cinema 4D function allows you to bypass this limitation. Just select a texture tag with an Indigo material applied and click Plugins > Indigo for Cinema 4D > Generate UVW Tag.



## Render Settings

The Indigo Render Settings let you control various aspects of how Indigo will render the scene.



Indigo for Cinema 4D Render settings

Most of the settings used for your renders will be copied directly from the existing Cinema 4D render settings, e.g. width and height come from the Output section of the Render Settings dialogue. However, there are some Indigo specific render settings that can be accessed via:

R10 and R10.5: Render menu > Render settings > Effect... > Indigo Render.

R11 - R12: Render menu > Render settings > General > Set the Render Engine to 'Indigo Renderer'.

R13: Select 'Indigo Renderer' from the 'Renderer' drop-down in the top left of the Render Settings window.

Note that you need to press the "Effect..." button on the Render Settings window to access the advanced Indigo Settings. You do not need any other render pass, Indigo does it all.

## Export Settings

See [Indigo Render Settings](#)

## Environment

See [Environment Settings](#)

## Render Settings

See [Indigo Render Settings](#), [Render Mode](#)

## Light Layer Names

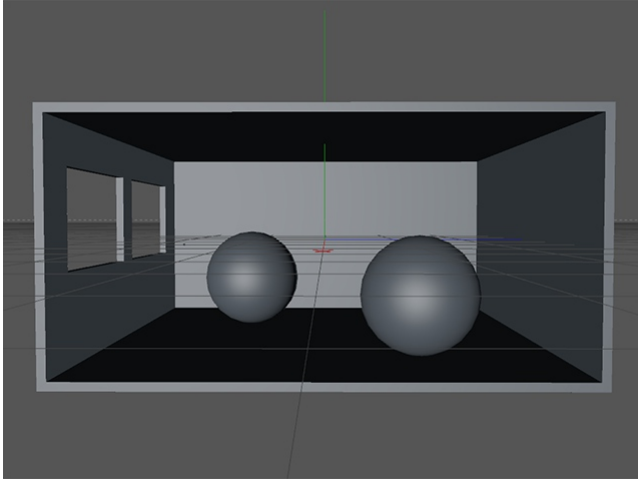
See [Light Layers](#)

## Tutorials

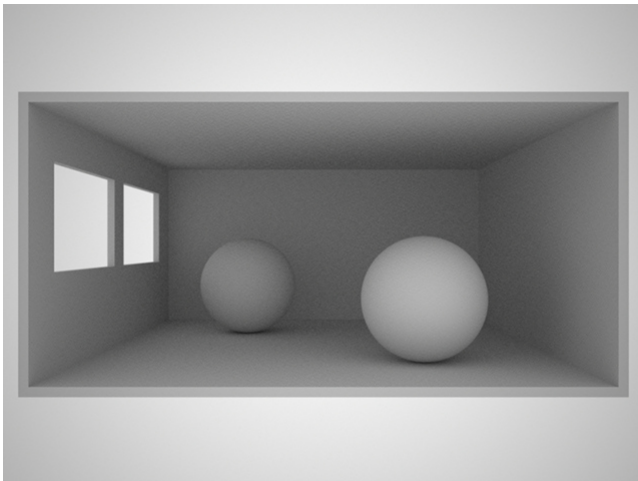
## Getting started tutorial

This is a quick tutorial to get you familiar with Indigo for Cinema 4D. A basic understanding of Cinema4D is required. The tutorial file is included with the download.

1. After installing Indigo and Indigo for Cinema 4D, open up Cinema 4D and construct a simple scene. This scene has two windows for light to shine in, and two spheres to test materials on.

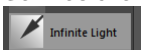


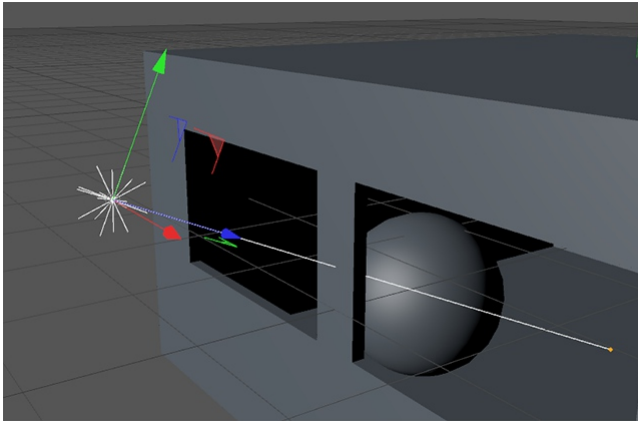
2. To render the scene with Indigo, select Plugins > Indigo for Cinema 4D > Render with Indigo from the menu, which will start the standalone Indigo application. Notice it is very plain and noisy, there is no need to render it for more than a few seconds as you can see right away that it needs work.



Quick render with Indigo

3. First, we will light it with the sun and sky model, which is the easiest and quickest way to add light. Add a C4D Infinite Light and point it in the direction that you want the sun to face. The Z axis (blue) defines the direction of the sun light.

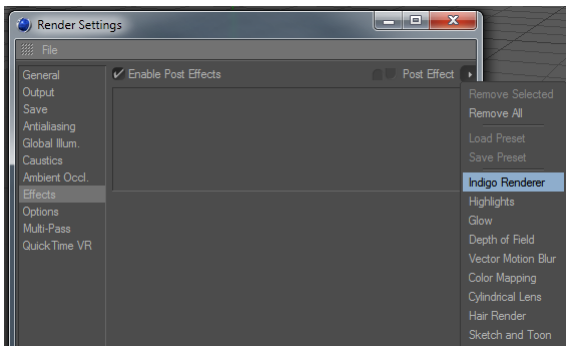




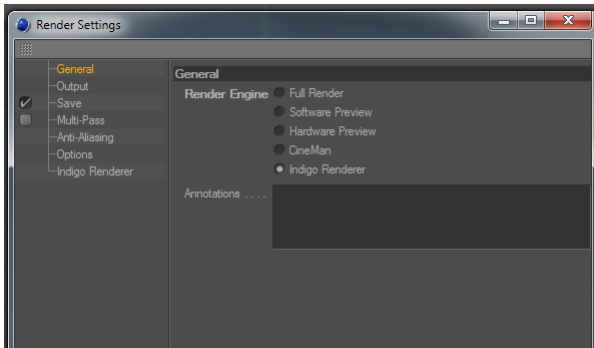
The sun will come in through the window

- Open up the Render Settings, and set Indigo as the renderer. If you are using C4D R10 or R10.5, you will need to add Indigo as a Post Effect.

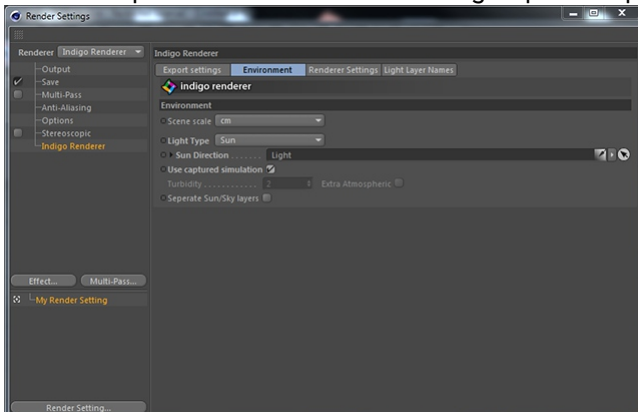
R10:



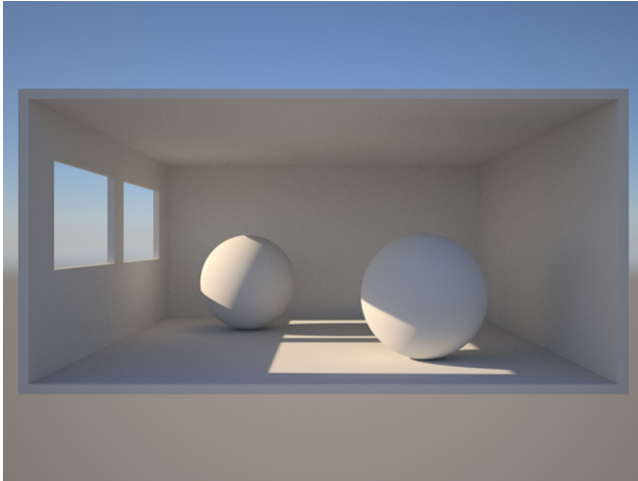
R11 - R12:



- Go to the Environment tab and click-and-drag your Infinite Light tag into the Sun Direction. Turn on the 'Use captured simulation' to use Indigo's pre-computed sky data.



6. Render again to see the effect of the sun light on the scene.

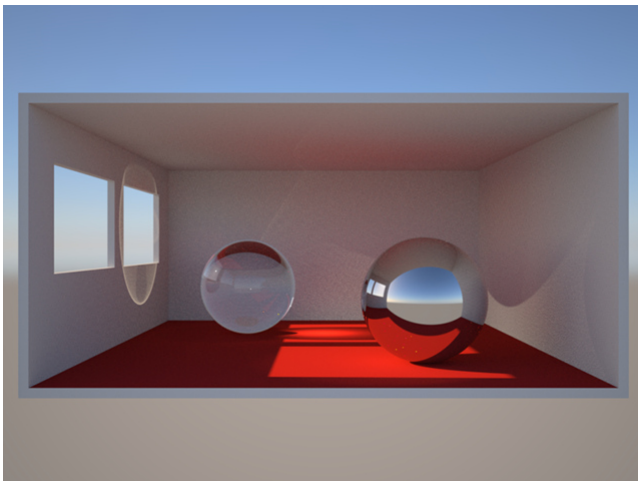


7. Now to add some materials to the scene. To Add a new material click: Material Manager > File > New Indigo Material. Change the material type to 'Phong' and increase the 'Exponent' to 100,000. Under the 'Phong' settings, turn on 'Use Specular Reflectivity'. Drag the new material on to the two spheres.

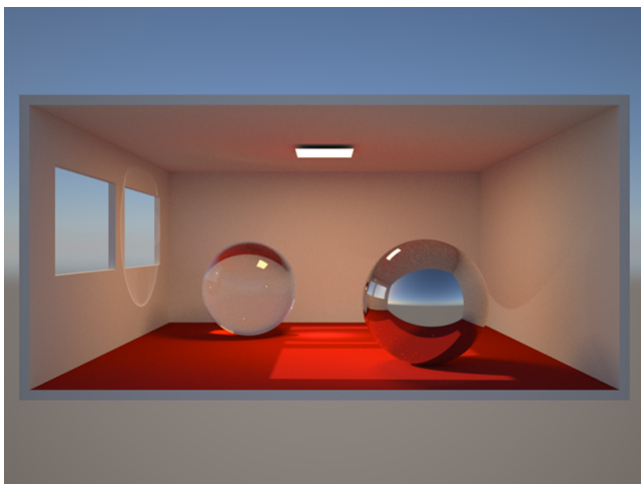
For the floor, we will make a material of type 'Diffuse' with the 'Color' (albedo), set to red.



8. Another render shows what the materials look like.



9. Last, we will add a light source for some interior lighting. Create an area light and apply a 'Indigo Light Tag' (right click on the light, Indigo for Cinema 4D > Indigo Light Tag). In the Light Tag, set the 'Light Spectrum Type' to 'Blackbody', change the 'Temperature' to 3500K and lower the 'Gain' to 0.005. If the light is too bright, the Reinhard tone mapping will compensate for it and make the sun & sky darker, which is why the gain has been set to 0.005.



## Tutorial Files



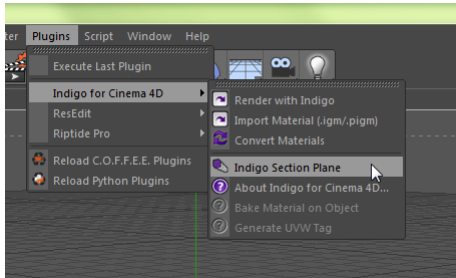
[Download the final scene of this tutorial.](#)

## Section planes tutorial

*Please note that section planes are not supported in Indigo RT.*

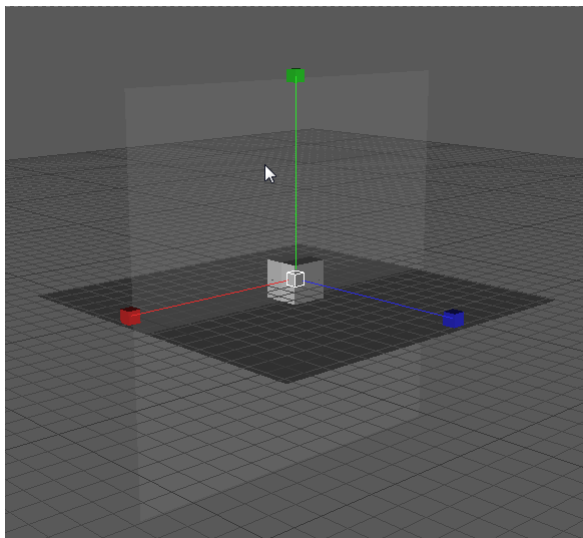
This is a short tutorial explaining the use of section planes with Indigo for C4D.

1. Add a section plane object to your scene. It can be found under "Plugins" -> "Indigo for Cinema 4D" -> "Indigo Section Plane".



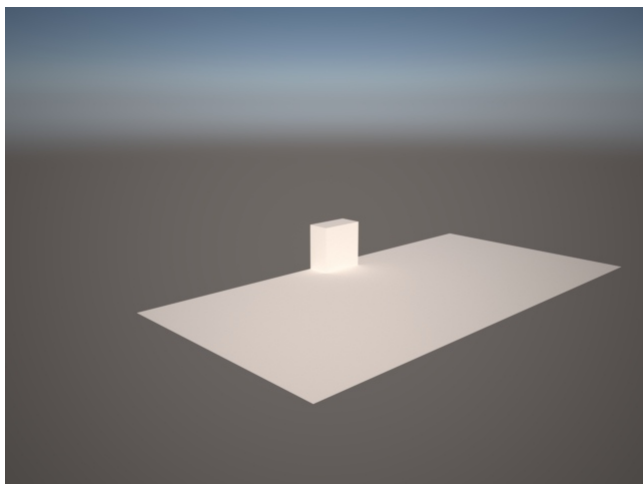
2. Move and rotate the section plane object in a way you would like it to cut your scene. The section plane will currently cut the whole scene. Everything in the opposite direction of the Z axis (blue) will be cut.

You can have as many section plane objects in your scene as you like.



3. Render the scene. If everything worked you should see a result like this:





## Tutorial Files



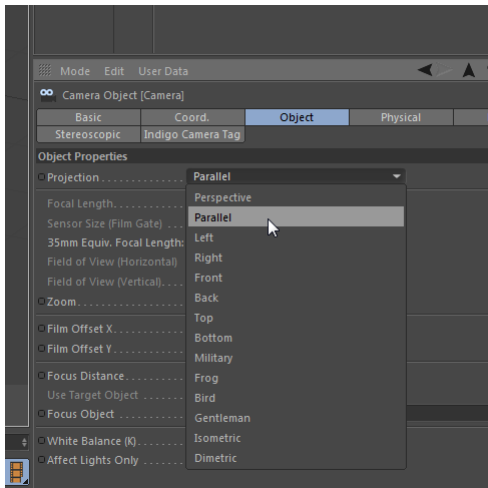
[Download the final scene of this tutorial.](#)

## Orthographic camera tutorial

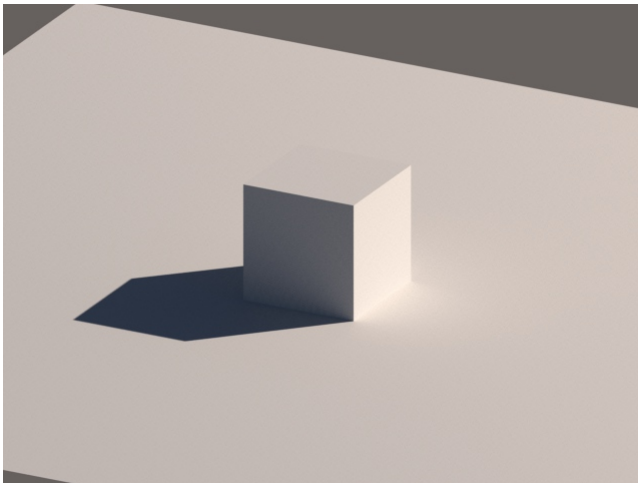
*Please note that orthographic cameras are not supported in Indigo RT.*

This is a short tutorial explaining how to use Indigo's orthographic camera feature in Indigo for C4D.

1. If not already present, add a C4D Camera object to your scene.
2. Select the camera and change the projection type in its properties to "parallel".



3. Set the camera as active camera and render the scene. If everything worked you should now be using Indigo's orthographic camera.



## Tutorial Files



[Download the final scene of this tutorial.](#)

## Example Scenes

Here you can find a few example scenes illustrating how to use Indigo for Cinema 4D.

The file contains scene files showing:

- Depth of field
- Exit portals
- Index of refraction
- Materials
- Sub-surface scattering
- Sun and Sky



[Download the example scenes. \(up to C4D R18\)](#)

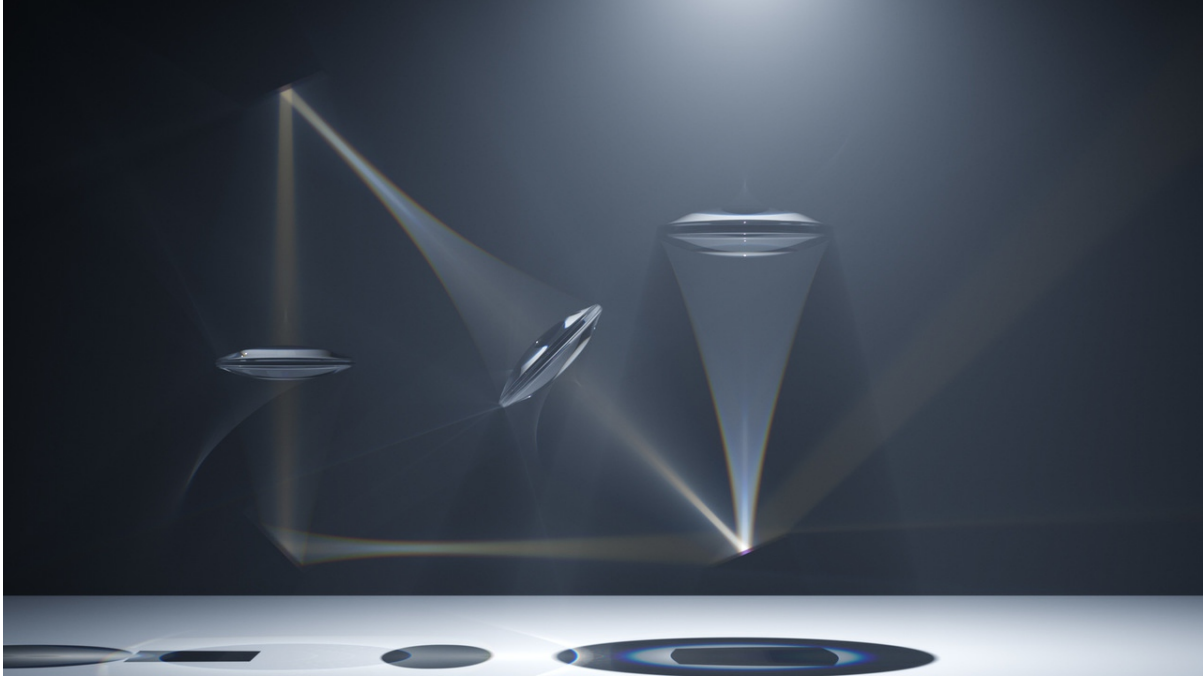


[Download the example scenes. \(R19 and up\)](#)

## Indigo for 3ds Max

Indigo for 3ds Max is the integrated rendering solution for Autodesk 3ds Max developed by Glare Technologies.

For general support inquiries please contact [support@indigorenderer.com](mailto:support@indigorenderer.com).



Lenses experiment by Silverwing

## Getting started

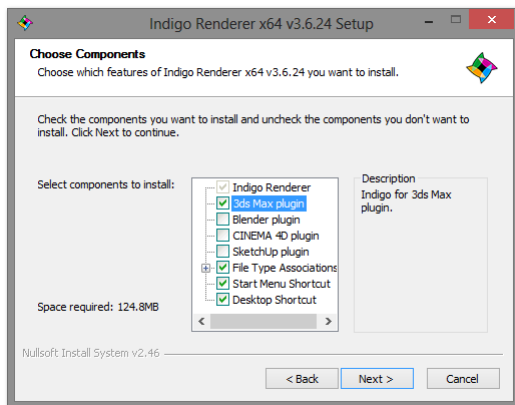
### 1. Ensure a supported version of 3ds Max is installed

A 64-bit version of 3ds Max is required.

### 2. Download and install Indigo

Download Indigo for your system [here](#); if you are unsure about which version to download, please see [this tutorial page](#).

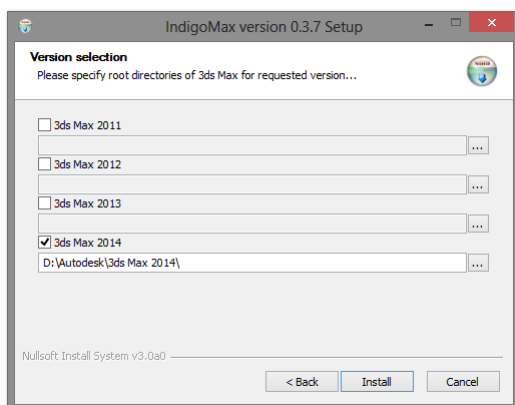
Upon running the installer, ensure that the "3ds Max plugin" component is selected for installation:



Continue with the normal install process for Indigo, after which the Indigo for 3ds Max installer will launch.

### 3. Install Indigo for 3ds Max

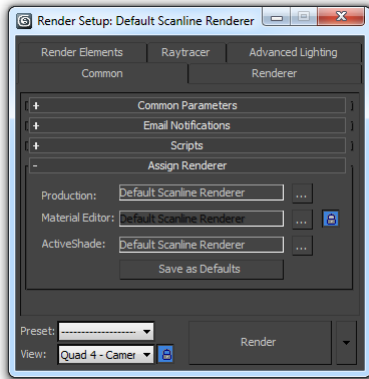
When the Indigo for 3ds Max installer launches, you will be prompted for the versions of 3ds Max you would like to use with Indigo:



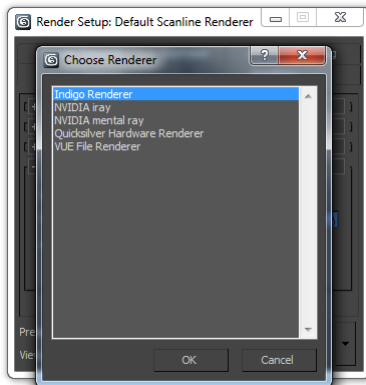
If you encounter any problems installing Indigo for 3ds Max, please email us at [support@indigorenderer.com](mailto:support@indigorenderer.com).

#### 4. Verify that Indigo for 3ds Max was correctly installed

After the Indigo for 3ds Max installer has completed, restart 3ds Max. Enter the Render Setup dialog by pressing F10, and click the "..." button next to the Production renderer entry in the Assign Renderer roll-out:



From the list of renderers, select "Indigo Renderer" :



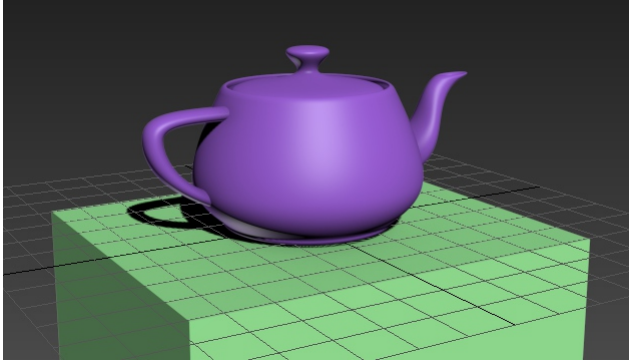
Indigo for 3ds Max is now installed and ready to use.

## Basic usage tutorial

This tutorial follows on from the [Getting Started guide](#), and covers the basic rendering and material editing workflow. Basic familiarity with 3ds Max is assumed.

### Rendering the basic scene geometry

We begin with an empty 3ds Max scene, and create a teapot on a pedestal as shown:



Before rendering, we must assign Indigo as the active renderer in the Render Setup dialog (keyboard shortcut: F10), as per the [Getting Started guide](#).

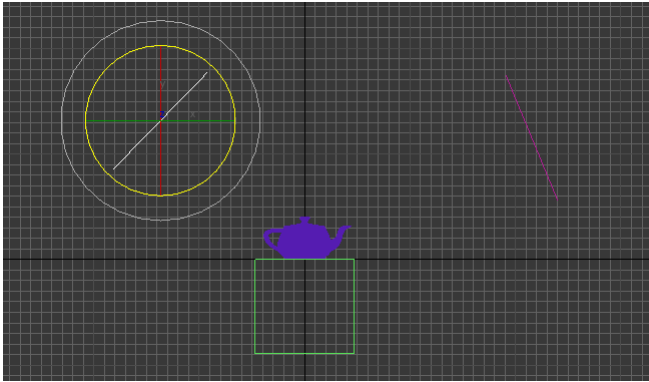
If there are no light sources present, which is the case in a newly created scene, IndigoMax will use a sun/sky background to illuminate the scene.

So if we now hit Render, we get an image like this:



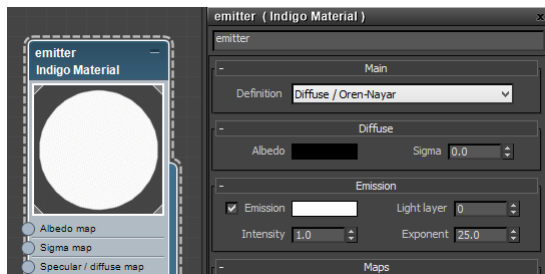
### Adding Indigo Material light sources

Next we will use our own light sources to illuminate the scene, instead of the default lighting. Lights in Indigo must have finite area, i.e. they cannot be point or directional lights; so we create the simplest possible surface, a quad patch, and point it at the teapot (two lights are shown here, illuminating from the sides):



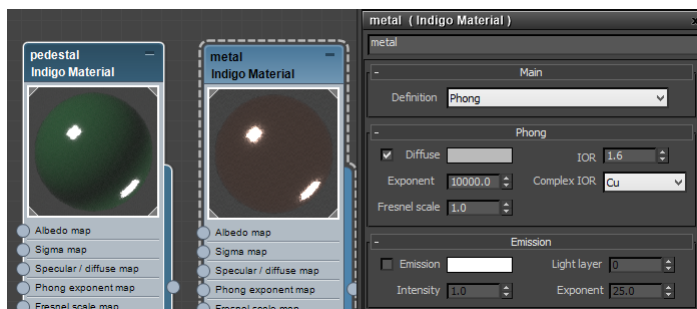
We must now create a material for this object, to specify an emission parameter. Open the Material Editor window and create a new Indigo Material, which we'll call "emitter".

In the emitter material settings, we'll set a black albedo colour (i.e. our light doesn't reflect any light) and a "white" emission colour:



## Adding Phong Indigo Materials

We'll also create two new Indigo Materials for the teapot and the pedestal. Change the teapot material definition to "Phong", then turn it into copper by choosing its chemical name "Cu" from the Complex IOR combo box. For the pedestal material we'll simply choose a dark green colour for the Phong albedo:



If we now apply the emitter material to the two light quads, and the teapot and pedestal materials to their respective models, we get the following rendered result:



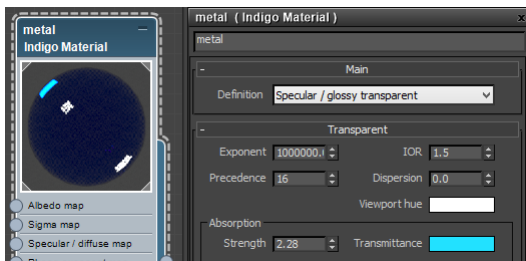


## Using specular materials

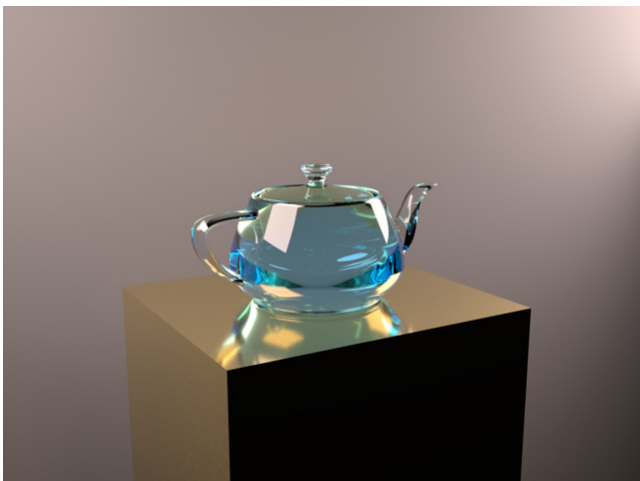
Another important material type is Specular, which simulates perfect reflection and refraction as governed by the physical Fresnel equations to produce realistic glass, water, acrylics etc.

The use of this material type is somewhat more complex than others, since objects with Specular materials must be closed (so as to produce a containing medium). For more information on this please see the manual page on [physically-correct glass modelling](#).

We'll turn the copper teapot material into a glass material by changing its material definition to "Specular / glossy transparent". To give the glass some colour, we set its transmittance colour to a cyan, and increase the Strength a bit to make it darker / more dense:



To better illustrate the effect of the Specular material we put a white wall behind the teapot and pedestal, and change the pedestal to be made of gold:



### Final notes

There is an increased amount of noise in the image, due to the complex light paths which can result from specular light interactions. Indigo's strictly unbiased rendering can have difficulty rendering such scenes cleanly using the normal sampling mode, and for these situations the [MLT sampling mode](#) can be of assistance.

HDR illumination is easy to set up, allowing for realistic and efficient illumination from distant light sources, as specified by Indigo Texture maps (illustrated here by the Uffizi Gallery HDR image by Paul Debevec):

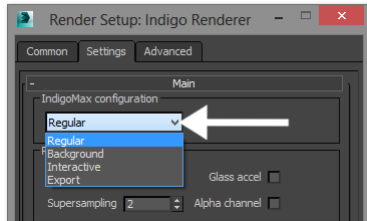


This concludes the basic usage tutorial. For more information on using Indigo please refer to the rest of the manual and our [Tutorials](#) section, and our Forum contains a wealth of useful information (besides being a great community to interact with!).

## IndigoMax configuration settings

IndigoMax supports 4 different basic configurations, or usage modes, to allow for greater workflow flexibility.

These configurations can be selected from the Render Setup menu, in the IndigoMax settings tab:



The 4 configurations work as follows:

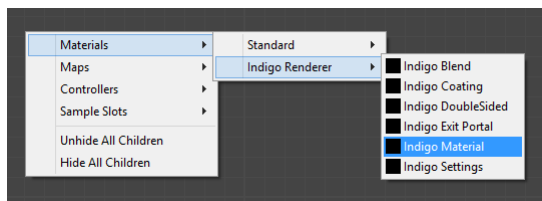
- Regular - Renders until the halting time or samples per pixel has been reached. This is the default mode, suitable for rendering animations. To cancel a render, press cancel on the render view.
- Background - Renders continuously, scene changes do not affect the render unless restarted. Not suitable for rendering animations. To stop a render, press the Stop button.
- Interactive - Renders continuously, scene changes (geometry, transforms, materials, cameras) cause the render to restart immediately. Not suitable for rendering animations. To stop a render, press the Stop button.
- Export - Exports a scene to disk (location is specified in the advanced settings tab), and optionally launches the standalone Indigo application to render the scene after export.

## Material settings

The Indigo Material in 3ds Max allows you to switch between the various basic material types Indigo supports, specifically Diffuse / Oren-Nayar for rough reflecting materials (e.g. matte paint, chalk), Diffuse transmitter for rough transmitting materials (e.g. lamp shades, curtains), Phong for smooth reflective materials (e.g. metals, plastic) and Specular / glossy transparent for water or glass (both smooth and frosted).

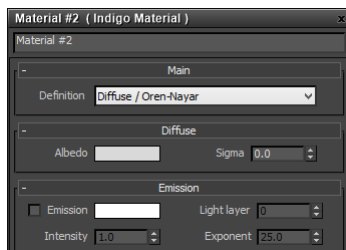
There are also three more complex Indigo material types, which use other Indigo materials as sub-materials: Blend, Coating and Double sided.

Indigo materials can be found in the Indigo Renderer sub-menu when creating materials:



## Diffuse / Oren-Nayar

For more information about the [Diffuse](#) and [Oren-Nayar](#) materials, please click the links to see the relevant documentation pages.

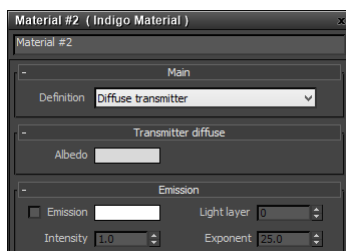


Diffuse material settings:

- Albedo - The material reflectance colour.
- Sigma - The roughness, where 0.0 is perfectly diffuse and 0.3 is very rough. For motion information on this parameter please see the Oren-Nayar documentation.

## Diffuse transmitter

For more information about this material type please see the [Diffuse transmitter](#) material documentation.

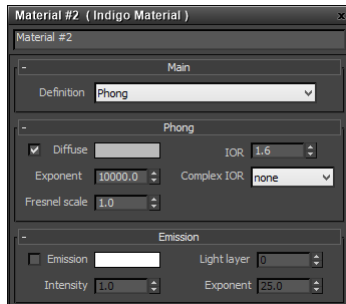


Diffuse material settings:

- Albedo - The material transmittance colour.

## Phong

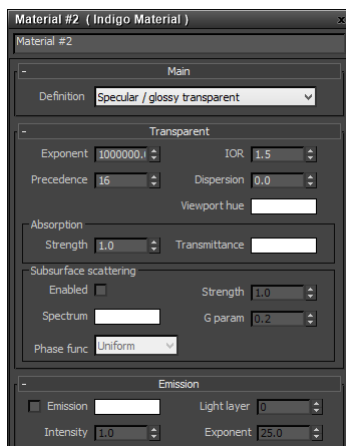
For more information about this material type please see the [Phong](#) material documentation.



- Diffuse - The colour of the diffuse base (or substrate).
- IOR - Controls the influence of the glossy coating; higher values produce stronger specular reflection. The IOR should be set to the real-world value for the material, if available; for example, the IOR of plastics is around 1.5 to 1.6, and oil paint binder has an IOR of around 1.5.
- Exponent - The exponent of the Phong specular highlight; larger values (e.g. 1000+) produce sharper highlights, while low values (e.g. 10) are quite rough.
- Complex IOR - Allows you to select from a list of lab-measured metal reflectance data. This overrides the IOR and Diffuse settings.
- Fresnel scale - Allows you to adjust the strength of the Fresnel reflection falloff.

## Specular / glossy transparent

For more information about the [Specular](#) and [Glossy transparent](#) materials, please click the links to see the relevant documentation pages.



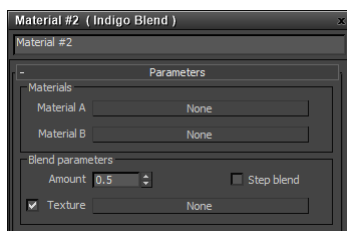
- Exponent - The Phong glossy reflection and refraction exponent. Set to a very large value (such as 1 million) for perfect specular / no glossiness.
- IOR - The index of refraction of the medium contained in this material; water has an IOR of ~1.33, diamond ~2.4.
- Precedence - The precedence of the medium inside this material. If there are multiple media occupying the same volume of space (e.g. ice in water in air), the one with highest precedence is used.
- Dispersion - Enables wavelength-dependent refraction (dispersion). Note that this can take

significantly longer to converge (tip: use MLT).

- Absorption Strength - Scales the material transmittance colour.
- Transmittance - The material transmittance colour.
- Scattering Strength - Scales the scattering spectrum.
- Scattering Spectrum - Defines the (possibly wavelength-dependent) scattering spectrum. Note that this can take significantly longer to converge (tip: use MLT).
- Phase function - The phase function used to describe scattering in the medium.
- G param - The Henyey-Greenstein G parameter, which specifies the preference for forwards ( $G > 0$ ), backwards ( $G < 0$ ) or uniform ( $G = 0$ ) scattering.

## Blend

For more information about the [Blend](#) material, please click the link to see the relevant documentation page.

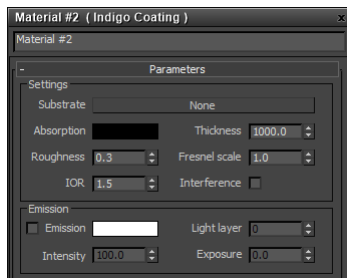


Blend material settings:

- Material A - The first material used for blending.
- Material B - The second material used for blending.
- Blend Amount - A constant blending factor; 1.0 means 100% of Material A is used, 0.0 means 100% of material B is used.
- Step blend - Disables partial blends so the result is either material A or B, depending on whether the blend amount (constant or map) is  $\geq 0.5$ .
- Texture - Allows you to use a texture map to control blending.

## Coating

For more information about the [Coating](#) material, please click the link to see the relevant documentation page.



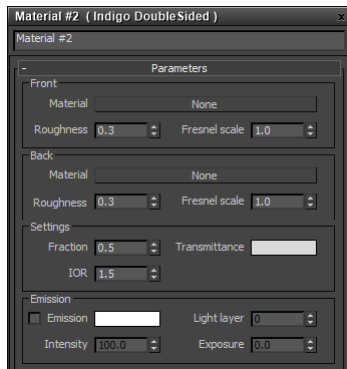
Coating material settings:

- Substrate - The base material which lies underneath the coating.
- Thickness - The coating thickness in millimetres.
- Roughness - Specifies the roughness of the coating, where 0 is perfect specular reflection and 1 is very rough.

- Fresnel scale - Allows you to adjust the strength of the Fresnel reflection falloff.
- IOR - The index of refraction of the coating medium.
- Interference - Enables the simulation of light interference in the coating, works best with thin coatings.

### Double-sided

For more information about the [Double-sided thin](#) material, please click the link to see the relevant documentation page.



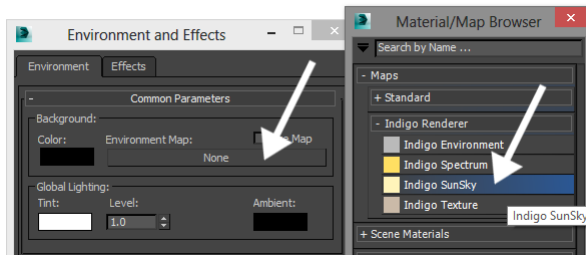
Front and Back material settings:

- Material - The material for the front- or back-facing side.
- Roughness - Specifies the roughness of the material's coating, where 0 is perfect specular reflection and 1 is very rough.
- Fresnel scale - Allows you to adjust the strength of the Fresnel reflection falloff.
- Fraction - The blending fraction between the front and back sides.
- Transmittance - The material transmittance colour.
- IOR - The index of refraction of the coating medium.

## Environment settings

The Indigo environment settings can be set from the normal 3ds Max environment settings, allowing a choice of physically-based Sun/sky or environment map illumination in your scene.

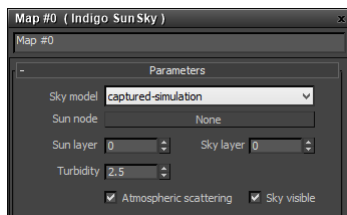
For example, to create a Sun/sky environment, open the Environment Settings dialog (keyboard shortcut: 8) and select the Indigo SunSky option from the Indigo Renderer Maps list:



The Indigo Sun/sky and environment map settings are described in the following sections.

### Physically-based Sun/sky

Indigo offers two Sun/sky models: either the classical Preetham Shirley Smits model, or the more accurate spectrally simulated model.

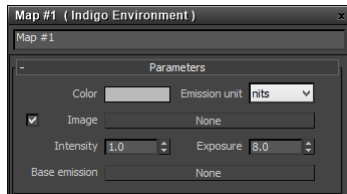


- Sky model - Selects the sun and sky model to use; the "classic" model is a standard physically-based sun and sky model, and the captured simulation is the [directly simulated sun and sky introduced in Indigo 3.2](#).
- Sun node - Sets the light source which will act as the sun. This must be specified in order for the sun and sky model to work.
- Sun layer - The light layer index in which the direct sunlight contribution is stored.
- Sky layer - The light layer index in which the skylight contribution is stored.
- Turbidity - A higher turbidity corresponds to more overcast skies. Currently only affects the original sky model.

### Image environment map

In this mode, an Indigo Texture (potentially in HDR) can be used as a background map to illuminate the scene. Usually latitude / longitude maps are saved in the compact .EXR format.



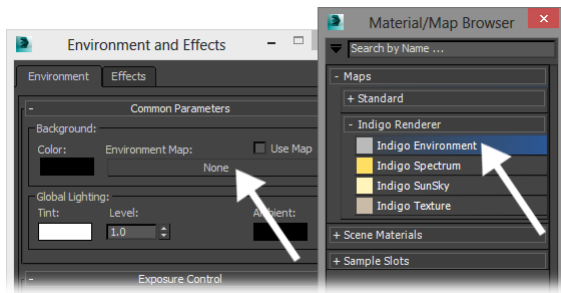


- Color - A constant background colour in the given emission units.
- Image - Allows you to specify an Indigo Texture map as an image source. For more information on setting up HDR environment maps, please see the subsection on [using image environment maps](#).
- Intensity - A linear brightness scale factor.
- Exposure - An exponential brightness scale factor.
- Base emission - Allows you to specify a given spectrum map as the basic illumination component, e.g. 6500K blackbody for "white". The Image and Color scale with this base spectral colour.

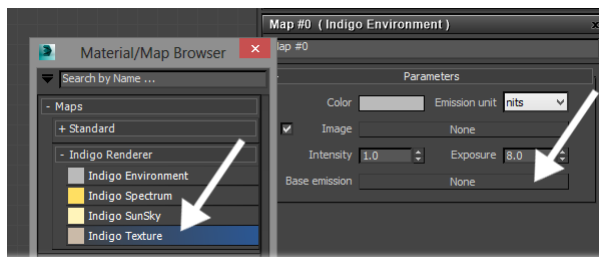
## Using image environment maps

In this section we'll cover using environment maps to illuminate a scene.

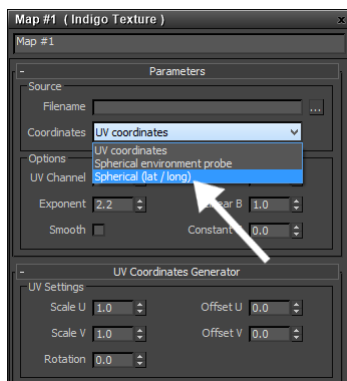
In a simple scene with the background clearly visible, open the Environment settings (keyboard shortcut: 8 key) and add a new Indigo Environment map:



In the Indigo Environment map settings, create a new Indigo Texture map for the Image channel:



In the Indigo Texture map settings, the coordinates used should be set to "Environment (lat / long)" if you are using a normal HDR environment map:



This is for the usual latitude / longitude format environment maps, which look like this (example by Paul Debevec):



If we render this setup with a simple "teapot on a pedestal" scene, we get the following result:



## Indigo for Blender



Old Room by Oren-Nayar

## Blendigo on Windows Tutorial

This tutorial will cover getting Indigo running with Blendigo on your computer. We will use the Blendigo exporter for Blender, that is used to export scenes to Indigo.

This tutorial is for Windows users.

If you have not purchased an Indigo licence, you can still follow this tutorial. Indigo will run in trial mode, which will apply some watermarks to Indigo renders.

### Step 1: Install Blender 2.58 or newer

If you already have Blender 2.58 or newer, you can skip this step.

Download and install Blender 2.58 or newer from here: <http://www.blender.org/download/get-blender/>

### Step 2: Download Indigo Renderer

The latest version of Indigo Renderer can be downloaded from this page:

<http://www.indigorenderer.com/download-indigo-renderer>

If you have a 32-bit operating system, or you are not sure, download Indigo Renderer for Windows 32-bit.

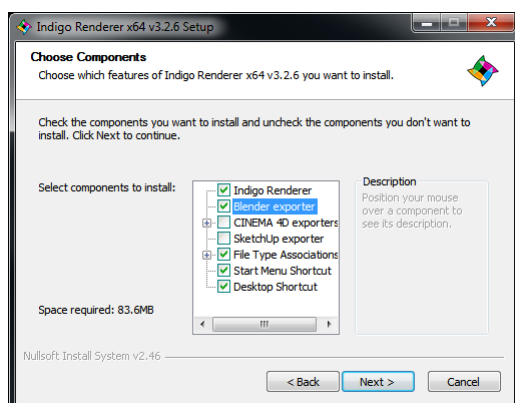
If you have a 64-bit operating system, download Indigo Renderer for Windows 64-bit.

### Step 3: Install Indigo Renderer

Once you have downloaded the Indigo installer program in step 2, run the installer program.

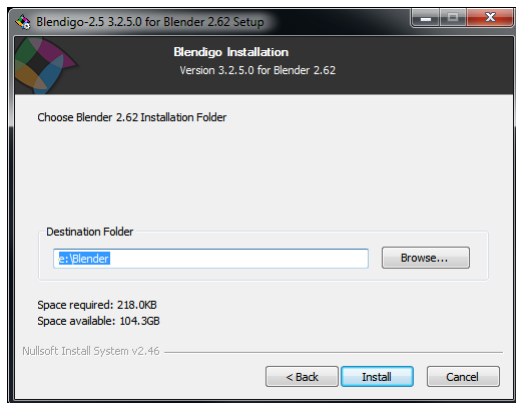
If the installer asks you 'Do you want to allow the following program to make changes to this computer,' select 'Yes.' Please carefully read the licence agreement, then click 'I Agree.'

On the 'Choose Components' page, make sure that the 'Blender exporter' is selected (otherwise Blender must be reinstalled), and press 'Next >'



On the 'Choose Install Location' page, leave the Destination Folder as it is, and press 'Install.'

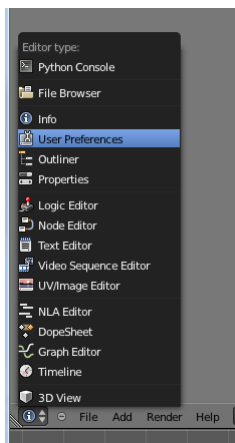
After the Indigo installer has completed, the Blendigo installer will launch. It should autodetect where Blender is installed (otherwise Blender must be manually located, or reinstalled):



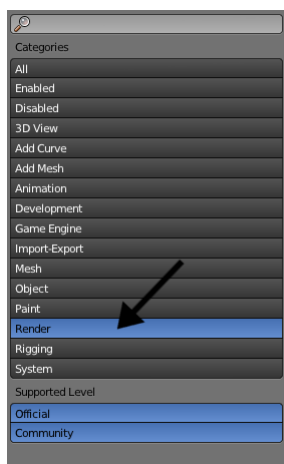
Press 'Finish'. Indigo will open after installation; close it for now, since we'll be using it via Blendigo.

### Step 4: Enable plugin in Blender

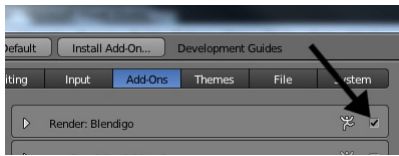
Select 'User Preferences' in the editor type selector:



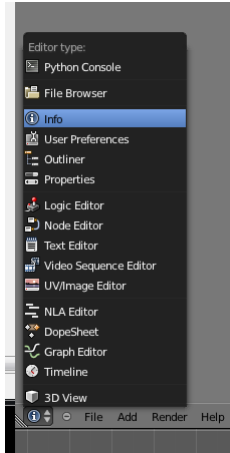
Then select the 'Render' category:



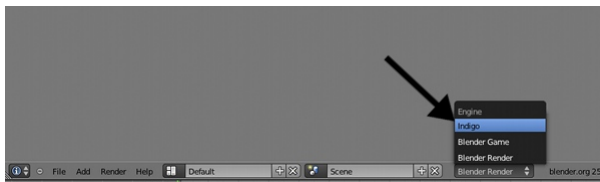
Then make sure the check-box on the right of the 'Render: Blendigo' box is ticked:



Select 'Info' in the editor type selector:



Now set the render engine to Indigo:



You should now be able to render scenes with Indigo by pressing F12.

## Installation

### 1. Getting the latest version of Blendigo

The Indigo installer comes bundled with the latest version of Blendigo, however it can also be downloaded stand-alone from our [Blender sub-forum](#), where it will be near the top of the top of the listed topics.

If you choose to use the stand-alone Blendigo installer, please note that you will still need to have the latest version of Indigo installed. For this reason we suggest using the Indigo installer to install Blendigo.

### 2. Ensuring the correct version of Blender is installed

Upon running the Blendigo installer, either from the main Indigo installer or the stand-alone installer, it will report the required version of Blender in the title, for example "Blendigo 3.6.25 for Blender 2.69".

If this version is not installed, install the appropriate version of Blender from <http://www.blender.org> before continuing.

### 3. Completing the installation

After you've verified that you have the appropriate version of Blender installed, allow the Indigo or Blendigo stand-alone installers to complete. If you're using the Indigo installer, ensure that the Blender exporter option was selected and installed.

As mentioned above, you must have Indigo installed in addition to the Blendigo exporter in order to render with Indigo from Blender. More detailed instruction for installing Indigo can be found [here](#).

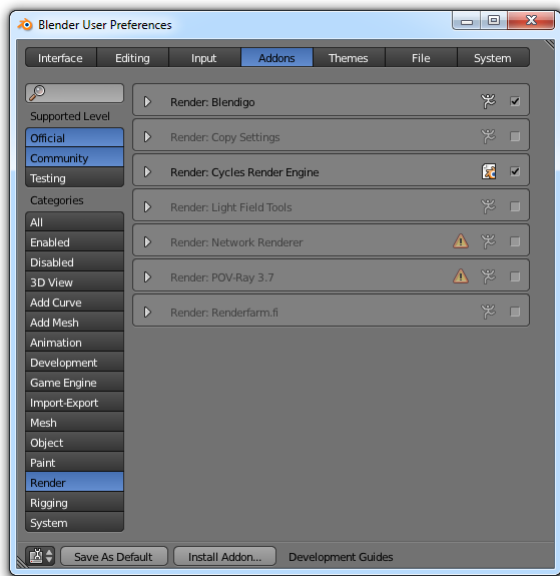
If the installer reports that it cannot find Blender on your system, either point it to your custom installation location, or follow the instructions from Step 1 again and ensure you install to the default locations. If you encounter further issues, please email us at [support@indigorender.com](mailto:support@indigorender.com).

### 4. Configuring Blender to use the Blendigo add-on

Open Blender, and from the File menu select "User Preferences". We want to enable the Blendigo add-on, so we select the "Render" filter from the list on the left near the bottom. Click the checkbox next to the "Render: Blendigo" option; it should become highlighted.

The dialog should look similar to this:



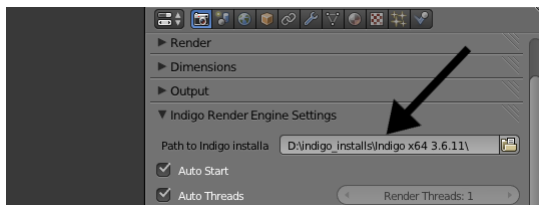


Click "Save As Default" so you won't have to do this every time you'd like to use Indigo as a renderer for Blender.

## 5. Setting the correct path to Indigo installation

You will need to make sure that the correct path to the Indigo executable is set in Blender:

You can find it here (Make sure Indigo is selected as the current render engine first - see next section)

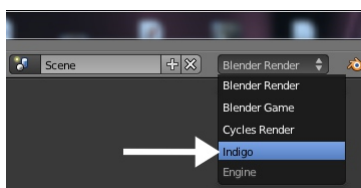


On Windows, the path should be to the directory that the Indigo executable is in.

On OS X, the path should end with "Indigo.app", *not* "Indigo.app/Contents/MacOS/".

## 6. Exporting a simple test scene

First we must set the current renderer to "Indigo" from the drop-down box at the top of the Blender window (the default being "Blender Render"):



If no lights are present in the scene, as is the case by default, then Blender will automatically add a Sun lamp to the scene.

Press F12 to initiate the export and render process, launching Indigo.

# Interface

Blendigo's functionality is tightly integrated into Blender, so the Indigo camera settings are located in the normal Blender camera options, the Indigo material settings are located in the normal Blender material options, etc.

For the most part these Indigo settings correspond exactly to those documented in the [corresponding manual section](#).

# Camera Settings

The settings in this section correspond exactly to the normal Indigo [camera settings](#), which should be referred to for more information about the various options.

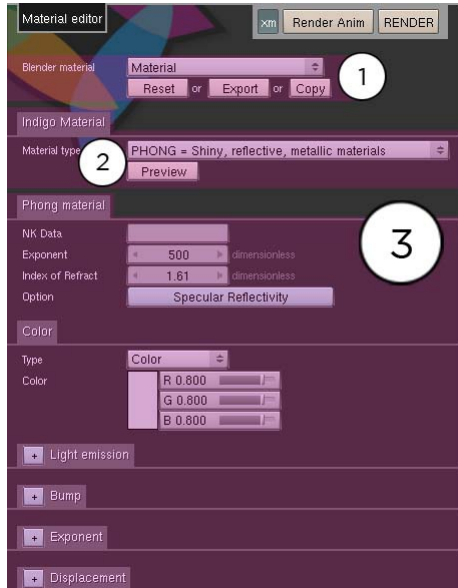
# Environment Settings

The settings in this section correspond exactly to the normal Indigo [environment settings](#), which should be referred to for more information about the various options.

## Material Editor

The materials setting page is used to configure the materials in the scene.

Blendigo will try and convert your Blender materials to Indigo materials as far as possible – but you are recommend to use Blendigo's material editor for more accuracy and control of your Indigo materials.



### 1. Blender material

This is the name of the blender material you are editing. When you select a new object, the objects material will be selected in Blendigo – so just right click on an object in your scene to select it in Blendigo.

**Reset** Reset to the original Blender material

**Export** Export to an Indigo Material file (.IGM).

**Copy** Copies the currently set Indigo material so you can copy it to another Blender material.

### 2. Indigo Material

See [Material Types](#)

### 3. Material Channels

See [Material Attributes](#)

# Mesh Subdivision

To use it you need to have blendigo open, then select a mesh in your scene and enable subdivision. Indigo subdivision works the same as the subsurf modifier available in Blender. It can be useful to use Indigo's built in subdivision because the models in Blender can have a low polygon count that are faster to work with – yet you still get a high quality render.

Subdivision in Indigo can also be helpful for getting smooth results on objects that have displacement mapping.

See [Mesh Subdivision](#)

# Render Settings

Renderer settings are for controlling the inner workings of Indigo. Most Indigo users will not have to modify from the default Blendigo settings.

See [Render Settings](#), [Render Mode](#)

# System Settings

System settings are for controlling the Indigo application. Most Indigo users will never have to modify from the default Blendigo settings.

See [Render Settings](#), [Network Rendering](#)



# Tone mapping

Tone mapping is the process whereby the high dynamic range (HDR) image internally stored by Indigo is converted to a low dynamic range (LDR) image for display on a normal computer screen.

The settings in this section correspond exactly to the normal Indigo [tone mapping](#) settings, which should be referred to for more information about the various options.

## Tutorials

## Section Planes Tutorial

*This tutorial applies to Indigo Renderer only.*

This is a short tutorial explaining the use of section planes with Indigo for Blender.

### Requirements

You will need to have Indigo for Blender (Blendigo) 3.4.9.1 or newer installed, as well as Blender and Indigo.

### Steps

#### Step 1

Start with an empty blender scene, then add a Suzanne mesh and a sun lamp.

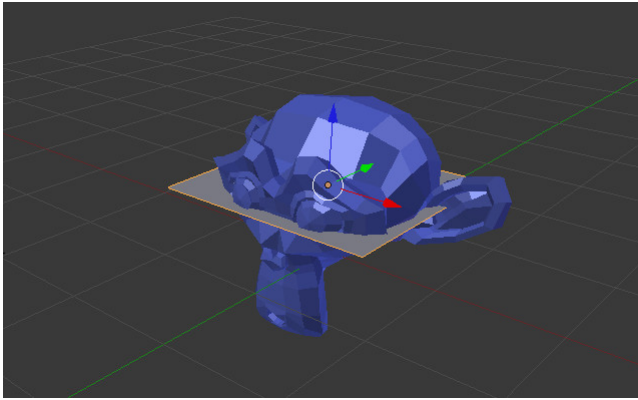


#### Step 2

Now we will add an object that represents the section plane.  
We will use a plane for this.

So add the object by using the 'Add' -> 'Mesh' -> 'Plane' command.

Position the object so that it intersects the Suzanne mesh. This is just so we can see exactly how the Suzanne mesh will be affected.

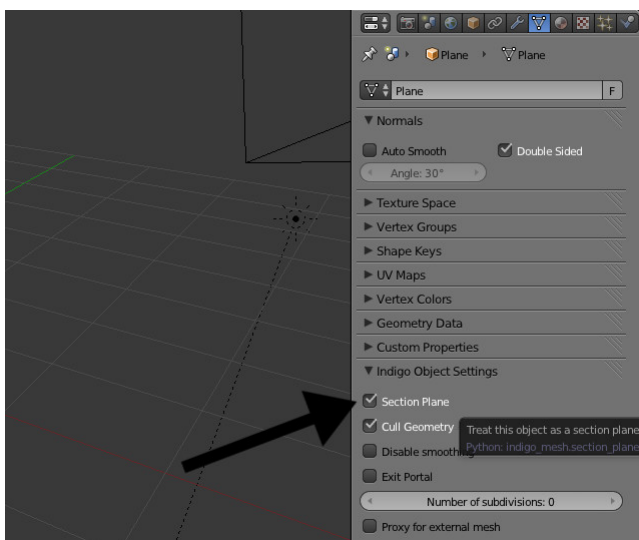


Everything below this plane will be removed or be made invisible.

We want to remove stuff above the plane, so rotate the plane by 180 degrees in the y axis.

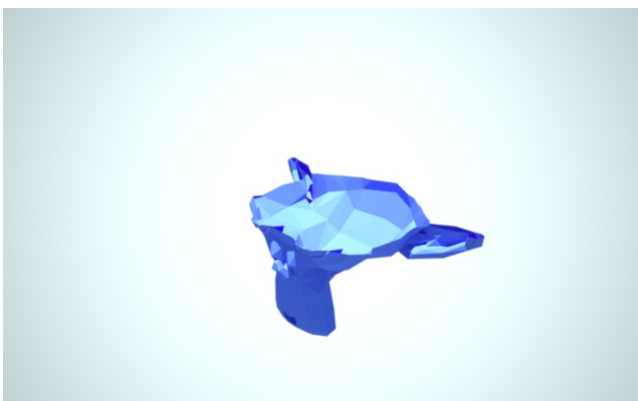
### Step 3

Now select the plane object, then enable the section plane tickbox. This will make the plane object into a section plane.



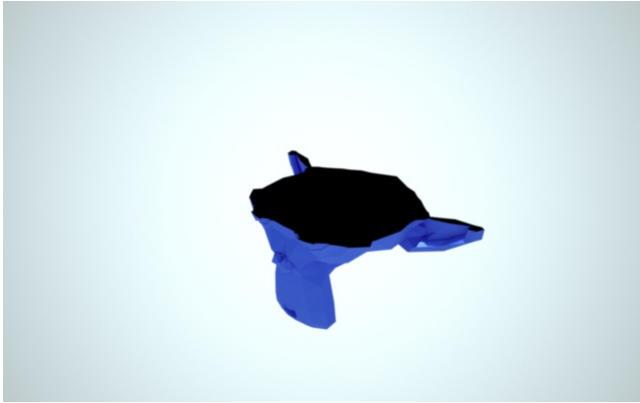
### Step 4

Now render the scene. It should look like this:



Because 'Cull Geometry' was ticked (the 'Cull Geometry' tickbox is located beneath the 'Section Plane' tickbox), everything above the section plane was removed. This allows light from the sun and sky model to illuminate the interior of the Suzanne model.

If 'Cull Geometry' is unticked, then the top half of the Suzanne model is still there, it's just invisible to the camera. This means that it blocks light from the sun and sky, resulting in a black interior:



### Tutorial Files



[Download the zipped .blend file for this tutorial.](#)

## Indigo for Revit

Indigo for Revit is currently in its beta phase and can be downloaded from <https://www.indigorenderer.com/revit/>



barcelona pavillon 2 by Camox

## Indigo for Revit Tutorial

This tutorial will cover getting Indigo running with Revit on your computer. We will use the Indigo for Revit exporter to export scenes to Indigo for rendering.

If you have not purchased an Indigo licence, you can still follow this tutorial. Indigo will run in trial mode, which will apply some watermarks to Indigo renders.

For this tutorial, we will assume that you have either Revit 2014 or newer installed already.

### Step 1: Install Indigo for Revit

Download and run the latest Indigo for Revit 'Free Trial' installer here:

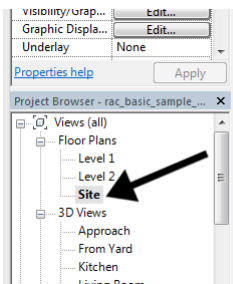
<http://www.indigorenderer.com/revit>

For more detailed information on installing Indigo for Revit, please see the [Installation](#) section of this chapter.

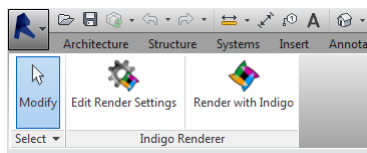
### Step 2: Rendering with Indigo from Revit

Start Revit and open the Sample Architecture Project.

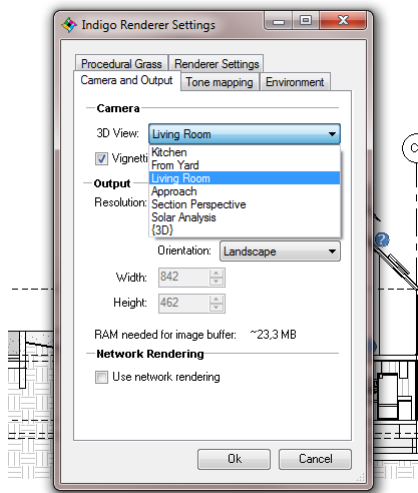
Due to limitations imposed on Revit plugins, Indigo for Revit cannot be used while in a perspective view. So we must first select a non-3D view from the views list, such as the Site view:



Having changed to a non-perspective view, we can access the Indigo Renderer section from the "Add-ins" menu:



Click on "Edit Render Settings", then in the options dialog select the Living Room view:



Click Ok to save the settings, then click "Render with Indigo". After a brief export process, it should launch Indigo Renderer and begin rendering the scene from the selected view, which should look like this (example from Revit 2014):



[Click to enlarge.](#)



## Installation

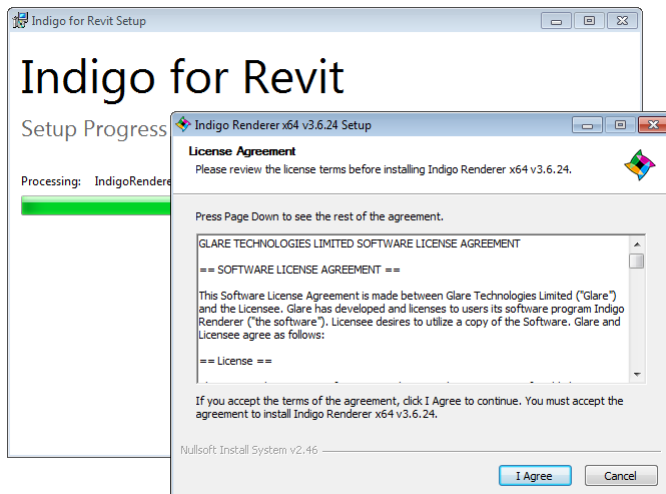
This guide will detail installing Indigo for Revit on your computer. Revit 2014 or newer must be installed first so that the installer can auto-detect the installation location.

The first step in the install process asks you to agree to the Indigo for Revit licence terms. Once you've read this, click the Install button; you will need an account with Administrator privileges to continue.

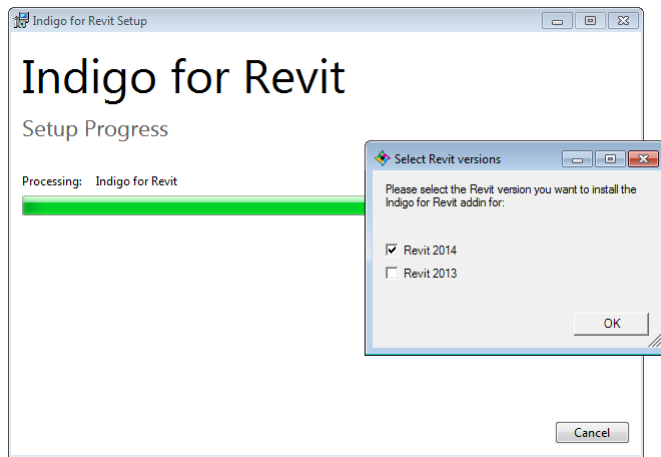


The first part of the installation process is installing the main Indigo Renderer application, so its installer will launch as part of the overall Indigo for Revit install process.

For more information on installing Indigo, please see the [relevant manual section](#).



Once Indigo has been installed, the Indigo for Revit installer will look for Revit installations and offer to set up the exporter addin for each:



After this has completed, Indigo for Revit will have successfully installed and is ready to be used.

## Material Editor

The Material Editor is used to configure the Indigo materials in the scene.

Indigo for Revit will automatically convert your existing Revit materials, however in Revit 2013 this conversion is somewhat limited by available information to plugins. To this end, the material editor allows you to edit the converted materials to make them more lifelike.

Revit 2014 exposes material definitions with lots of additional information compared to previous versions, allowing for a much higher quality conversion. Therefore, Indigo for Revit 2014 does not currently feature a material overriding system.

To create full custom Indigo materials from scratch, use the Indigo Material Editor that comes with your Indigo installation.

The online Indigo Material database is a warehouse for quality user-created materials that can be downloaded and used for free under the Creative Commons Licence. You can find it here: [Indigo Material Database](#)

Start by choosing the material type you would like to make, and then adjust the attributes below it. Changing the material type will change the options available.

Most materials have similar material appearance settings, allowing you to specify a single colour or an image map as the basic material colour. Image map is used to link to external texture files. Indigo can use PNG, TIFF, JPG/JPEG, TGA, BMP and EXR file formats. Bump maps add fine details to surfaces without creating more geometry, a great way to add realism.

Sample width specifies the size in mm of the image map , then tile.

See also: [Indigo Materials](#), [Material Types](#), [Material Attributes](#).

### Matte

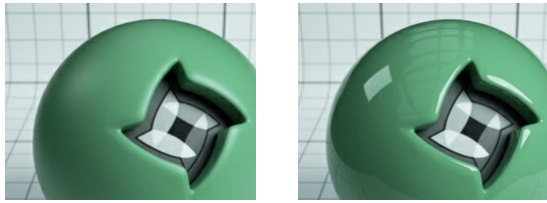
The matte material type models surfaces which reflect light almost equally in all directions, such as wall plaster, clay and other rough materials.

### Glossy

This material type can be used for shiny or glossy materials such as plastics, wood, fabrics etc.

### Smoothness

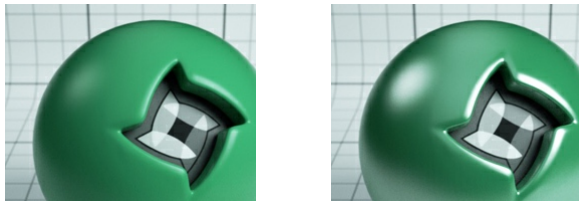
You can define how sharp the reflection is by the 'smoothness' slider. 'Rough' gives a matte surface that diffuses the reflection; 'smooth' gives a sharper, glossy reflection.



A rough glossy material    A smooth glossy material

## Reflectivity

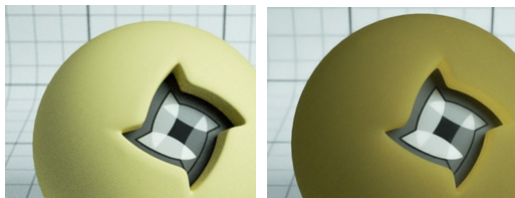
This value controls how much light is reflected by the surface. Insert values into the reflectivity box for fine control.



Rough with low reflectivity    Rough with high reflectivity

## Translucency

Translucency allows light to pass through the material surface and light the opposite side without being transparent or clear. Can be used for thin, coloured plastics, or even thin fabrics like a curtain or lampshade.



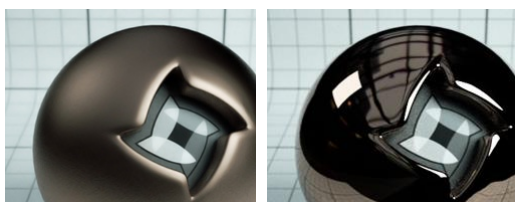
Fully Opaque

80% translucent

## Metal

The metal material type is used to create metallic surfaces.

You can define how sharp the reflection is using the smoothness slider. 'Rough' gives a matte surface that diffuses the reflection; 'smooth' gives a sharper, glossy reflection.

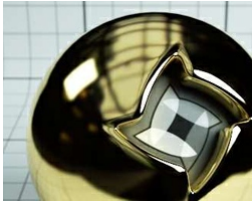


A rough metal

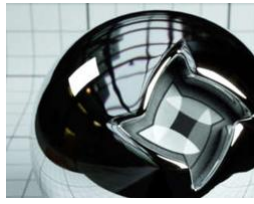
A smooth metal

## Lab-Measured Metals

Indigo comes with accurately measured metal materials that can be found here. They refer to their element names, as given by a Periodic Table.



Gold: Au



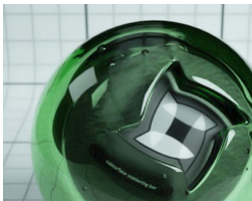
Aluminium: Al

## Transparent

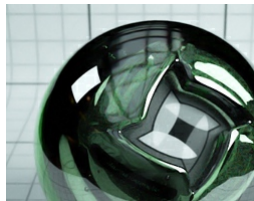
This is used for any materials through which light can pass e.g. glass, clear plastics or water. Frosted Glass introduces the following extra parameters:

## Reflectivity

This value controls how much light is reflected by the surface.



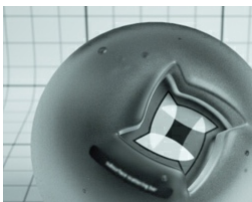
Low reflectivity



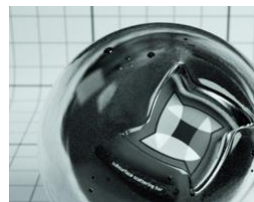
High reflectivity

## Smoothness

You can define how sharp the reflection is by the 'smoothness' slider. 'Rough' gives a matte surface that diffuses the reflection; 'smooth' gives a sharper, glossy reflection. This attribute is only available for Frosted Glass.



A rough frosted glass

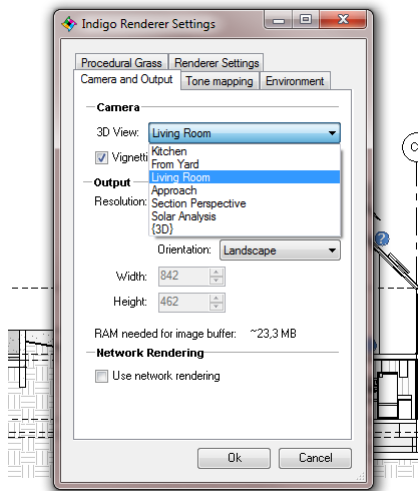


A semi-smooth frosted glass

## Render Settings

The Indigo for Revit Render Settings control how Indigo renders your scene.

Here is an example screenshot of the render settings window:



The Camera and Output tab contains options to select which 3D view to render from and the rendered image dimensions. Higher image resolutions will generally take longer and use more memory to render.

The Use network rendering option is useful for automatically starting network renders on export, which use the [Indigo Network Manager](#) to handle distributed rendering.

Tone mapping determines how the huge brightness range of real physical scenes gets compressed into the limited range displayable on computer screens; it is worth experimenting with these to get the best result for the particular scene being rendered.

The Environment tab contains options for the lighting environment; this can be either entirely background lighting (via physical sun and sky, or an HDR environment map), or optionally include the illumination from other light sources in the scene.

The Procedural Grass tab allows you to easily add fully 3D grass to your scenes, simply by choosing the material(s) on which grass should be scattered. The drop-down combo box lists the materials in the scene, which can be added to the list of grass scattering materials via the Add button; for example, adding the Grass material with the default settings produces a nice lawn in almost any scene. The view culling option is useful to only create grass in direct view of the camera, which speeds up the scene building before rendering; the grass density (in number of instances per square metre) can also be tweaked.

The Renderer Settings tab contains more advanced options such as options for [Metropolis Light Transport](#). This is an advanced method for computing illumination that enables faster rendering of complex scenes, however it is not recommended for simpler scenes.

## Using Indigo for Revit

After installing Indigo for Revit, a tool bar for Indigo Renderer will be added under the Add-Ins menu.

Please note that this menu is inaccessible in perspective views due to Revit plugin limitations.

### The Indigo for Revit workflow

To render your scene with Indigo, ensure that you have a 3D camera set up, then hit the Render with Indigo button in the Indigo Renderer addin menu.

The general Indigo for Revit workflow:

1. Create or position a 3D camera to be used for rendering.
2. Edit the Render Settings.
3. If you are using Revit 2013, [tweak the materials and their attributes using the Indigo Material Editor](#).  
Revit 2014's material system is vastly improved, and no manual tweaking is necessary for accurately converted and beautifully rendered materials.
4. Render with Indigo.