

Baking Next-Gen Illumination to Point Clouds

In this tutorial we are going to check out Turtle's new Point Cloud baking tools.



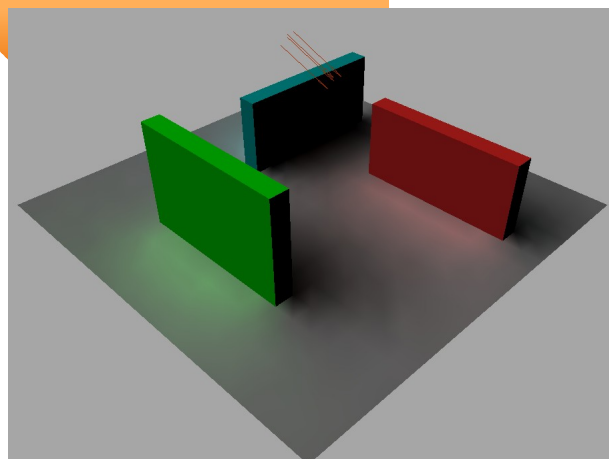
ILLUMINATE LABS™

LIGHTING THE NEXT GENERATION OF GAMES

In this tutorial we will work with Turtle 5.1's point cloud baking feature. We'll use spherical harmonics to capture different types of light for different purposes.


The tutorial is divided in two main sections, the first handling dynamic lighting, which can be used for dynamic shadowing and color bleed, and the second handling static lighting, which can be used to generate arbitrary points in space containing irradiance information. In a game, for example, these two types of spherical harmonics can be used in conjunction to produce diffuse environment shading of rigid objects moving through a scene with static lighting.

The information from the bake can be stored in xml format, making it easy to use or reformat the information in your own application.

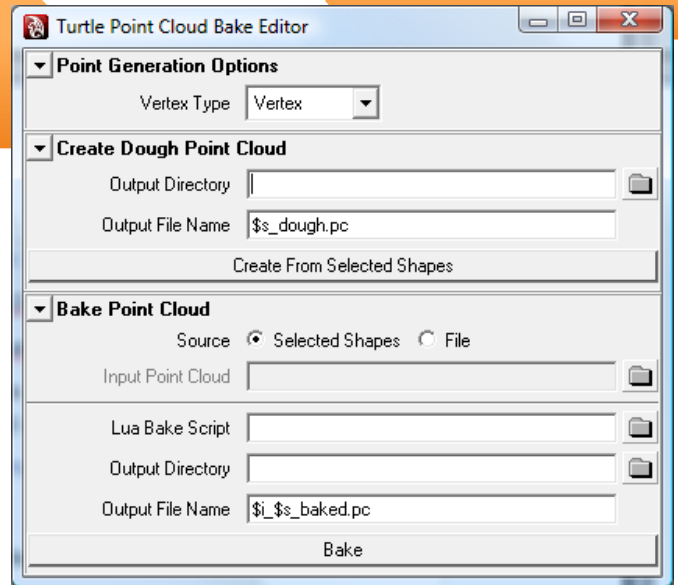
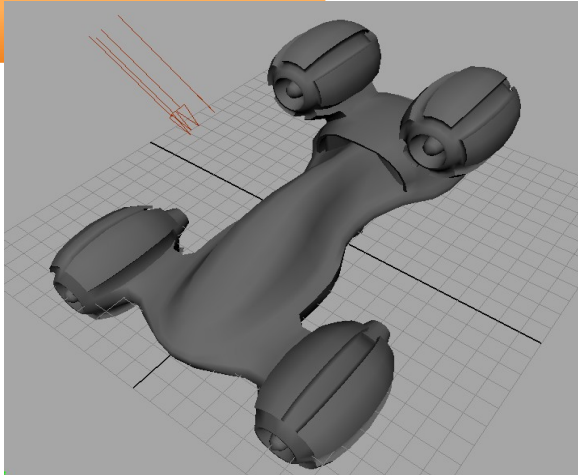


Baking Dynamic Self-Shading

We'll start off by baking dynamic shadowing on a craft model.

 Open the [craft.mb](#) scene in the point cloud baking project.

Next open the point cloud bake editor, available through [Window](#) → [Rendering Editor](#) → [TURTLE](#) → [Point Cloud Bake Editor](#).



The editor consists of three parts, [Point Generation Options](#), [Create Dough Point Cloud](#) and [Bake Point Cloud](#). The first part contains the [Vertex Type](#) option, specifying whether to use the vertices or face vertices to generate points. The difference between a vertex and a face vertex is described in the Maya


documentation. The second part is used to generate a point cloud file which can be used as an input for the point cloud baking step. We call this type of input file a 'dough' point cloud. A dough point cloud does not necessarily need to be generated by Turtle, you can create your own dough point cloud file to use for baking as well. This is useful if you want to set up the points and bake externally. Simply select a shape, change the output file extension to 'xml' and hit [Create From Selected Shapes](#) to get a template file to work from.

Later in this tutorial we will show how to create and use dough point clouds generated directly from shapes.


 In the point cloud bake editor, use the following attributes:

- Source: Selected Shapes
- Lua Bake Script: 'turtle/bakeScripts/sh_surface_shadow.lua'
- Output Directory: 'turtle/pointClouds/'
- Output File Name: '\$s_baked.xml'

Setting the output file extension to 'xml' will force Turtle to output the data formatted as an xml file. '\$s' in the file name will be replaced by the source shape name, and '\$i' will be replaced by the input point cloud file name, if baking from a file.

 Now we are ready to bake the information we want. Make sure the craft is selected in the viewport, then hit [Bake](#). The craft is geometry intense, and since we're baking to face vertices, the rendering may take a little while. Now you will

have a fresh xml file in the point cloud output directory.


 Next, we want to actually see the results. Turtle offers two ways to do this, either via a point cloud shape, or by applying an [ilrHwBakeVisualizer](#) to the craft material. Let's start with a point cloud shape. You can create the node from the Turtle shelf if loaded, otherwise run the following command in the script editor:

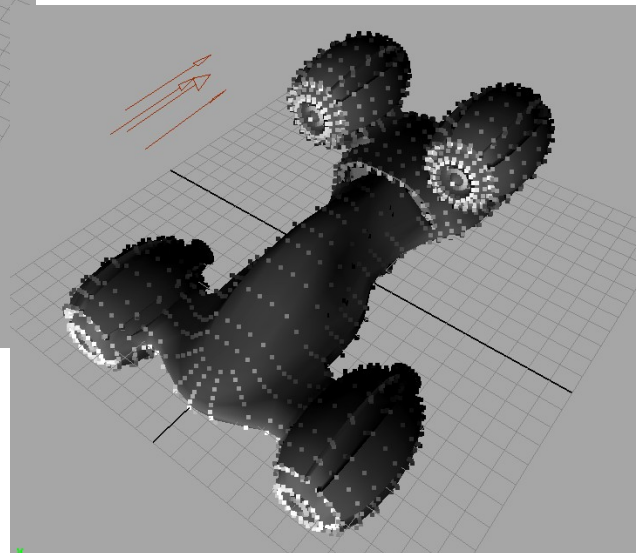
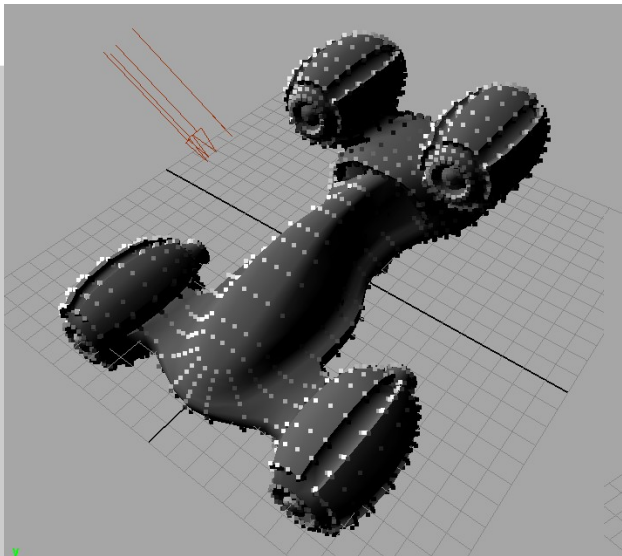
```
createNode ilrPointCloudShape;
```


Now to set up the point cloud shape to display our baked spherical harmonics:

- Input File: generated xml file
- Enable: Use Point Colors
- Point Size: 5
- Displace Along Normal: 0.1
- Coordinate System: Object Space

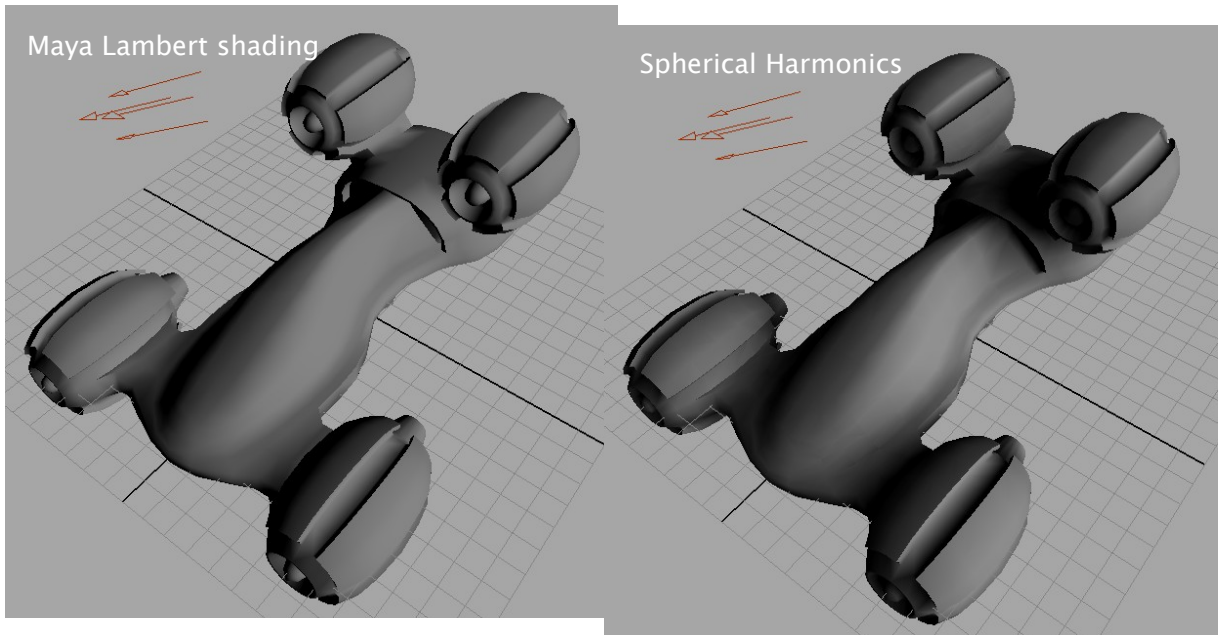
We displace the points a bit along the normals to avoid depth cull by the craft. [Coordinate System](#) should be set to [Object Space](#), since the spherical harmonics were baked in object space.

 In the viewport menu, set [Lighting](#) → [Use All Lights](#) to enable the use of the directional light in the scene. Note that for spherical harmonics evaluation, only directional lights will be used. Rotate the directional light and see how the lighting in the points changes.



 Next, we want to apply the spherical harmonics directly on the craft. Start by hiding or deleting the point cloud shape. In the viewport menu, enable **Shading** → **Hardware Texturing**. Select the 'craftMaterial', and connect an **IlrHwBakeVisualizer** to its **Hardware Shader** attribute (available in the **Hardware Shading** roll out).


In the **Spherical Harmonics (Vertex Lighting)** roll out, set **Coordinate System** to **Object Space** and set **Input File** to the generated xml file. We also need to enable the effect, so check the **Spherical Harmonics (Vertex Lighting)** checkbox in the **Hardware Shader Features** roll out.




Notice how Spherical Harmonics are able to capture the self-shadowing of the craft body. You can really see it when the engines cast their shadows onto the fuselage. No matter how you turn the directional light, the shadows will dynamically update to match the lighting.

Baking Dynamic Shadows

Dynamic illumination can be used for shadowing in space as well as on surfaces, and as a result, for simple occlusion as well. In this example we will bake shadowing information to a set of points in space. The points we bake will be defined by the vertices of a piece of geometry.

 Open craft.mb again.

 Create a cube. Set the scale to 25 in order to enclose the craft, and set the subdivision of the cube to 20 in all directions. This will give us a nice set of vertices to generate points from.

The cube will be used as a source for points, but we don't want the geometry of the cube to affect the gather of dynamic illumination. Therefore we need to first create a Dough Point Cloud file from the cube shape. After that we can remove the cube from the scene and use the newly created file as input for the baking.

 Open point cloud bake editor ([Create Dough Point Cloud](#) roll out)

- Output Directory: 'turtle/pointClouds/'
- Output File Name: '\$s_dough.xml'

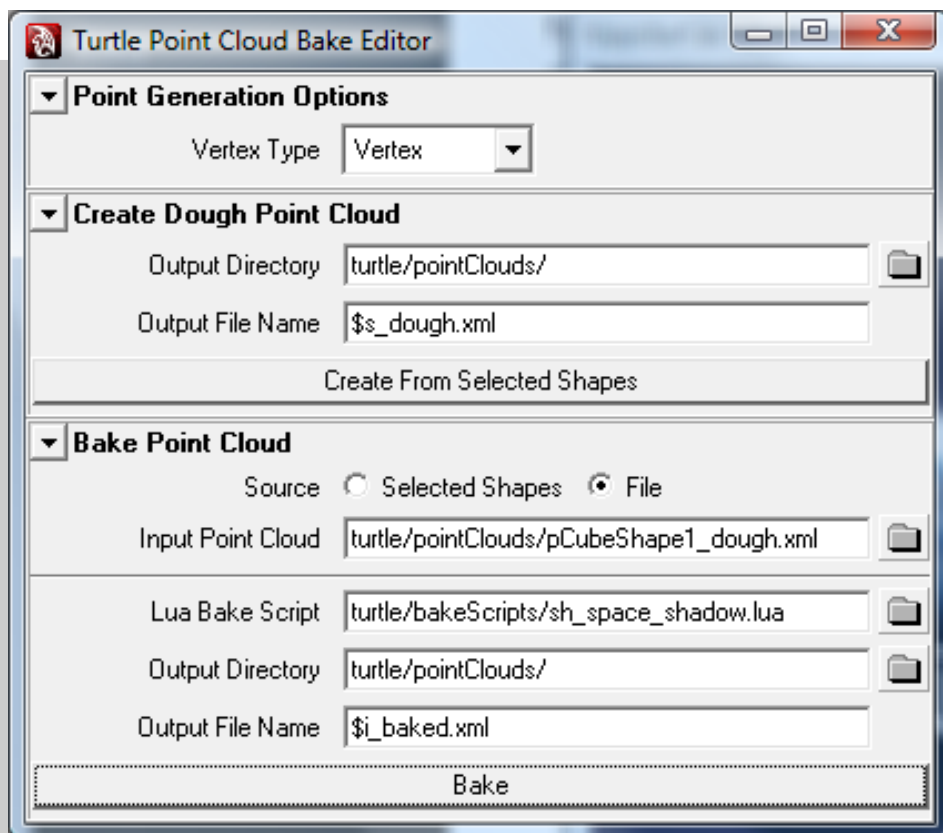
Now select the cube and hit 'Create From Selected Shape'. From now on we won't need the cube anymore so go ahead and remove or hide it from the scene.

 Open point cloud bake editor ([Bake Point Cloud](#) roll out)


- Source: File
- Input Point Cloud: generated dough xml file
- Lua Bake Script: 'turtle/bakeScripts/sh_space_shadow.lua'
- Output Directory: 'turtle/pointClouds/'
- Output File Name: '\$i_baked.xml'

The main difference between this LUA shader and the one used in the previous example, is that this time we do not cosine weigh the light falling in on the point, and we're baking in world space.


 Hit 'Bake' in the point cloud bake editor.

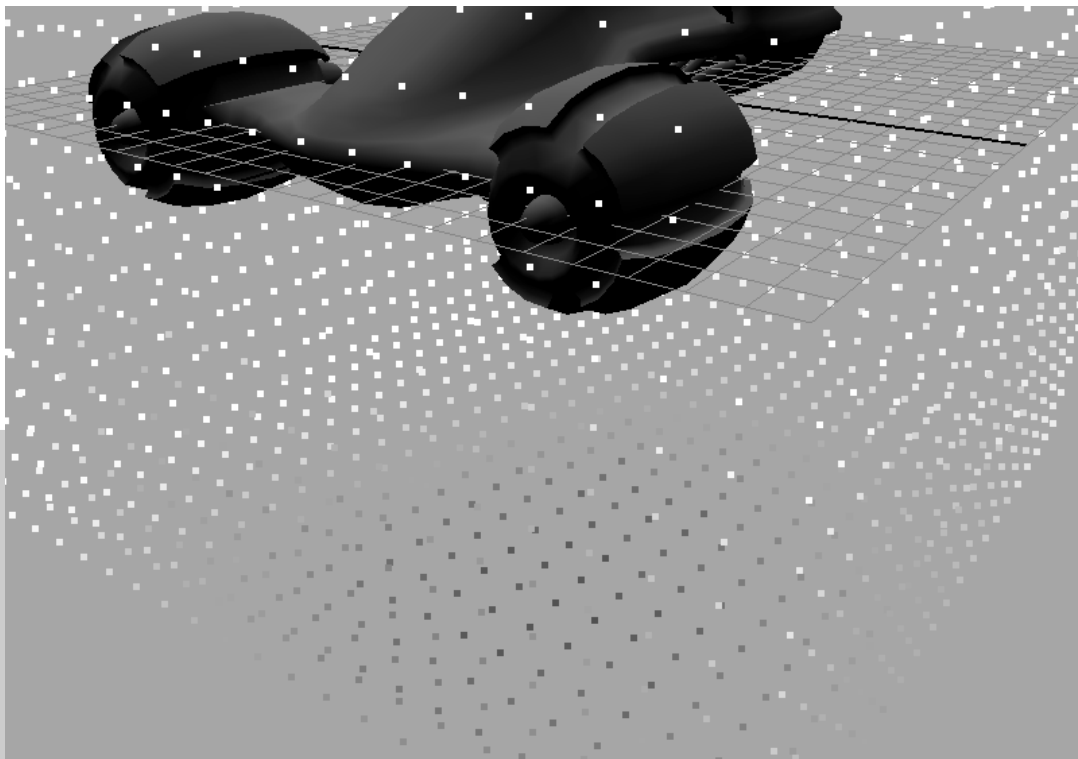


Now we don't want to apply the points to any geometry, so we'll use the point cloud shape to visualize the point cloud.

 Create a point cloud shape.

- Input File: generated point cloud xml file
- Enable: Use Point Colors
- Point Size: 5
- Coordinate System: World Space

 In the viewport menu, make sure [Lighting → Use All Lights](#) is set.



Note how the shadows from the craft were captured in the points, so that they are shadowed correctly no matter how you orient the directional light. If you create a point cloud adapted to your scene geometry, you will be able to bake really cool shadowing effects for use in real-time graphics.


Full Dynamic Illumination

In this example we will bake dynamic illumination, including color bleed, to a surface.


 Open the [color_bleed.mb](#) scene.


In order to get dynamic color bleed, we need to use the [Dynamic Photon Map](#). The dynamic photon map works roughly the same way as the regular grid based photon map, but instead of emitting photons from static lights, photons are emitted from a sphere surrounding the scene. In the final step this photon information is converted to spherical harmonics, which represent the dynamic lighting in the scene.

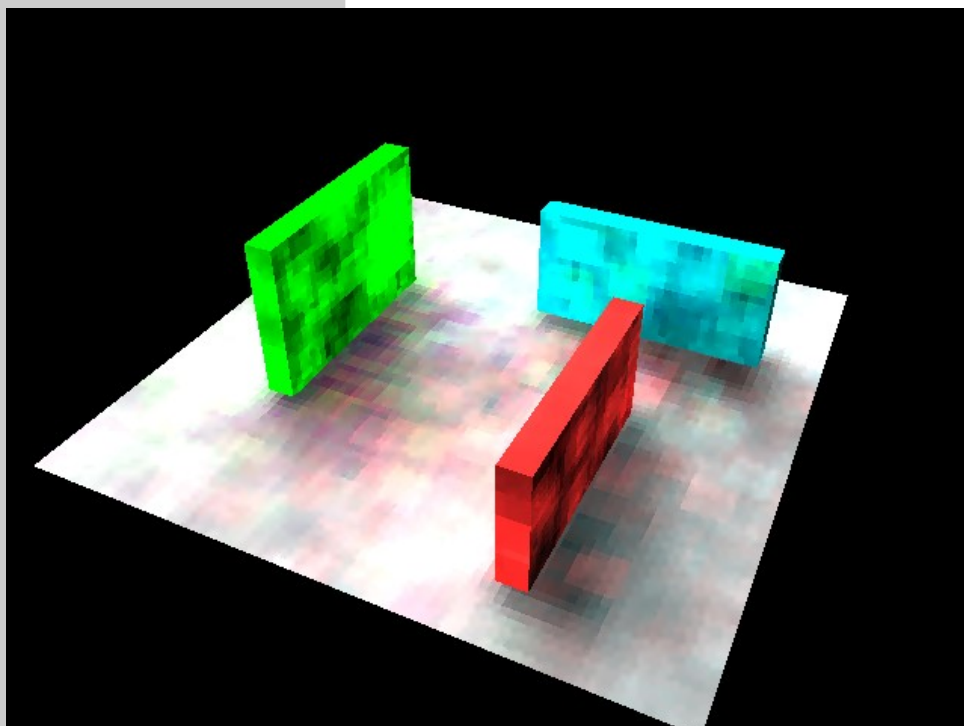
We will generate a dynamic photon map for the scene and preview it before using the map for baking. When previewing the dynamic photon map, you set a light direction from which to evaluate the map. This will then be used as an ambient light source to light the scene during regular rendering. In order to get a good preview of the map, we want to make sure no other lights are used for lighting the scene.


 Make sure there are no lights in the scene and disable the default light.

 In Turtle's render settings, open the [Global Illumination](#) tab. Enable GI and set both primary and secondary GI to [Dynamic Photon Map](#).


 To get a more pronounced color bleed effect we pump up the intensity of the indirect light. Set the [Secondary Intensity](#) setting in the [Color Balance](#) roll-out to 20 and the [Primary Intensity](#) to 5.


 Now let's render the scene. The result will look somewhat blocky, but since this information will only be used for indirect light, that is not a problem.



 Now we're ready to use the map for baking. Open the [Point Cloud Bake Editor](#).

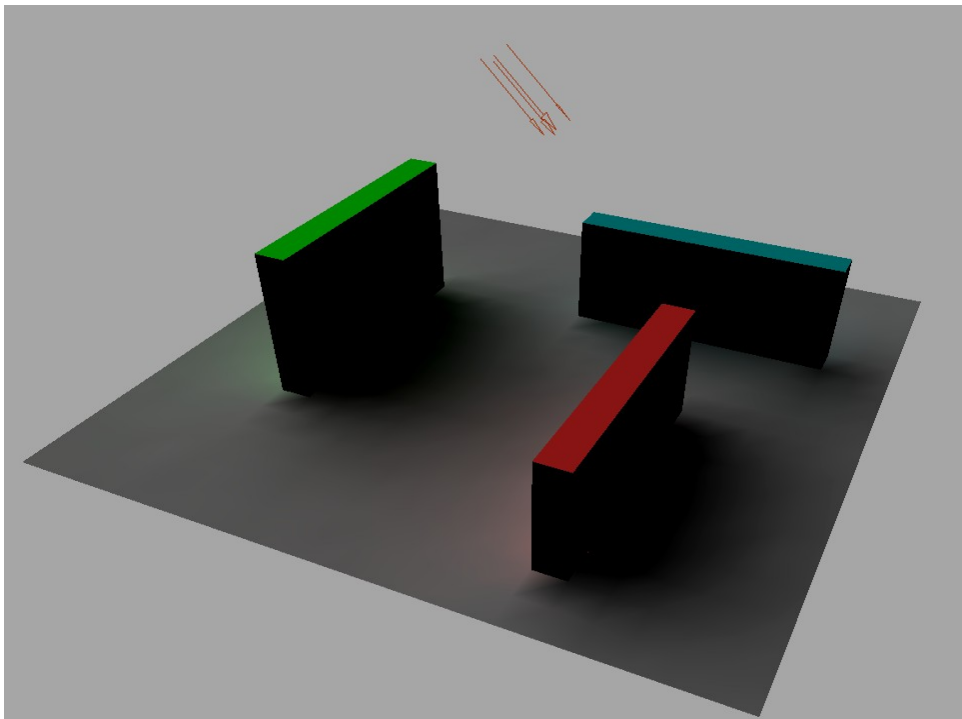
- Source: Selected Shapes
- Lua Bake Script: 'turtle/bakeScripts/sh_surface_color_bleed.lua'
- Output Directory: 'turtle/pointClouds/'
- Output File Name: '\$s_baked.xml'

 Select the plane and hit [Bake](#) (make sure the dynamic photon map is still enabled).

 Enable [Hardware Texturing](#) for the viewport, set [Use All Lights](#) and create a directional light. Connect an [ilrHwBakeVisualizer](#) to the [Hardware Shader](#) attribute of the plane's material. In the created [ilrHwBakeVisualizer](#), for [Spherical Harmonics \(Vertex Lighting\)](#), set:

- Input File: generated point cloud file
- Coordinate System: Object Space

Also make sure to enable the effect in the [ilrHwBakeVisualizer](#) node's [Hardware Shader Features](#) roll out.




The result shows the color bleed effect we have baked down to spherical harmonics. Since direct light was included in the computation, we also get self shadowing. Rotate the directional light source, and the effect will change accordingly.


In the next sections we will show you how to bake diffuse static lighting from the environment to spherical harmonics in a point cloud.

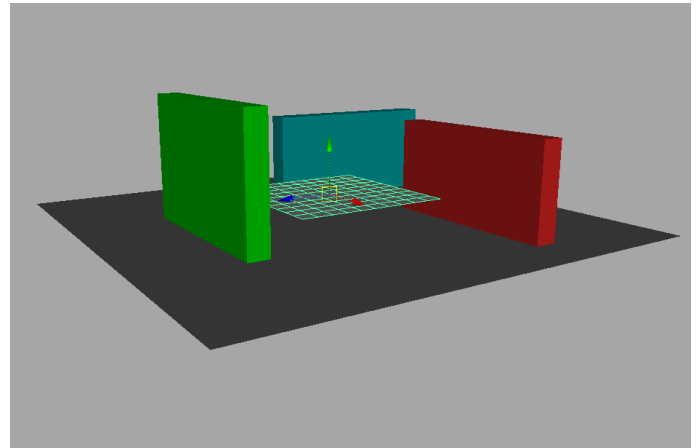
Indirect Static Lighting

We will start off by baking indirect light. For this purpose, we can take advantage of Turtle's regular photon maps, but in this example, we're only going to bake one depth of indirect light and so don't need to enable a photon map.

 Open [color_bleed.mb](#).

 In order to get indirect light, we need a light source. Create a point light and position it roughly 10 units above the ground plane.


 To define a set of points to bake to, create a plane, scale it up to about 5.0 and move it a bit above the ground. Set subdivision to 10 in both directions.



Create a dough point cloud from the plane. We don't want the geometry of the created plane to interfere with the gather.

 Open point cloud bake editor ([Create Dough Point Cloud](#) roll out)

- Output Directory: 'turtle/pointClouds/'
- Output File Name: '\$s_dough.xml'

 Now select the plane and hit 'Create From Selected Shape'. From now on we won't need the plane anymore so go ahead and remove or hide it from the scene.


 Open point cloud bake editor

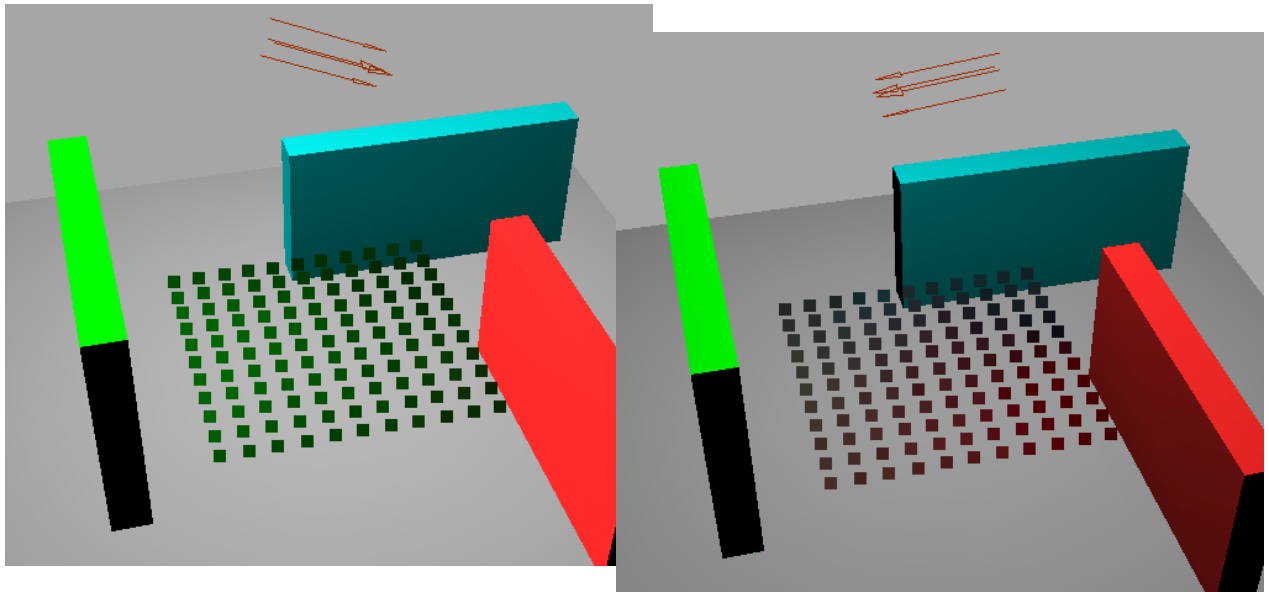
- Source: File
- Input Point Cloud: generated dough xml file
- Lua Bake Script: 'turtle/bakeScripts/sh_irradiance_indirect.lua'
- Output Directory: 'turtle/pointClouds/'
- Output File Name: '\$i_baked.xml'

 Hit [Bake](#).

 Create a point cloud shape and assign the created file.

- Enable: Use Point Colors
- Point Size: 5
- Coordinate System: World Space

 Create a directional light and make sure [Lighting → Use All Lights](#) is enabled in the viewport menu.



What the point cloud is visualizing is the amount of static light that passes through each point. The spherical harmonics function in each point is simply evaluated in the direction determined by the directional light. Note that the intensity and color of the directional light still affects the evaluated color of each point.


Full Static Lighting

Now we will continue the previous example and add direct light to the spherical harmonics information.


 Open [Point Cloud Bake Editor](#) and change the Lua script:

- Lua Bake Script: 'turtle/bakeScripts/sh_irradiance.lua'

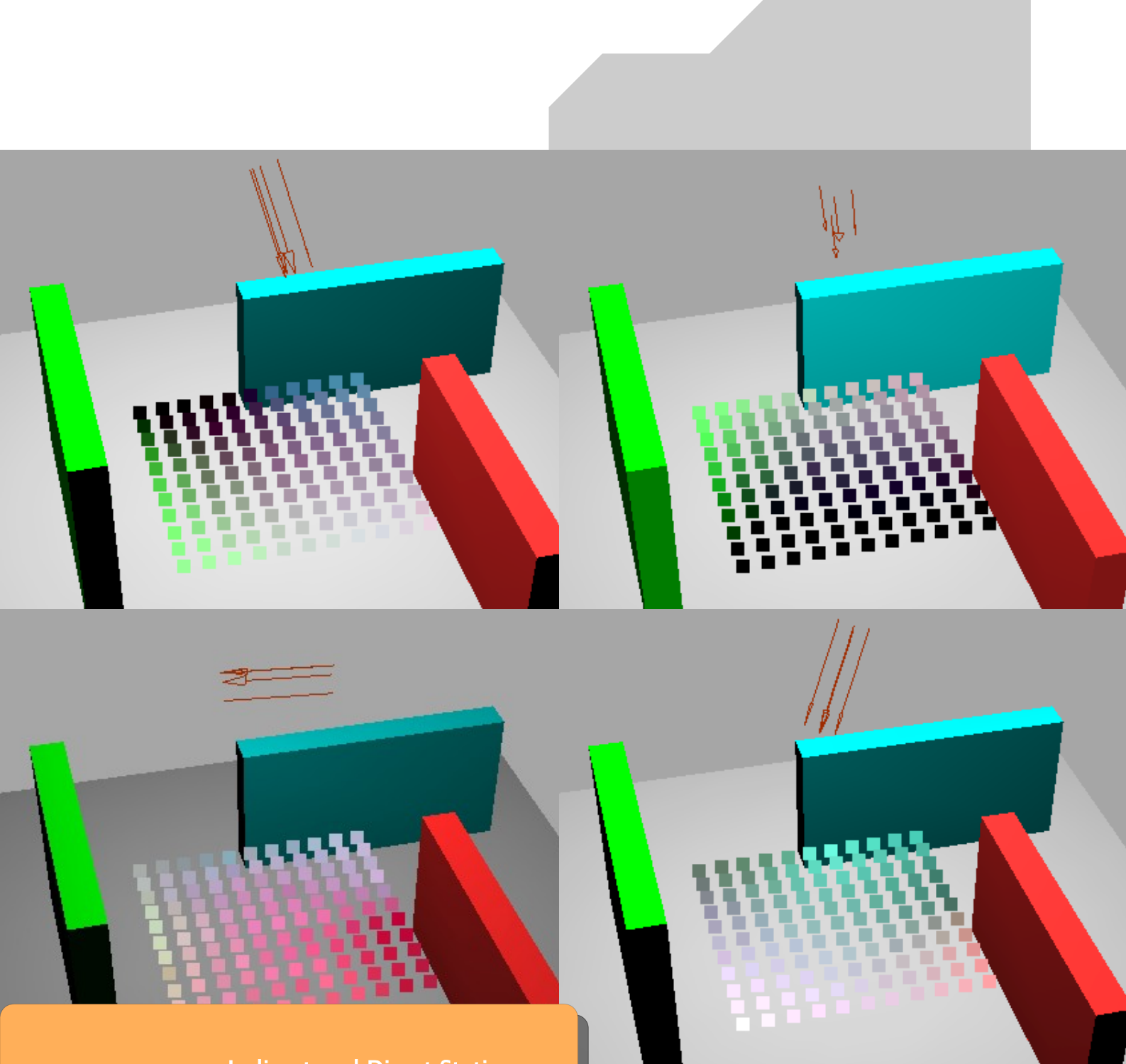
This script has been set up to add static direct light and boost the indirect light a bit.

 We only want direct light from the point light, so set the [Intensity](#) of the directional light to 0.0.

 Now [Bake](#).

 In the point cloud shape we used to visualize the previous point cloud, press the [Reload](#) button to read the updated point cloud file (assuming you didn't change the file name from the last example).

 Now reset the [Intensity](#) of the directional light to 1.0



- Indirect and Direct Static Lighting
- 16-coefficient Spherical Harmonics
- Static Color Bleeding
- Full Spherical Sampling

Get your eval license of Turtle now!
www.illuminatelabs.com