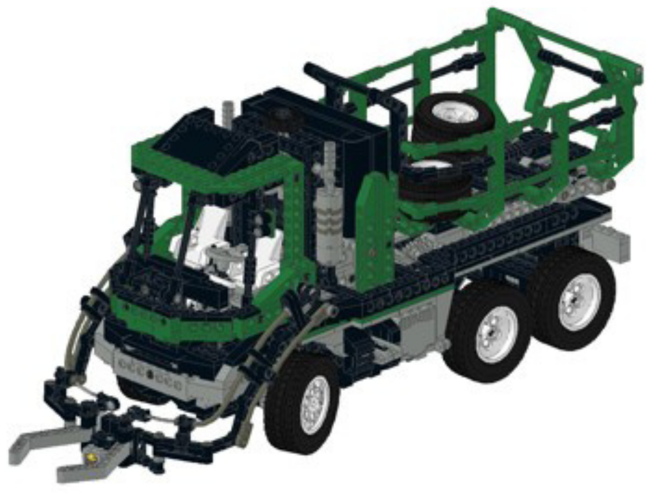
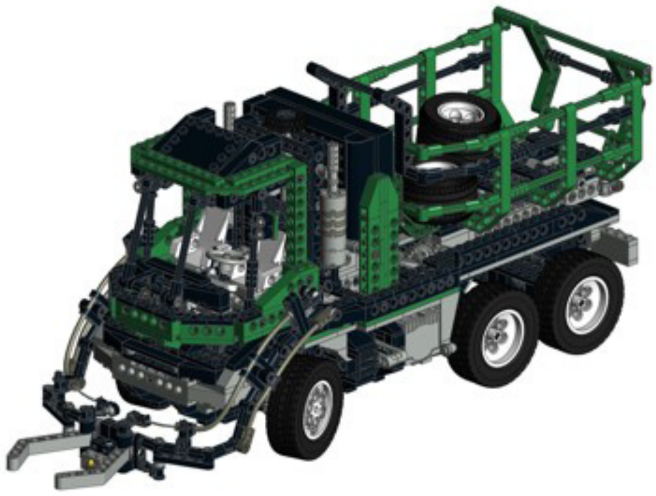
POV-Ray Tutorial

By Eric Albretch

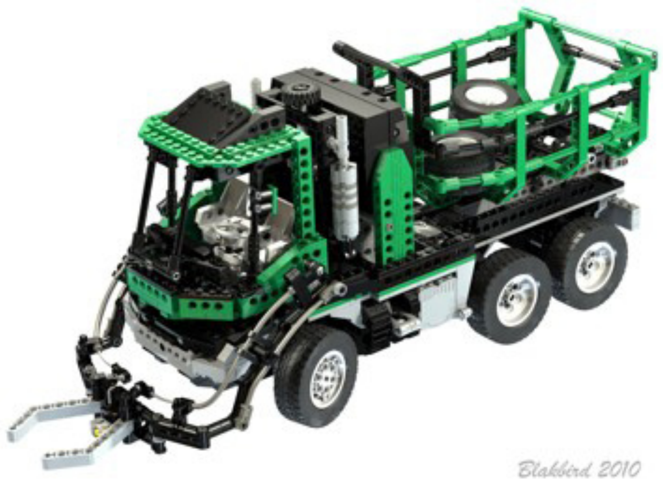
If you have been reading Jetro's LDraw tutorials in HispaBrick Magazine® or on TechnicBRICKs, then you already know about the LDraw parts library and about LDraw editors like MLCAD. You know how to install unofficial parts (parts which are not officially released as part of the LDraw library yet), create virtual models, and how to look at them in viewers like LDView. Maybe you can even generate complex flexible parts with LSynth. Perhaps you have tried making your own instructions with LPub. Now that you have spent so much time making a virtual model, the logical next step is to make it beautiful. Editors like MLCAD are efficient and quick at displaying your models, but they do not use complex lighting or smoothing so you get something like looks like this



Standalone viewers like LDView use real-time rendering and, with the appropriate preferences selected, can produce a much more striking image. This image has smooth shading and anti-aliased edges. It looks good, but it still doesn't look real.



To really make your models shine, you need to use a ray tracing program. These are complex programs with very advanced capabilities, but the rewards for learning them can be stunningly realistic images. This image was produced by starting with the same LDraw file used for the first image, but converted, enhanced, and rendered using POV-Ray, a free (open source) ray tracer.



This tutorial will start by assuming that you have used the LDraw All-in-One Installer and therefore already have the following necessary software:

· LDraw library
· MLCAD (or another editor)
· LDView 4.1 or newer (PC or Mac)
· POV-Ray 3.6* (PC or Mac)
· LGEO library*

Items marked with an asterisk (*) are not installed by default, so you may have to go back and install again and make sure to select them. If you are using a Mac, you can still do everything in this tutorial but you will have to install the needed software manually because there is no All-in-One installer. In addition to the above software, some advanced features in later tutorials will require Mega-POV, a POV-Ray patch, which you may choose to install now. (http://megapov.inetart.net/)

A knowledge of the different types of files and how they work will be invaluable in troubleshooting later. Let's start by reviewing what LDraw files really are, at their core. LDraw files consist of nothing but lines, triangles, and quadrilaterals. For purposes of rendering, we don't care about the edges because they will never be displayed in a photo real image, so let's forget about them. Triangles and quads can only create surfaces, not solids, so LDraw parts are hollow. Triangles and quads can also never create curves, but can only approximate them with facets. This means that the LDraw language is not very good for making smooth, complex curvature.

LDraw parts are stored as text files in a library. Each part file usually calls out "primitives" which are smaller portions of parts which are used over and over (such as studs). An LDraw model file is also a text file which calls out many parts, assembles them in space, and assigns them colors. The position and rotation in space are controlled with a transformation matrix. You don't need to know how a transformation matrix works to use these tools, but it helps. Below is a line from an LDraw model file which positions a single red 2x4 brick in space. The first number, 1, just means that this line is calling a part from the library. The second number, 4, is the color red. The next 3 numbers, 0 0 0, are the position of the part in X Y Z space. The next 9 are a 3x3 matrix of the rotations in the X Y Z axes (this is the hard part). Finally, 3001.dat is the part number for a 2x4 brick.

```
1 4 0 0 0 1 0 0 0 1 0 0 0 1 3001.dat
```

A LDraw model will be made up of hundreds or thousands of these parts, possibly broken into submodels. In turn, each part will be broken up into primitives, and each primitive will be made of lines, triangles, and quads. All of this put together forms your model. However, this language is unique to LDraw and cannot be read by POV-Ray or any other ray tracer. This means we need to convert it.

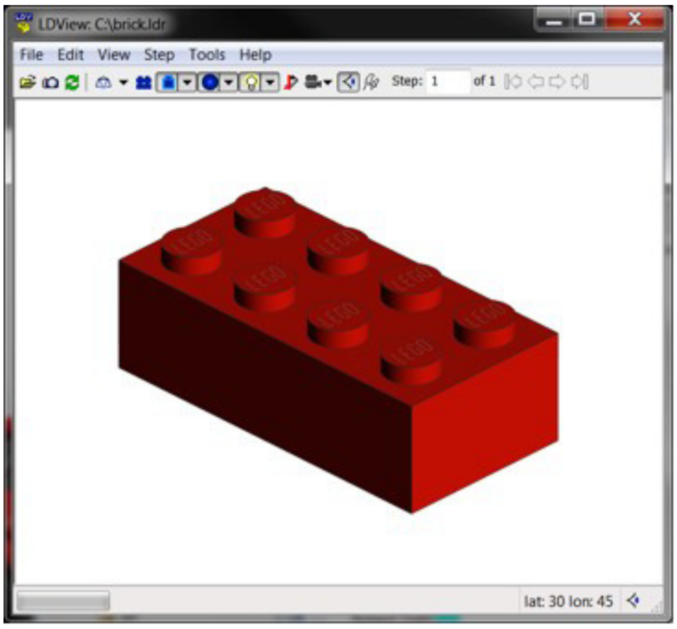
POV-Ray creates shapes using CSG (Constructive Solid Geometry). This method involves starting with simple solids such as cones, spheres, and boxes and then combining them with Boolean operations such as addition and intersection to create complex shapes. Obviously this is a lot different than LDraw so the process to convert from one to the other is not simple. Luckily for us, people have already solved this problem for us. Also luckily for us, POV-Ray places parts in space using the same sort of transformation matrix as LDraw and also consists of text files which call libraries of "include" files, so there are many things which will look familiar.

There have been a number of tools over the years which can perform the conversion from LDraw to POV-Ray, but my current favorite is LDView. It does an excellent job and will produce a good render without any further effort. With minimal additional editing of the POV-Ray file, the results can be improved even more. Finally, with advanced editing and changes to some of the core libraries, truly stunning results can be achieved.

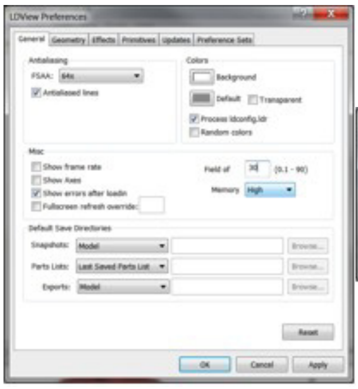
Let's work through an example from beginning to end using the simplest settings. We'll start with the model described above: a file containing a single red 2x4 brick. Create a file called brick.ldr that looks like this:

```
0 Brick Render
1 4 0 0 0 1 0 0 0 1 0 0 0 1 3001.dat
```

If you open this in LDView, you should see this:



When you export from LDView, it will also export the view information including the position of the camera. Let's make this a little more interesting by changing the amount of perspective and rotating the part a bit. Open the preferences panel and change the field of view to 30 degrees. Now bring up the view window with CTRL-9 to enter a latitude and longitude of 25 and 65. After that, your model should look like this.


LDView Preferences

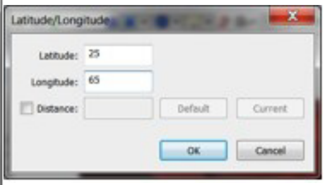
General | Geometry | Effects | Primitives | Updates | Preference Sets

Antialiasing
FSAA: 64x
[✓] Antialiased lines

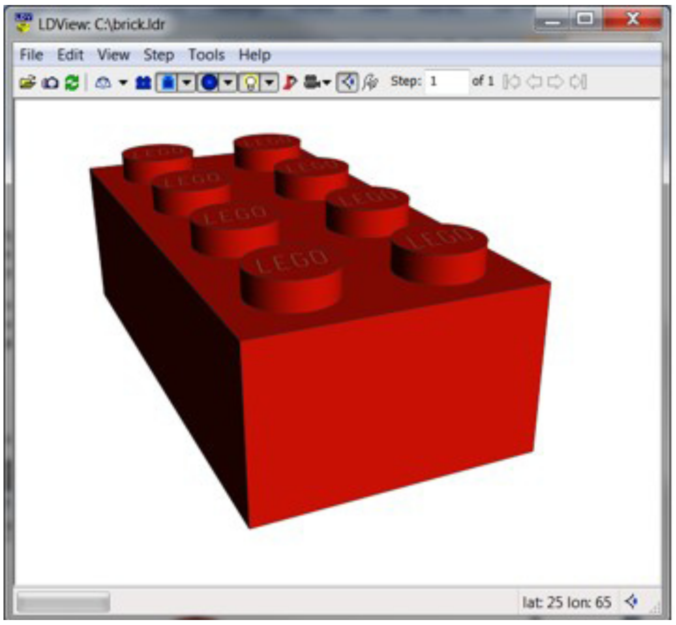
Colors
Background
Default [ ] Transparent
[✓] Process ldconfig.ldr
[ ] Random colors

Misc
[ ] Show frame rate
[ ] Show Axes
[✓] Show errors after loadin
[ ] Fullscreen refresh override:
Field of: 30 (0.1 - 90)
Memory: High

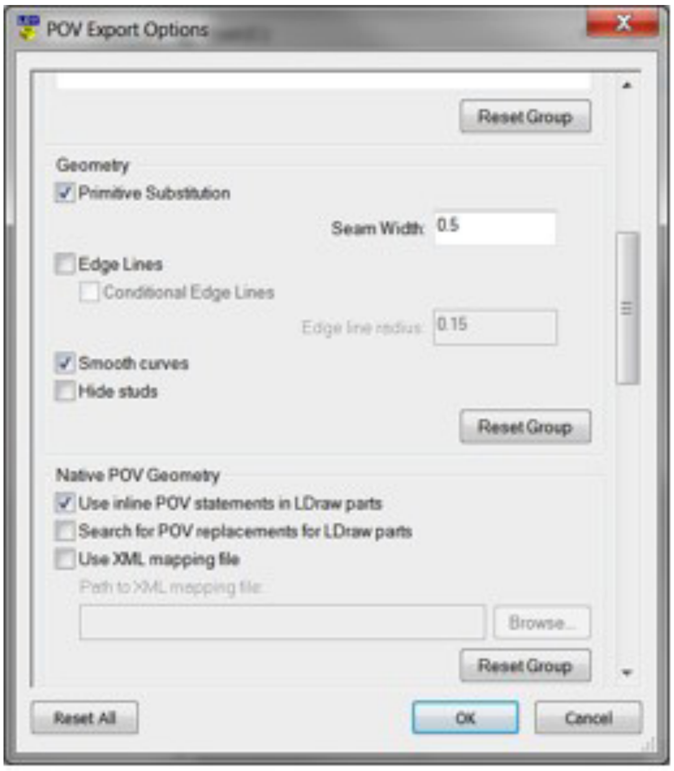
Default Save Directories
Snapshots: Model
Parts Lists: Last Saved Parts List
Exports: Model


Latitude/Longitude

Latitude: 25
Longitude: 65
[ ] Distance:



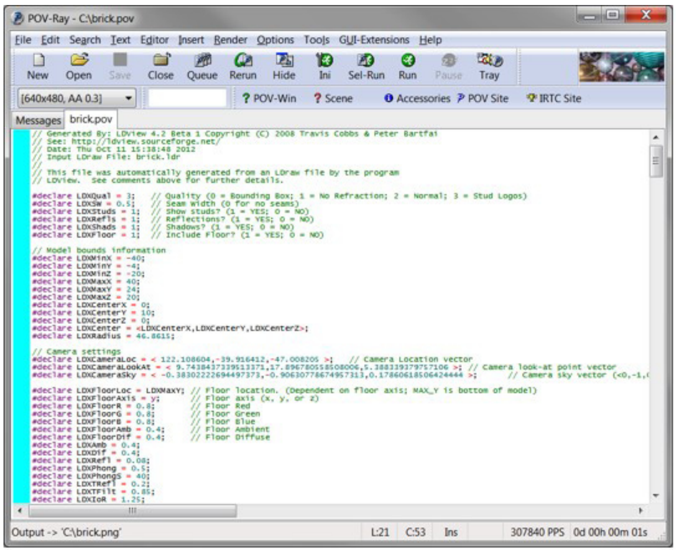
Now it is time to export to POV-Ray. You can either use CTRL-E or choose Export from the File menu. This will bring up a dialog box asking you to name the file. Let's just call it "brick". The extension ".pov" should be added automatically. The "Type" should be "POV-Ray Scene File". If you click the "Options" button, you will see a lot of things that can be configured about the POV conversion. We're going to start with all the defaults, except make sure you change "Quality" to "Include Stud Logos" and scroll down and deselect "Use XML Mapping File". Now you can select OK and save the file.


POV Export Options

Geometry
[✓] Primitive Substitution
Seam Width: 0.5
[ ] Edge Lines
[ ] Conditional Edge Lines
Edge line radius: 0.15
[✓] Smooth curves
[ ] Hide studs

Native POV Geometry
[✓] Use inline POV statements in LDraw parts
[ ] Search for POV replacements for LDraw parts
[ ] Use XML mapping file
Path to XML mapping file:

Now run POV-Ray 3.6. If you open "brick.pov", you should see this. It will look a little different on a Mac.


POV-Ray - C:\brick.pov

File Edit Search Text Editor Insert Render Options Tools GUI-Extensions Help
New Open Save Close Queue Rerun Hide Ini Sel-Run Run Pause Tray
[640x480, AA 0.3]
POV-Win Scene Accessories POV Site IRTC Site
Messages brick.pov

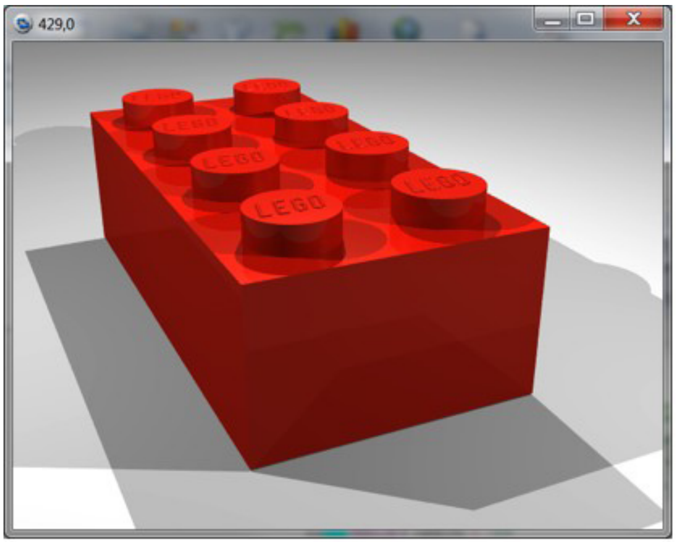
```
// Generated By: LDView 4.2 Beta 1 Copyright (C) 2008 Travis Cobbs & Peter Bartfai
// See: http://ldview.sourceforge.net/
// Date: Thu Oct 11 15:38:48 2012
// Input LDraw File: brick.ldr
//
// This file was automatically generated from an LDraw file by the program
// LDView.  See comments above for further details.

#declare LDXQual = 3;   // Quality (0 = Bounding Box; 1 = No Refraction; 2 = Normal; 3 = Stud Logos)
#declare LDXSW = 0.5;   // Seam Width (0 for no seams)
#declare LDXStuds = 1;  // Show studs? (1 = YES; 0 = NO)
#declare LDXRefls = 1;  // Reflections? (1 = YES; 0 = NO)
#declare LDXShads = 1;  // Shadows? (1 = YES; 0 = NO)
#declare LDXFloor = 1;  // Include Floor? (1 = YES; 0 = NO)

// Model bounds information
#declare LDXMinX = -40;
#declare LDXMinY = -4;
#declare LDXMinZ = -20;
#declare LDXMaxX = 40;
#declare LDXMaxY = 24;
#declare LDXMaxZ = 20;
#declare LDXCenterX = 0;
#declare LDXCenterY = 10;
#declare LDXCenterZ = 0;
#declare LDXCenter = <LDXCenterX,LDXCenterY,LDXCenterZ>;
#declare LDXRadius = 46.8615;

// Camera settings
#declare LDXCameraLoc = < 122.108604,-39.916412,-47.008205 >;   // Camera Location vector
#declare LDXCameraLookAt = < 9.743843733951337l,17.896780558508006,5.388339379757106 >; // Camera look-at point vector
#declare LDXCameraSky = < -0.38302222694497373,-0.90630778674957313,0.17860618506424444 >;      // Camera sky vector (<0,-1,

#declare LDXFloorLoc = LDXMaxY; // Floor location. (Dependent on floor axis; MAX_Y is bottom of model)
#declare LDXFloorAxis = y;      // Floor axis (x, y, or z)
#declare LDXFloorR = 0.8;       // Floor Red
#declare LDXFloorG = 0.8;       // Floor Green
#declare LDXFloorB = 0.8;       // Floor Blue
#declare LDXFloorAmb = 0.4;     // Floor Ambient
#declare LDXFloorDif = 0.4;     // Floor Diffuse
#declare LDXAmb = 0.4;
#declare LDXDif = 0.4;
#declare LDXRefl = 0.08;
#declare LDXPhong = 0.5;
#declare LDXPhongS = 40;
#declare LDXTRefl = 0.2;
#declare LDXTFilt = 0.85;
#declare LDXIoR = 1.25;
```

Output -> 'C:\brick.png'   L:21   C:53   Ins   307840 PPS   0d 00h 00m 01s

POV-Ray needs to know what size image you intend to make, and on Windows this is controlled in an ini file. Luckily, you only need to take care of this once and then POV-Ray will remember your settings. We want to try a render at 640x480 using anti-aliasing. To do this, select "Edit resolution ini file" from the "Tools" menu. A text editor with a file called "QUICKRES" will come up, and you can add this to the file:

```
[640x480, AA 0.3]
Width=640
Height=480
Antialias=On
Antialias_Threshold=0.3
Output_File_Type=N
Output_Alpha=On
```

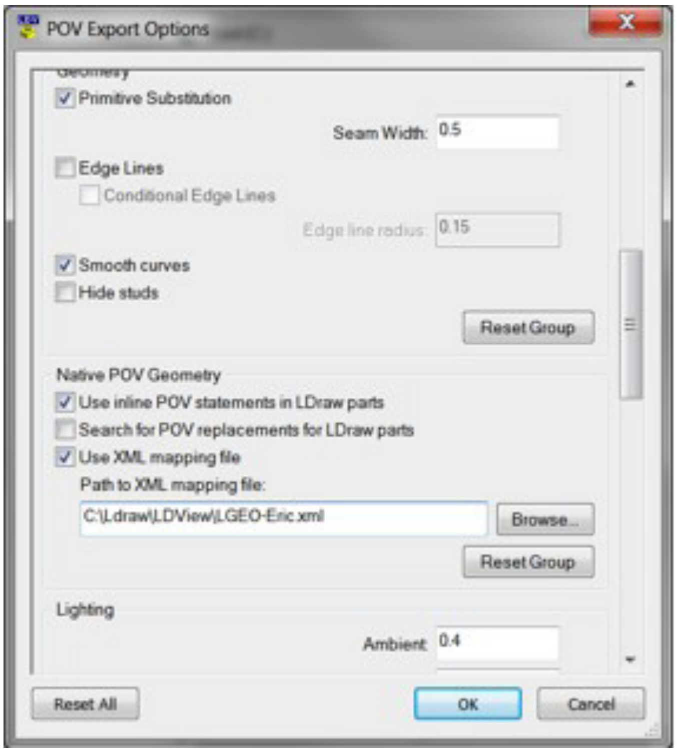
Save the file and quit the text editor. This will add an option called "640x480, AA 0.3" to a drop down menu that you can see at the top left of the previous image. (Note that you may have to quit and restart POV-Ray for the new option to appear.) It sets the height and width of the image, adjusts the anti-aliasing settings (which will smooth the edges), and outputs a PNG file. Don't worry about the details of the anti-aliasing settings; we'll just always leave them the same. Now all you have to do is hit the "Run" button, and a few seconds later you should have your first render.



This is a big improvement! But it has a long way to go. You'll probably notice that there are a lot of overlapping shadows. You will also notice that, since this is a direct conversion from LDraw, the edges are completely sharp and square. LDView was smart enough to make the studs look round for you instead of faceted and it also added the stud logos, but this still doesn't look like a real part. The next big improvement we can make is to use the LGEO library.

LGEO is a library of LEGO® parts made by Lutz Uhlmann using the native CSG language of POV-Ray. This means he wasn't limited to just triangles and quads, but could make almost any shape he wanted. He modeled the parts in much more detail including softly curved edges. The LGEO library of POV-Ray parts should also have been installed by the All-in-One installer. In order to use it, we need to make sure a couple of things are in place. Again, this is something you should only need to do once. POV-Ray needs to know where to find the LGEO library. Go to POV-Ray and select "Edit master POVRAY.ini" from the "Tools" menu. You'll see a file which has some paths at the bottom. Make sure that the All-in-One installer has added the path to the LGEO library here. It should look something like this.

If it is not there, you may need to add it manually (the actual location of this file depends on where you installed it on your system).
Library_Path="C:\Ldraw\LGEO\lg"
Remember when we exported from LDView and we deselected “Use XML mapping file”? Now we want to turn that back on. LDView comes with an XML file that maps each LDraw part to its matching LGEO part and performs scaling or rotations to make them line up. All you need to do is select the mapping file that came with LDView. It should be in your LDView directory. (Note that I have my own customized file which makes some changes and additions to the default file, but the default works fine.)


POV Export Options

Geometry

- [✓] Primitive Substitution
  - Seam Width: 0.5
- [ ] Edge Lines
  - [ ] Conditional Edge Lines
  - Edge line radius: 0.15
- [✓] Smooth curves
- [ ] Hide studs

Reset Group

Native POV Geometry

- [✓] Use inline POV statements in LDraw parts
- [ ] Search for POV replacements for LDraw parts
- [✓] Use XML mapping file
  - Path to XML mapping file: C:\Ldraw\LDView\LGEO-Eric.xml

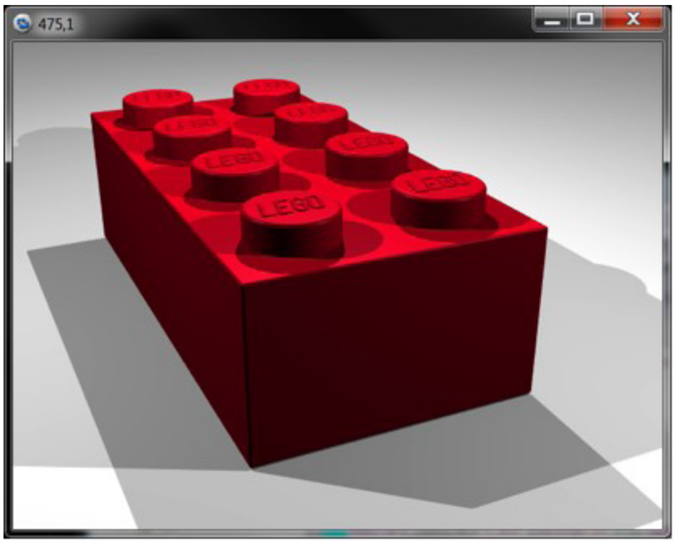
Browse...
Reset Group

Lighting

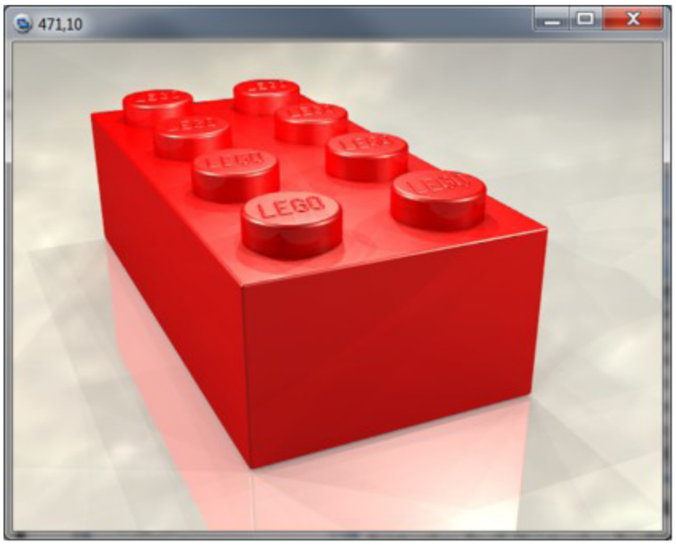
- Ambient: 0.4

Reset All | OK | Cancel

This time we’ll call the file “brick-lgeo”. If you have configured everything right, this file should work in POV-Ray without any further changes. Just open it and click “Run”. Now you should get this.



You can see the rounded edges, the different color, and the clearer logo. The LGEO library doesn’t contain every single part that the LDraw library does, so in practice when you export a large model some of the parts will be replaced with LGEO equivalents and some of them will not. Usually this is not a problem.
With the skills you’ve learned here, you should be able to make basic renders. Although this tutorial may have seemed complicated, once you get everything set up the process is very quick. I can create the sample file containing the brick, open it in LDView, export it, and render the image above all in less than 30 seconds. In future tutorials we’ll expand on what you’ve learned so that you can understand the content of the POV-Ray file and modify it to include more sophisticated light sources, use other features like radiosity and high dynamic range lighting, add backgrounds, and even create animations. After the next lesson, you should be able to replace the last image with this.



Eventually, you should be able to do this.

Happy rendering!
#