Virtual LEGO®
The key to amazing renderings

By Mattia Zamboni, brickpassion.com

I. Introduction

Let's be clear: nothing really beats playing with real bricks, no doubt about that. However there is something somewhat close to it which can be appealing as well.
About 10 years when I was still in my own dark age not thinking about LEGO, I happened to randomly stumble onto a software called MLCAD, the well-known program which allows users to virtually build with bricks on the PC. It didn't take me long to fall in love not only with the software but with LEGO again. Using this software was pretty convenient for me, since at that time I didn't have any of the bricks from my childhood anymore. What a dream to be able to build anything using any kind of bricks… in any color!
But it didn't last very long and soon I started to feel the need for something more realistic: that's when my long journey in the Computer Graphics (CG) world started.
After long tests and investigations I determined the best strategy for transferring models from virtual LEGO CAD software to a 3D CG software specialized in photorealistic rendering.
I then started to have so much fun that I was dedicating every single minute of my spare time to it. This passion got so serious that I started creating books about cool LEGO models. Today the most rewarding feeling is whenever a reviewer comments on the photography in my books without having realized that it's not photography at all, but all computer generated instead.

Over the years I have had several requests from people asking how I generate my renderings and whether I could share some of my secrets.
Well, the truth is that achieving a level of photorealism able to fool people is not as trivial as you might think, there's no two ways about it. In other words there is unfortunately no magic trick here: highly accurate photorealistic rendering is time consuming.

However, in this article I will do my best to explain the basic golden rules behind great CG renderings and I will describe the techniques I developed over the course of the years to generate the images for my books. In addition, I will mention a few of the newest tools which allow the achievement of decent results in an impressively short time.

II. The Golden Rules

So, what does it really take to generate highly photorealistic LEGO model images? Generally speaking I would say that great attention needs to be paid to details, because it is the details that eventually makes the final render look real.
But let's start with the 3 most fundamental key points you need to consider in the process of generating great renderings:

- The 3D model's accuracy
- The model's materials
- Scene illumination

Each of these is equally important. Just like with a group of climbers roped together, if one falls there is a chance that they all fall. Let's elaborate more on the above points.

A. The 3D model's accuracy

The world we live in is analog, and so are we. What we see is all analog at the source since our eyes are analog sensors. Computers, however, are digital. Information is represented with discrete quantized data using digital bits. This of course applies also to the 3D CAD environments. This entails for example that a circle in CAD software is not a perfect a circle, but a polygon with enough sides to make it look circular. The more sides/edges your object has the more "analog" it will look. So, if you want to generate a photorealistic render you have no choice other than to have an accurate 3D model, which means having it composed of a very high number of polygons. There is indeed no way to create a good picture if your model is not accurate and does not contain enough details. The following image illustrates just this concept.

Fig. 1. A low polygon pig on the left, a highly detailed one on the right. Which one is the most accurate?

In the world of bricks there are a range of sources available for virtual parts, but the tough reality is that the parts models in these libraries are not entirely accurate. This of course was done by design, in order to allow the handling of bigger assemblies, but it simply doesn't help in the rendering process.

B. The model's materials

The way we see the world around us with our eyes is the complex result of laws of physics affecting the light's path before and after it hits objects. There are several fundamental concepts like reflection, refraction, and diffraction just to mention a few. Every material behaves differently which results in an effect which our mind associates with **real**.

So, the second thing you really need are materials (in the CG world also called shaders) to assign to your model which generates realistic effects. If this is not done properly, your model will simply look dull and unrealistic.

Take a close look at the following figure:

Fig. 2. In the above scenes the models are identical, but one has more realistic materials applied. I'm sure you can tell which one.

Again, it's the reflections, refractions and all the real life physics phenomena which brings life to the models in your scene.

C. The scene illumination

What is the key factor to good photography? One word: **lighting**.
Just like in photography, lighting plays a fundamental role here, because depending on how your scene or subject is illuminated it can look appealing or, on the contrary, it can look flat.
Just consider for example how a picture of a landscape can change depending on the time of day the picture is taken.
Generally speaking, you want to create lighting such that it emphasizes the edges and eventually provides more volumetric information.
The next figure should help to clarify this:

Fig. 3. Which illumination provides the most interesting effect on Tomb Raider's Lara Croft?

Since we are trying to create photorealistic renderings, it is important to invest time in playing with lighting, as this factor contributes no less than the two previous ones to realism. The subject, the materials and the illumination: we want everything to look as real as possible.

With that being said let's take a look at what tools and solutions are available out there to render LEGO models.

III. The Available rendering solutions

There are several options available nowadays to generate realistic renderings. Here is a list of the most common:

1) **POV-Ray**: this has been the most popular and documented free rendering solution in the past. It isn't necessarily trivial to exploit its features and its render engine is not very fast.

2) **Bluerender**: currently this is one of the simplest choices out there. This free software can take your LDD files (LEGO Digital Designer) and generate renders in a less than 15 minutes. It unfortunately doesn't produce shine on the transparent parts.

3) **Mecabricks**: This is natively an online LEGO editor. Its ecosystem includes among its many features the possibility to generate decent renders.

4) **Custom process**: this solution essentially involves transferring your model from either LDD or an LDraw based editor to a powerful rendering software. There are free options like **Blender**, and many commercial (and sometimes expensive) products like **3D Studio Max**, **Cinema 4D**, and **Maya** just to mention a few. More recently, very simple software solutions dedicated to rendering have been introduced, with **Keyshot** being one of the most popular.

In my case I decided years ago to go for the last listed option. It is by far the one with the steepest learning curve, but if you like CG graphics, the invested time is well worth it. Last but not least, this option comes with an extra bonus: once you are able to create a nice render, you can start to animate it!

IV. My solution

The main reason for my choice is that it is really the only one offering no limits in what you can create, and the final rendering quality provided is limited only by your skills.
My personal favorite is 3DS Max and I am using it in combination with a great converter by Okino Computer Graphics to import models.

In order to generate hyper realistic renders, however, there are a few obstacles to overcome so as to comply with the above mentioned golden rules.
The most relevant is that the imported model is not very accurate (low polygons). As mentioned above, LEGO CAD editors use rather low-detail parts.
Although there are high quality parts libraries available (such as LGEO for POV-Ray), for my tool chain I couldn't use it.
In addition I wanted to add even more details and I therefore started to create my very own library by remodeling the bricks from scratch.

The next figure shows an example of a part:

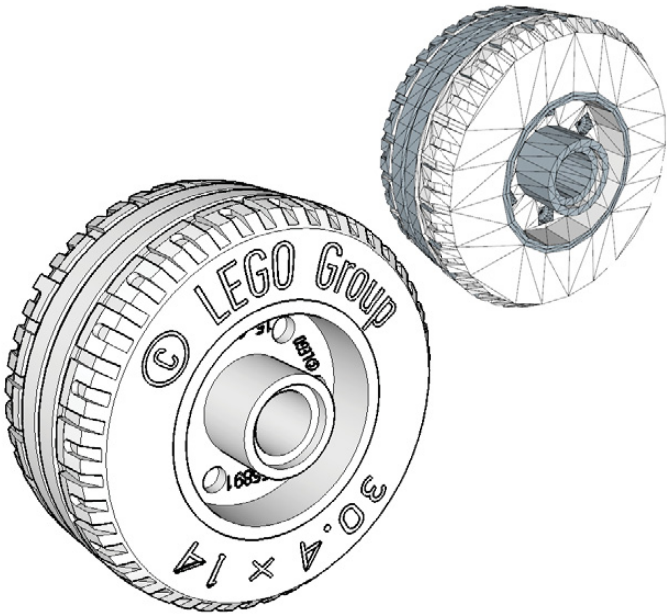

Fig. 4. Comparison between a part as imported (left) and the remodeled version (right) from my library. The level of detail I packed in includes tire text, part number and LEGO logo.

For some parts I decided to go crazy by adding even further details, which helped me in close-up shots to bring realism to the next level.

The next challenge was to replace the low accuracy parts with the highly detailed ones. This can of course be done manually, which by the way is what I did for all the models included in the "LEGO Build-it books: Amazing Vehicles". But for large models this can take ages. It was when I was contributing to the cool buildings of the "The LEGO neighborhood book" that I started to write a script to perform the automatic parts replacement which picks parts from my library.

Once this step is done we are ready with a very accurate model, so it is now time to dress the model with the proper materials.

To achieve this goal you need shaders which simulate the materials used by LEGO. Luckily the amount of these is limited: it is of course for the vast majority the typical ABS plastic (in several different colors). Then the transparent plastic (again in several colors), the rubber for tires, and some special material to give silver/golden/chrome effect to the painted parts. I had to spend quite some time tweaking the materials' parameters in order to get realistic results, but this is a one time job and once finished you have your own LEGO materials library.

A trick I found was to use a visual setting in the materials parameters which shaves off a tiny bit of the edges of the bricks only at rendering time. This results in a very realistic look, compared to the native sharp edge defined in the 3D elements. It also allows me to keep the models simpler.
On top of this let's not forget about stickers or painted parts. This is a whole separate job, in which you need to digitalize the artworks by scanning and importing them as bitmap textures into the software to be applied to parts.

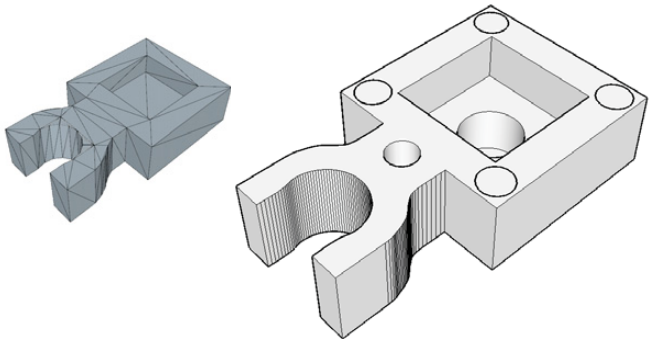

Fig. 5. Example: in the "Plate 1x1 with vertical clip" I decided to include the molding marks on the bottom side. A scrupulous eye can notice these details in close up shots.

For the parts replacement, I also wrote a script to perform an automatic material assignment based on the original color of the imported model. To help in this process I included a few smart features which for example recognize specific parts like tires by their part number and automatically assigns the black rubber material.

Once materials are properly assigned it is time to have fun by preparing the scene and lighting. This setup is no less important than preparing a stage for a photoshoot.
This involves placing a background and a set of lights to illuminate your model. As fun as it is, this part can be one of the most time consuming. You want to make sure the lighting on the model is sufficient to create a pleasant shadow and provide good reflections which are not too strong.

To save time at this stage I again automated the process, which in this case uses a brute force method which rotates the lights all around the subject and generates several previews. This process can take a while but at least it can proceed unassisted, allowing me to just play the role of artistic director. I just have to look at all the previews and choose the one I like best, and if needed perform some fine-tuning on it.

Fig. 6. Example of rendering from my last book. Take a look at the soft pleasant shadows, generated in this case by a natural lighting (HDRI ambient image).

After the final rendering has been generated there is the post processing step in which I enhance the picture by working on brightness, contrast, and colors, and I apply some additional filters to make the image look the best possible. This step is really needed since it is not possible to have the rendered images look perfect right out of the rendering engine. During this phase it is suitable to use a color calibrated monitor.

At this point the model is highly detailed, it has realistic materials applied to it and a good lighting setup. Everything looks perfect, but… even too much! In fact this sometimes turns out to be a problem with renderings: they look too perfect which eventually makes them look fake. I therefore developed one of my favorite functions to add to my toolset, which I named RealWorld™.

RealWorld™ is an advanced piece of software which analyzes your 3D LEGO model and automatically introduces realistic imperfections in the way the model is assembled.

This is a list of just a few features of RealWorld™ :

- **Logo on studs**: all the bricks get rotated to make sure the logos are not oriented in the same direction (especially 1x1 round plates!)
- **Creation of seams**: all the bricks gets scaled by a random factor very close to 1 (example 0.996), which shrinks them ever so slightly generating the typical seams/gaps between bricks and it is performed so that they are not all identical.
- **Parts rotation**: Bricks which have a small play while assembled (for example 1x1 bricks/plates/tiles) gets rotated by a random tiny fraction of angle.
- **Parts tilt and lift**: all bricks which don't have other bricks placed on top of them (especially tiles) gets lifted and tilted around their x and y axis by a very tiny random amount. On surfaces (i.e. a sidewalk) this creates a nice non uniform assembly effect.
- and more …

"Details make perfection,
and perfection is not a detail."

\- Leonardo Da Vinci

Once I developed these techniques to process models and generate renderings automatically, I decided to raise the bar further. My dream was indeed not to limit all this to a simple model, but to extend it to entire dioramas.
This turned out to be a major task since I had to revise all the software written up to that point and optimize it thoroughly by paying special attention to speed and memory allocation. Importing and processing 500 bricks is indeed not the same as with 100,000 bricks.

Fig. 7. One of the most appreciated renderings in my latest book was rendered at 5125 x 3375 pixels and took about 16 hours to render.

The additional required step has been to get my hands on a very powerful workstation able to process all that data in reasonable time. I ended up building a custom station with the following specs (for the nerds among you ;-)):

2x CPU E5-2640 Xeon CPU
8 cores/16 threads
64GB DDR4 RAM
512GB SSD storage
NVIDIA GTX Titan X with
12GB VRAM

A due note is that generating highly realistic renders can require a lot of CPU power, mostly when generated at very high resolutions, with accurate shaders and with DoF effects. The next image shows an example of rendered diorama from my book "Tiny LEGO Wonders" designed by Alexander Bugiel. I must admit that at this point I am quite happy with my current results even if there is definitely still room for improvement. The content of this article is the result of more than 5 years of CG techniques and scripts development. However, the biggest challenge has been to explain and summarize all of this in a few pages, and I hope I was at least able to transmit some of my passion about this fantastic world!

Mattia Zamboni
www.brickpassion.com

You can see some of the fabulous renders made by Mattia Zamboni in the following Nostarch titles:

The LEGO Build-It Book I / The LEGO Build-It Book II / The LEGO Neighborhood Book / Tiny LEGO Wonders
#