

# XLE Getting started exporting

---

Max'ing it up a little bit

# Exporting from Max

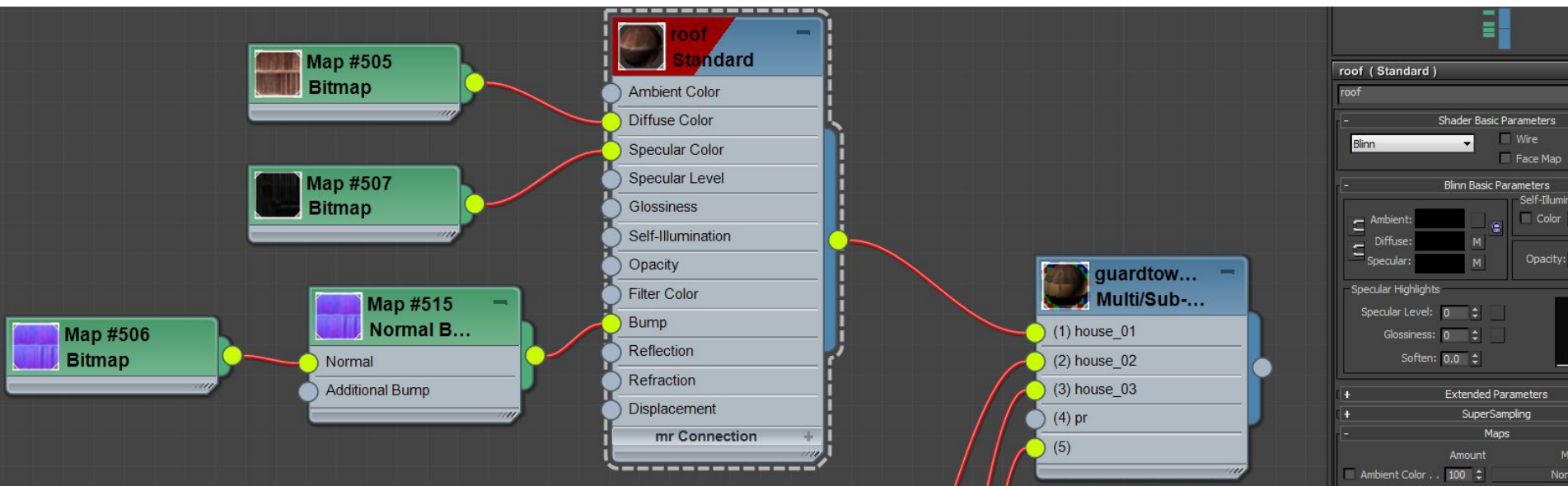
---

# Exporting steps

1. Setup textures and material settings using Standard materials
2. Export using “OpenCOLLADA”
3. Preview export in MaterialTool

# “Standard” material with “Blinn”

Most of the time, we can use a “*Standard*” material with “*Blinn*” lighting



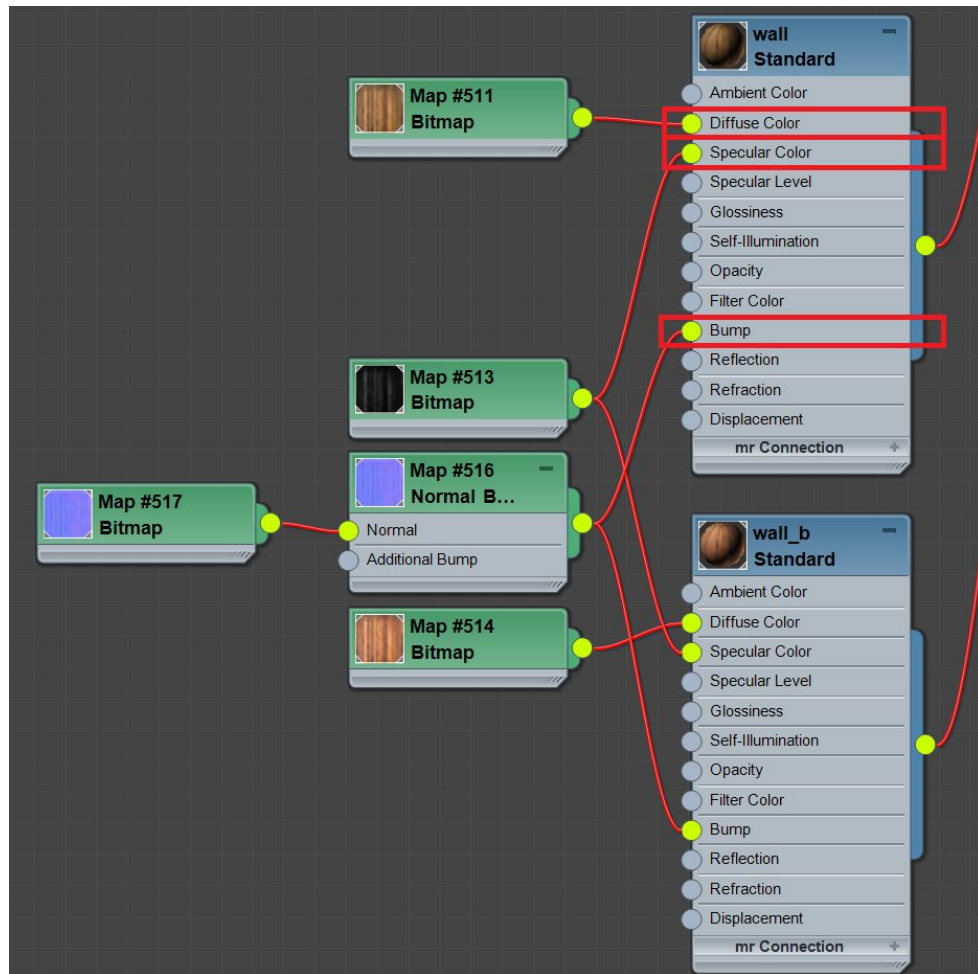
# Textures

Attach textures to:

1. Diffuse
2. Specular Color
3. Bump

The other ones don't do anything.

*(Opacity must be in diffuse alpha channel)*

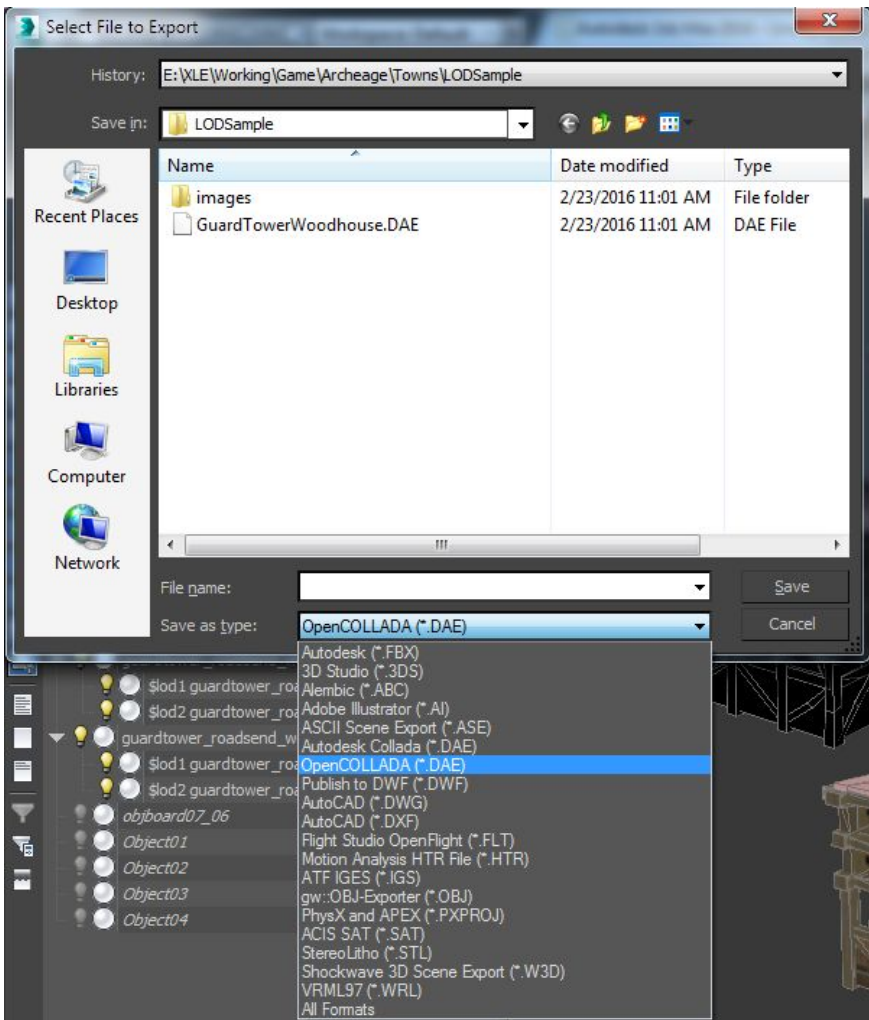


# Install OpenCOLLADA exporter

OpenCOLLADA can be gotten from the KhronosGroup Github page:

<https://github.com/KhronosGroup/OpenCOLLADA/wiki/OpenCOLLADA-Tools>

Alternatively, you can download the source code and compile it yourself. For large projects, you may find it useful to have a custom built exporter (so that you can make modifications to the exporter as necessary).



# OpenCOLLADA

Export the scene using “**OpenCOLLADA**”

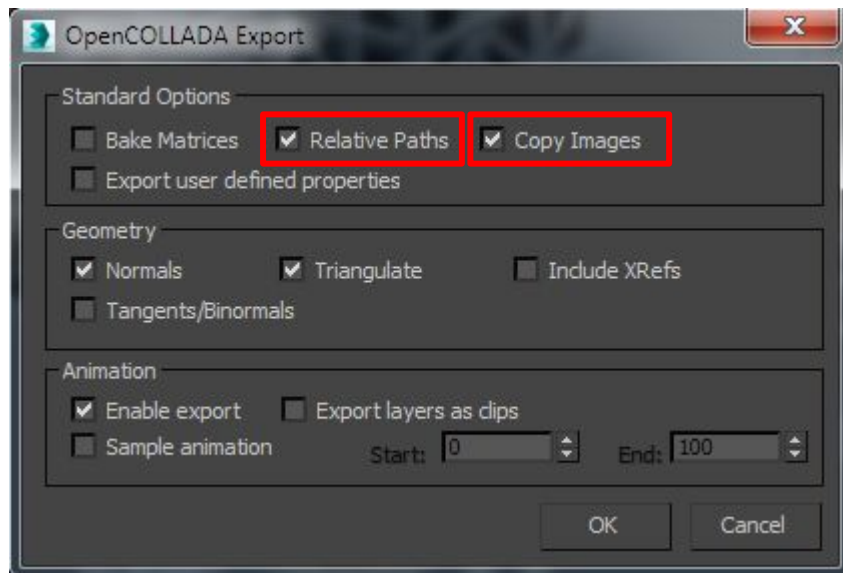
Don't use “*Autodesk Collada*”! Autodesk Collada will sometimes work, but it sometimes produces non standard Collada files. OpenCOLLADA has been tested with many models with XLE.

# Default options usually good

- Check **Relative Paths** and **Copy Images**

## *Technical:*

- If “*Tangents/Binormals*” is unchecked, XLE will generate tangents
- XLE algorithm seems to be identical to Max, so usually doesn't matter





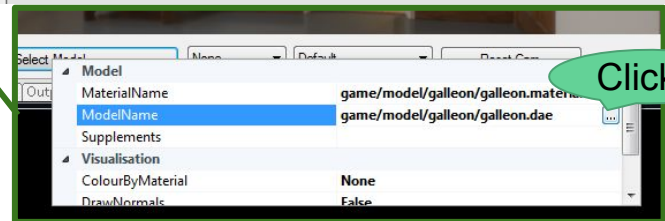
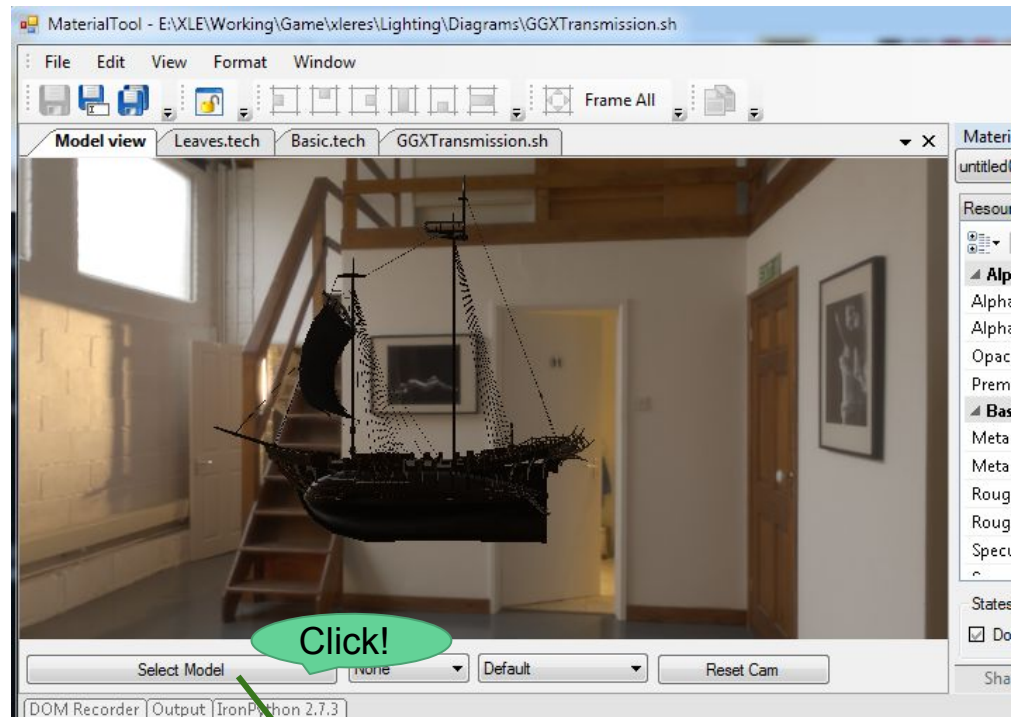
# .dae Collada file

The Collada exporter will generate a **.dae** file. This is an xml file with a text based representation of the scene.

# View in MaterialTool

We can use the MaterialTool to preview the model in XLE.

MaterialTool can be used to customize the material... But normally, this isn't required.



# Model cleanup

---

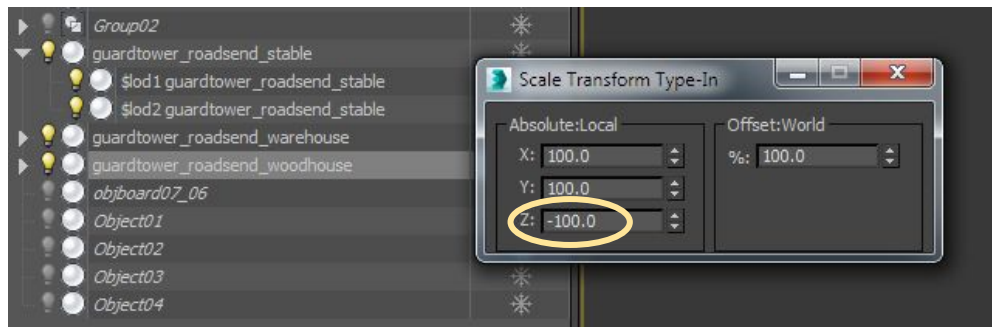
# Model cleanup

Some models might require additional changes.

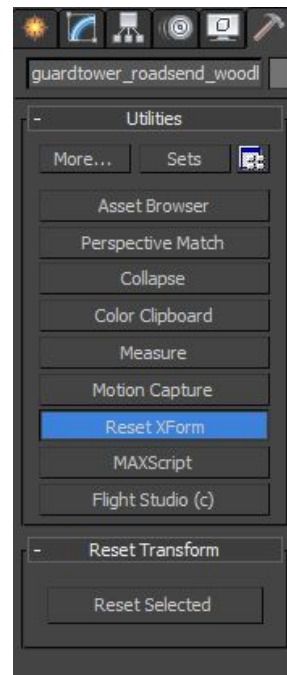
Here are a few common problems...

# Remove mirrors

Some models have a “mirror” in the transform. This happens when there is an odd number of *negative* scale values.



Use **Reset XForm**  
to remove the  
mirror.



- we can add support for mirrored transforms to XLE, if needed (just ask)

# Model units

The LevelEditor assumes a scaling of 1 unit per 1 meter.

But model files can use any units. In the .dae file, there is a configuration element:

```
<unit name="centimeter" meter="0.01"/>
```

The correct units must be specified in the Max file so that this is correctly initialized to the right value.

# LOD naming convention

LOD names should be named like this:

```
“$lod<number> some_name”
```

“\$” should be the first character. And the number should be followed by a space.

LOD root nodes can appear anywhere in the hierarchy

# Advanced usage

---



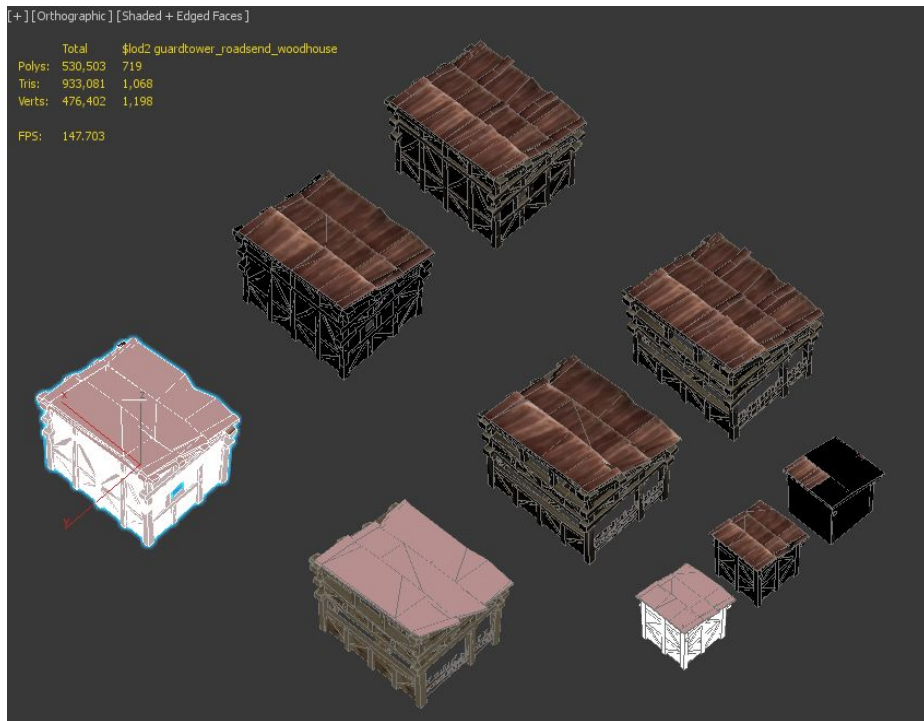
# Arranging exports

Ideally we want 1:1 relationship between *.max* file and *.dae* files

Max files often have multiple models with multiple LODs. XLE can also support multiple models in a single export.

Imagine a scene, *GenericTree.max*, that contains 10 different trees. We can export all trees with a single export (particularly handy if trees share textures).

Big projects should consider keeping ***dae files*** and ***max files*** **1:1**... This will make managing thousands of exports easier.



# Default environment/lighting config

*Working/defaultenv.txt* contains settings that describe the default environment settings used in the MaterialTool

- background texture, IBL lighting, directional lights, tonemapping, shadows, etc...
- you can export it from the LevelEditor (use the ***File/Export to Game*** functionality)

# XLE materials are ***Hierarchical***

This is important, but a little more advanced.

Imagine we had a game with many tree models. We can create a single “***Library/Nature/nature.common***” material settings file that is shared by many models.

This contains material settings that are shared by all objects with ***leaves*** and ***foliage***. Setting up smart material inheritance will make managing many models easier!

# XLE materials are ***Hierarchical***

Think about how different objects share settings...

eg:

- “*flags with wind animation*” share a material settings file
- “*anisotropic hair*” share a material settings file
- “*chrome highlights*” share a material settings file
- ...etc

Smart usage will save time later!

# Intermediate files

XLE caches some intermediate files in the directory “int”















This is for compiled shaders and compiled models

- “int/log\_\*.txt” -- log files
- “int/screenshots” -- output from “cv.Screenshot”
- “int/u” -- *shadowing* folder (more later)
- “int/d64\_0”, “int/r64\_0” -- cache folder

# Intermediate cache folders

- Intermediate cache folders are categorised *per configuration* and *per vertex number*.
  - 2 different versions of XLE can be run on the same PC. They will use separate cache folders to avoid conflicts.
  - Useful when comparing this week's version to last weeks' version

Cache folders are locked by each app instance. If you open 2 XLE apps at the same time, they will each use different folders.

Name	Date modified
 log_MaterialTool.txt	2/23/2016 1:12 PM
 log_LevelEditor.txt	2/22/2016 8:12 PM
 log_environmentsample.txt	2/17/2016 3:12 PM
 log_XLEApp.txt	2/2/2016 11:25 AM
 log_GraphNodes.txt	1/18/2016 12:23 PM
 log_BasicGUI.txt	1/13/2016 6:53 PM
 log_testplatform.txt	12/29/2015 4:04 PM
 log_unittest.txt	12/18/2015 5:00 PM
 log_texturetransform.txt	11/12/2015 5:12 PM
 log_IterativeSystemDebugger.txt	10/15/2015 4:46 PM
 TextureFormatCache.dat	8/21/2015 4:26 PM
 r64_0	2/23/2016 1:08 PM
 screenshots	2/5/2016 6:45 PM
 u	11/24/2015 2:49 PM

# Model cached data

Typical cached files for a model:

- guardtowerwoodhouse.dae-skin
  - binary model data (*file size == GPU in-memory size*)
- guardtowerwoodhouse.dae-skin-metrics
  - debugging and profiling data
  - human readable (*useful for finding problems*)
- guardtowerwoodhouse.material-guardtowerwoodhouse.dae-resmat
  - binary material data
- guardtowerwoodhouse.dae-rawmat
  - material data extracted from Collada file
  - human readable

# skin-metrics

This is a breakdown of how the model appears once it's loaded into memory. You can see how XLE interprets a model file.

- Geometry objects
  - source data for vertex buffers, index buffers, input assembly and draw calls
- Command stream
  - stream of “geo calls” and “skin calls”. These are events that invoke a draw of a geometry object
- Transformation stream
  - flattened and optimised representation of the node hierarchy
  - minimizes the number of CombineTransform operations while still allowing for animation



# Reload from disk

All asset files (models, textures, placements, shaders, material files, game object files, etc) reload from disk automatically when changed.

Try exporting from Max, or changing a texture in Photoshop with an XLE app open. The asset will update in XLE immediately.

Programmers can apply this behaviour to any type of data file easily.

# OpenCOLLADA changes texture names

The OpenCOLLADA exporter changes the texture names by adding  $N_*$  in front of the name.

This might not be ideal for all projects. An option is to use an alternative path for exporting textures (textures don't have to be exported from Max, they can be assigned in the material tool). Alternatively, modify the exporter to change texture naming behaviour.