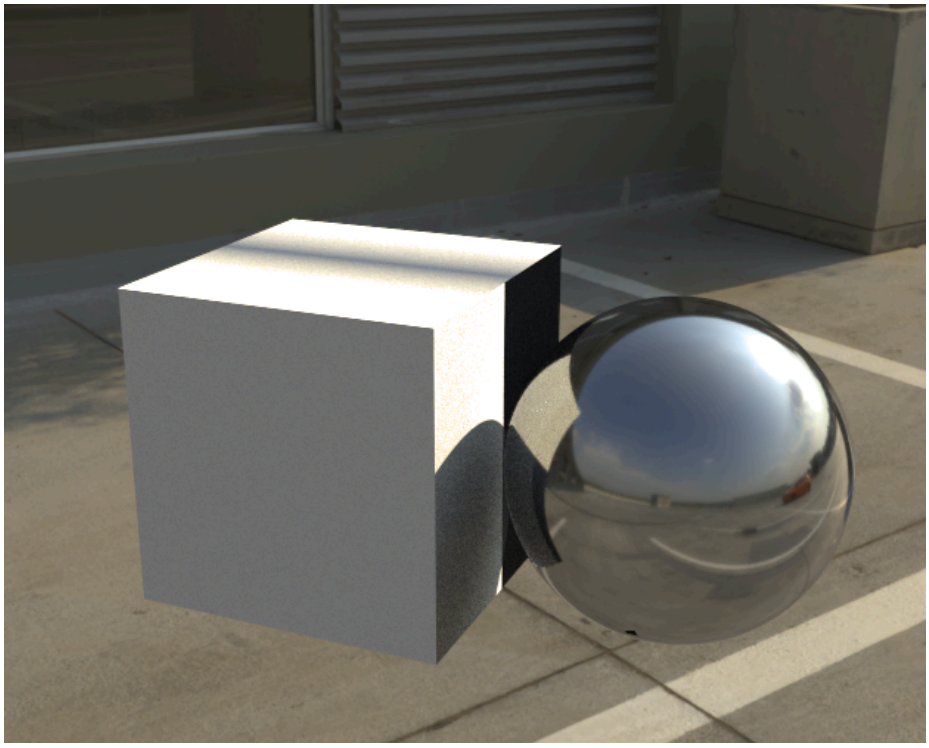
Rivaliant/VirgoRival Reality Bug documentation

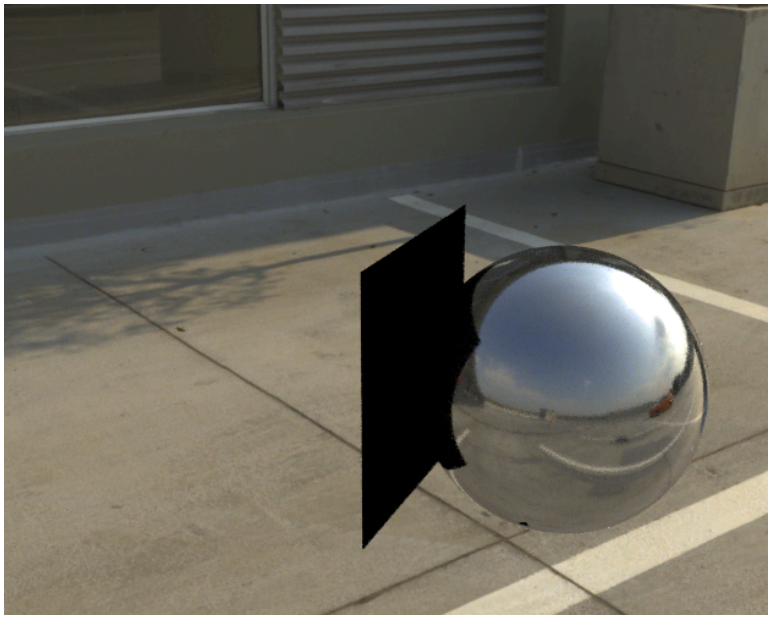
Reality not exporting properly
A long standing issue I have once pointed out
Without Reality 2.5 for comparison, I can't prove that Reality 2.5 was exporting the scene from Daz "correctly" or more correctly then Reality 4.5 is now.

Upon export, Reality doesn't seem to triangulate the geometry fully and is missing vertices or something along those lines
To recreate, just load a primitive Cube, that's it

Which results in this



(Chrome sphere to show the sun in the IBL to prove nothing is making the black band on the cube's side
And shadowy band on the top)



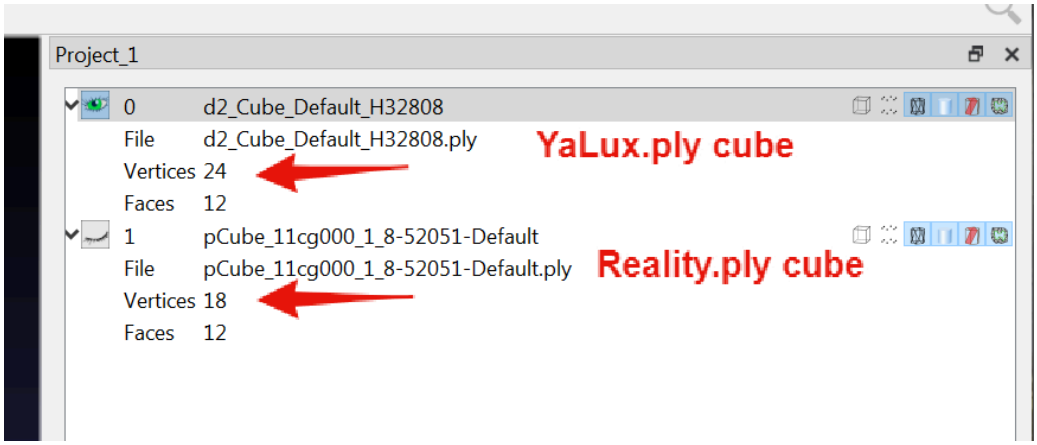
Setting the cube to Null
Reveals the a full black face on that side
When it should had been 'nulled' out

When Set “Geometry Format” to “Lux Text” in the output tab

It writes out the all the geometry information as a string of text inside the .LXI file
Which appears as so

```
# Mat Default (Null). 12 polys
AttributeBegin
NamedMaterial "pCube_11cg000_1_8-52051:Default"
Shape "mesh" "string name" ["pCube(11cg000)1_8-52051"]
"integer triindices" [
0 1 2
0 2 3
4 5 6
4 6 7
8 9 10
8 10 11
1 4 7
1 7 2
3 2 12
3 12 13
14 15 16
14 16 17
]
"point P" [
-0.897053 -0.500000 0.000000
0.102947 -0.500000 0.000000
0.102947 -0.500000 1.000000
-0.897053 -0.500000 1.000000
0.102947 0.500000 0.000000
-0.897053 0.500000 0.000000
-0.897053 0.500000 1.000000
0.102947 0.500000 1.000000
-0.897053 0.500000 0.000000
-0.897053 -0.500000 0.000000
-0.897053 -0.500000 1.000000
-0.897053 0.500000 1.000000
0.102947 0.500000 1.000000
-0.897053 0.500000 1.000000
-0.897053 0.500000 0.000000
0.102947 0.500000 0.000000
0.102947 -0.500000 0.000000
-0.897053 -0.500000 0.000000
]
"normal N" [
0.000000 -0.998311 0.000000
0.000000 -0.998311 0.000000
0.000000 -0.998311 0.000000
0.000000 -0.998311 0.000000
```

0.000000 0.998311 0.000000
0.000000 0.998311 0.000000
0.000000 0.998311 0.000000
0.000000 0.998311 0.000000
-0.998311 0.000000 0.000000
-0.998311 0.000000 0.000000
-0.998311 0.000000 0.000000
-0.998311 0.000000 0.000000
0.000000 0.000000 0.998311 ← (note here on the normals, Every set has 4 normals going the same
0.000000 0.000000 0.998311 ← way Except this pair here, two are missing from this face)
0.000000 0.000000 -0.998311
0.000000 0.000000 -0.998311
0.000000 0.000000 -0.998311
0.000000 0.000000 -0.998311
]
"float uv" [
0.00000000 0.00000000
0.33000001 0.00000000
0.33000001 0.50000000
0.00000000 0.50000000
0.66000003 0.00000000
0.99000001 0.00000000
0.99000001 0.50000000
0.66000003 0.50000000
0.66000003 0.50000000
0.99000001 0.50000000
0.99000001 1.00000000
0.66000003 1.00000000
0.33000001 1.00000000
0.00000000 1.00000000
0.33000001 0.50000000
0.66000003 0.50000000
0.66000003 1.00000000
0.33000001 1.00000000
]
AttributeEnd



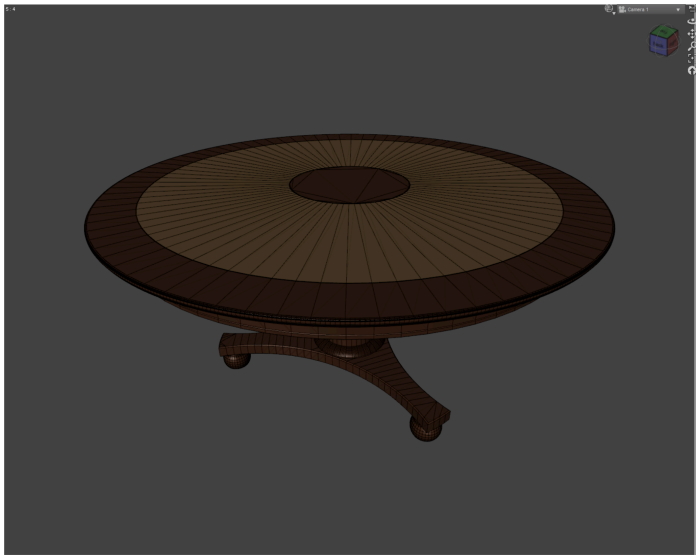
I just remembered YaLuxPlugin, and manage to pull the .ply File from its export and found this using MeshLab
The Reality Export is straight up missing 6 whole vertices Along then with their normals and other attributes

Viewing the wireframe between the two
It seems Reality Triangulates the faces While YaLux does not? Or does what Daz does and its There, but not a

Hard triangle poly but has a “triangulation” line since it still has 12 polys (2 triangles per face, 6 faces, 12 polys/faces)

For a more complex object that this issues effects

The Table from the Suite 2101 by Hole (looks like its no longer listed on Daz3d)

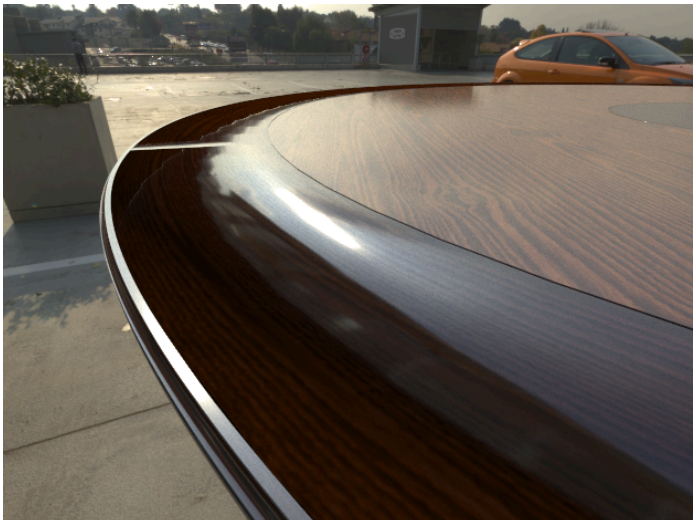


Viewport view
easy simple, round flat top table

All materials set to 9900 (99%) Gloss

Weird bubbled center with a target shape and edging unable to reflect IBL lighting, Legs perfectly fine

(the outer_edge surface appears to be the one with the issue on export)



This error makes the prop appear to have a concave curve when going towards the edge line
Indicating the normal is not pointing straight up at the edges
(or in the evidence shown above, missing vertices as a whole)

The solution I found for this issue is to use Micro-facets.

With a dummy Flat Gray texture and set its strength to 0% (as Micro-facets need a texture to function in Lux)
And set the subdivision to 10-50

This will create enough vertices to push the error into out of sight
(the gaps in the table near the edging is in the geometry, It isn't perfectly sealed)
This however doesn't always work for some props
And generally increases memory uses and rendering times as more geometry creates more need for calculations it seems

I hope this formatting is sufficient for my explantation

And hope i didn't butcher any basic 3DGC terminology to get my explanation across O n O;;;;

Thank you

-Rivaliant

(oh and also, using an IES file on a spotlight make its it come out of the spotlight in the Y+ direction rather than the direction the light is pointing)

Updates: Dec. 5th 2021

danielbui78

Thanks again for the detailed observations in your Google Doc. Here are my thoughts:

Regarding the vertex differences between Reality and the yalux plugin, you are correct: Reality does some "optimization" while yaluxplug just outputs almost exactly what Daz gives it (except for splitting all quads into triangles). Just like Paolo mentioned previously, Reality is actually abstracted from the LuxRender program and is designed to work with potentially any renderer. My main development machine is tied up building UE4 source code for the next 10-20 hours, but from what I remember, Reality pools all the mesh data together and runs optimizes the data to remove duplicate information. This is probably why there are fewer vertices than with the yalux plugin. However, I don't know why it didn't reduce the vertices

further -- maybe the UV or normal data was unique? I'll look at the normal and UV data that you posted when I get more free time.

However, my first impression when looking at the render output is that it looks very similar to rendering artifacts caused by bugs in some OpenCL drivers. Try rendering in pure software (native CPU) mode instead of OpenCL GPU/CPU. If the artifacts disappear, then it could be an incompatibility with how LuxRender is translating the scene data into OpenCL shader code. And if that's the case, then you could try changing to a different LuxRender version (you could also try changing to a different version of your GPU drives, but in my opinion, changing to older GPU drivers to fix artifacts is not a longterm solution).

Hopefully some of this info is helpful and enlightening.

Rivaliant: I have DM’s them back and confirmed while the the change in rendering modes seem to fix the Cube, it did not however fix the table

So it doesn’t seem to be the rendering mode and/or OpenCL and might be this Geometry Optimization they are speaking about.
More as this story develops O 3 O