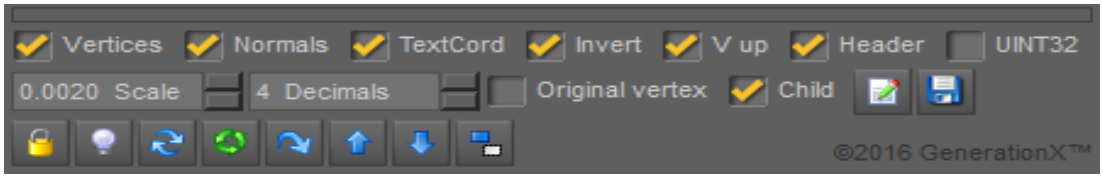
DirectX Mesh Plugin For Daz3D Studio 4.9



About
Use by game's programmers for creating 3D mesh objects using a DirectX mesh format.

Key Features

- Automatically converts quad to triangles face's.
- Vertex compression.
- Mesh normals pre recalculated.
- 16bit or 32bit Index's
- Child objects can be included with the mesh.
- Invisible object wont be added to the mesh.
- Setup your 3D objects faster than any other 3D file format, faster start-ups.
- DirectX 11 C++ text outputs information given when saved.
- Write button for c++ text output for small meshes.
- Save button for mesh data.
- Vertex data rotation buttons for 3D imports that lay flat or upside down.
- Scaling Spin-Box, Vert*=0.002; is the default, which will decrease the size.
- Decimal place's Spin-Box, 4 is the default, 0.1234f

To Use Plugin
Select an object in the scene that you want to convert, if its a small object and you just want a text file to be compiled by Visual Studio press the text output button, copy the text and insert to you document, large object should be saved as raw data. The order is Vertex(Verts,Norm,UV) then Index

To Unlock Plugin
Click the yellow padlock. Buy key-code using your pay-pal-account, after you have paid you will be given a key-code-number.
Copy and paste this number in to the text-box and click Unlock-DirectX plug-in.
If you change your computer at later date you can recover you key-code-number using paypal-email and purchase-date.

Tips
There is no file format to worry about, just setup a pointer to the correct buffer location.
To copy the text, just highlight the text and right click mouse button and select copy from menu.

Note
At startup this plugin is switch off, click the lightblub to activate the plugin.
Writing 3D to text can take up to 10 sec to 10 min and should only be used with less complex models. Updates will continue on the DirectX plug-in. Some knowledge of C++, Direct 11 and HLSL pixel and vertex shaders is required to use.

Trouble Shoot Texture Maps U Vs

Note: The resolution level should be set to Base and the Subdivision set to 0. If you don't set the parameters then your texture mapping will be wrong. Use texture Atlas to compact the main & child object in to one texture and make invisible the parts you don't see such as feet.

C++ Programming Information:

```
//Finding the index buffer location is easy.
int cube_VertexCount = 24;
int cube_VertexByteSize = 480; //<---= Info given when you save the file
int cube_IndexCount = 36;  //<---=36 short index's
int cube_IndexByteSize = 72; //<---= 36*sizeof(short) = 72

D3D11_SUBRESOURCE_DATA vertexBufferData={0};
D3D11_SUBRESOURCE_DATA indexBufferData={0};
vertexBufferData.pSysMem = OpenData; //<---= Put Memory location of the .dat file here
indexBufferData.pSysMem = OpenData + cube_VertexByteSize;

//Your vertex and index buffer our ready to be created...

D3D11_BUFFER_DESC vertexBufferDesc = { 0 };
vertexBufferDesc.ByteWidth = cube_VertexByteSize;
vertexBufferDesc.Usage = D3D11_USAGE_DEFAULT;
vertexBufferDesc.BindFlags = D3D11_BIND_VERTEX_BUFFER;
vertexBufferDesc.CPUAccessFlags = 0;
vertexBufferDesc.MiscFlags = 0;
vertexBufferDesc.StructureByteStride = 0;
DX::ThrowIfFailed(m_deviceResources->GetD3DDevice()->CreateBuffer(&vertexBufferDesc,
&vertexBufferData, &m_vertexBuffer));

//Index Buffer
CD3D11_BUFFER_DESC indexBufferDesc(cube_IndexByteSize, D3D11_BIND_INDEX_BUFFER);
DX::ThrowIfFailed(m_deviceResources->GetD3DDevice()->CreateBuffer(&indexBufferDesc,
&indexBufferData, &m_indexBuffer));
//Store index's
indexCountList[Indexs] = cube_IndexCount;
Indexs++; //next index's
```