Daz LOD System (v1.4)
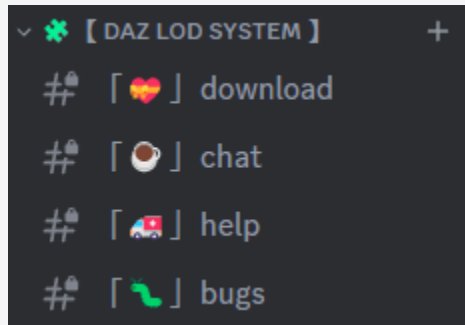
DOCUMENTATION

keyclap | Gumroad | 2025

Thank you for buying this product!

Don't forget to join keyclap.dev (https://discord.com/invite/z3qdxYhrB6)
Discord server to receive updates and send feedback.

Temporary difficulties!

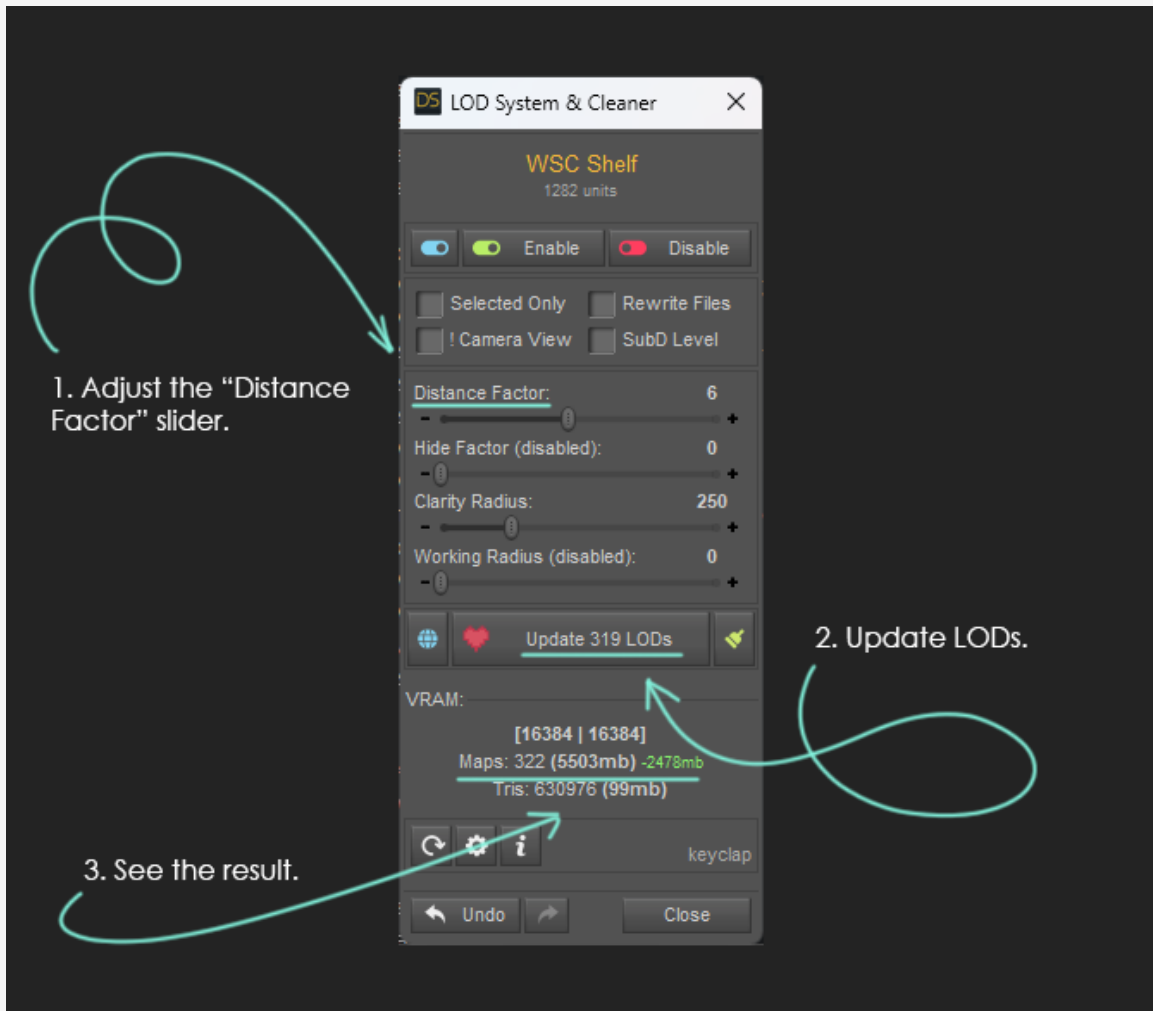
For now, to get the role and gain access to the chats above you have to send your payment
receipt from RenderHub or Gumroad to me (LaUwUrence).

Installation

Put the contents of the "Scripts" folder anywhere you want. The best place would be your
own "Scripts" folder in your Daz library.

Quick Start

1. **Adjust the "Distance Factor" slider:**
   The lower the value, the smaller textures will be created and applied.

2. **Update LODs:**
   Start collecting data of visible in scene and in render objects, send it to the Python site, create images in lower resolution right next to the original ones, send the changes back to Daz and apply the created textures.

3. **See the result:**
   The VRAM usage is calculated automatically on each LODs update. The first time takes longer than usual.

Complete Guide

All the buttons, sliders, checkboxes and texts have tooltips. Hover over them for a second to see their description.

Some Info:

In this guide I will use "LODs" (Levels of Detail) as a designation of a group of textures with the LOD prefix and "LOD" as a single texture.
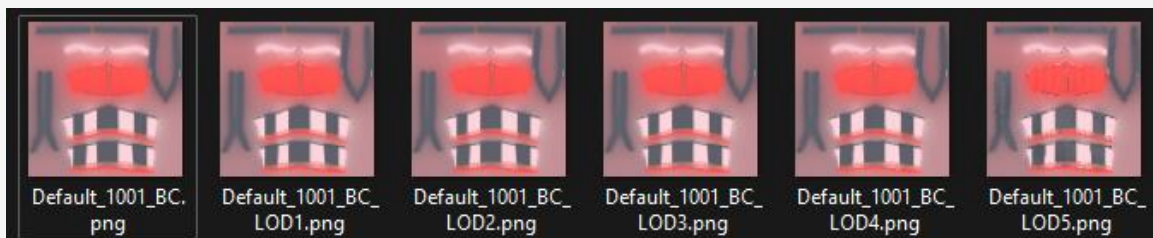

The original file does not have a suffix, which is logical. LOD1 has a twice smaller resolution than the original image, LOD2 - four times, LOD3 – eight times and so on.

Solid images that do not carry useful information are always removed from objects and converted into values.

The minimum texture resolution is 256 pixels (can be changed in settings). This means that the script will not create LODs with 128x128 or 64x64 resolution.

Selected Nodes:
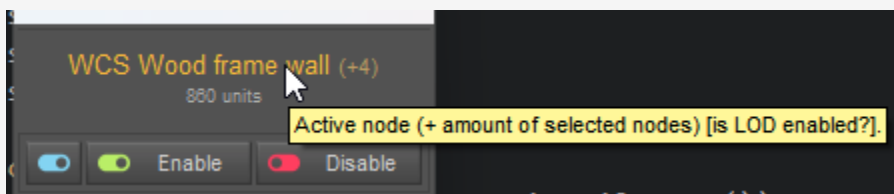
This label shows the current active object and the amount of other selected objects. The number below it shows the distance to the active node in Daz units.

"Enable All", "Enable" and "Disable" buttons marks all or selected objects as enabled or disabled for the script. By default, all objects are marked as enabled. Disabling means that the objects will always be updated with their original textures after the next Update LODs process.

Checkboxes:
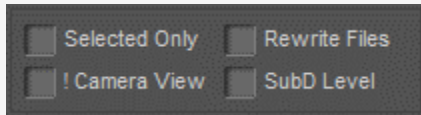
Selected Only:
Should the script update LODs on selected objects only?

Rewrite Files:
Should the script rewrite previously generated LODs while creating new ones? Useful in case if you changed or edited original textures.

Camera View (beta):
Should the script take into account the camera view? All the objects behind the camera will be updated with the lowest texture resolution. Right now, this function does not work well and have known bugs.

**SubD Level:**
Should the script affect object Mesh Resolutions and lower their Viewport/Render values depending on the distance? Creates additional "Target View SubD Level" and "Target Render SubD Level (Minimum)" in General/Mesh Resolution.
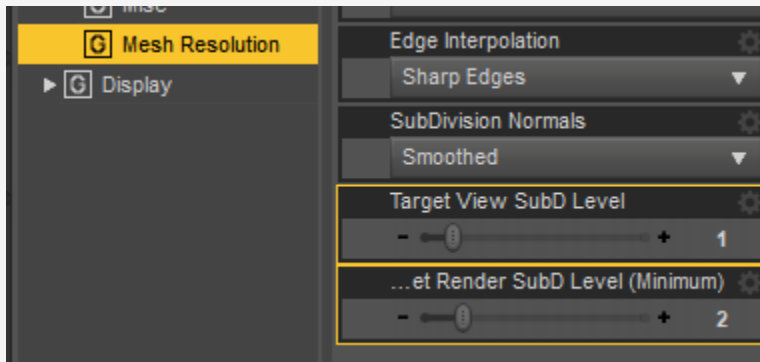
These values define the preferred SubD value. The closer the camera to the object is, the higher SubD Level will be applied to the original slider.

Sliders:
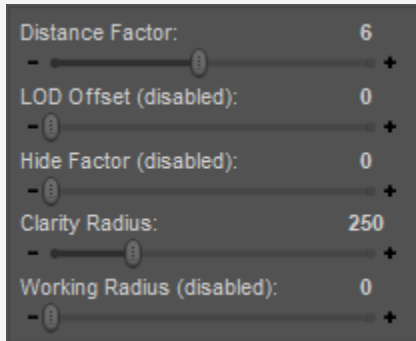
You can click on any number to start typing value. Press Enter to apply or Esc to cancel. Alt + LMB to reset to default value.

Distance Factor:
The main slider that is a factor for multiple values that define the final LOD. Technically the value is clamped to 0.25-12.0, so the zero value is actually 0.25. The lower the slider value is, the smaller textures will be created and applied (or in other words, the higher LODs will be created).

LOD Offset:
In cases where you need extreme optimization, you can resort to using this slider. On top of the final calculations, it further increases the LOD, noticeably reducing picture quality.

Hide Factor:
If non-zero, the factor that tells the script what objects with what scale should be temporarily hidden from the scene. Useful for scenes with a lot of very small objects that cannot be seen from faraway.
This doesn't hide objects with emissive surfaces.
Doesn't unhide objects that were hidden by the user.

Clarity Radius:
If non-zero, the radius within which all the objects will be updated with their original textures. Useful when you want to make sure that objects around the camera have their original textures.
Draws a blue sphere that shows the radius and highlights the affected objects.
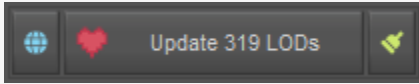
Working Radius:
If non-zero, the radius within which the script will work. Useful when your scene is extremely large and you only want to update a small number of objects around the camera.
Draws a red sphere that shows the radius and highlights the affected objects.
The slider is disabled when Selected Only is checked.

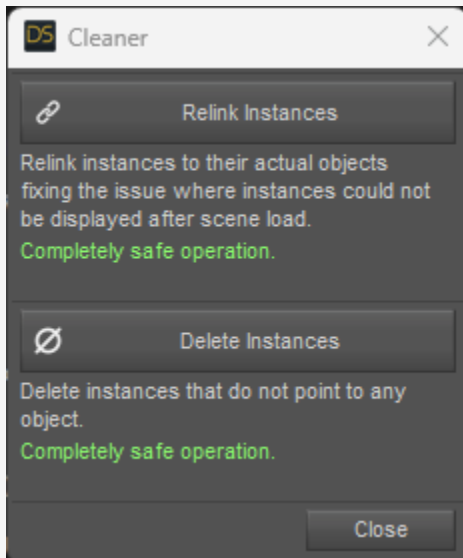## Update LODs Buttons:

Reset LOD(s) *(a blue globe)*:
Resets objects to their original texture size. Does not apply solid textures back as they're super useless.

Update ... LOD(s) *(a pixelated heart)*:
The button that starts the main process. Shows the number of objects that will be processed by the script.

Show Cleaner *(a green brush) (WIP!)*:
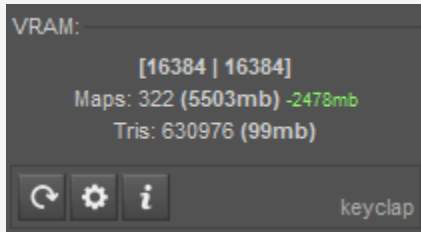Show the window with multiple buttons.

Relink Instances
Relink scene instances to their actual targets. Fixes the bug when instances do not draw mesh even though they are linked.
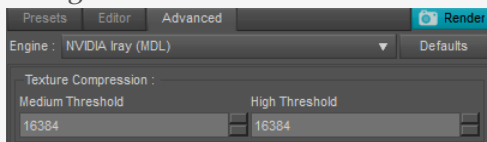
Delete Instances
Delete useless scene instances.

VRAM Calculator:

Compression Threshold [16384 | 16384]:
Shows current texture compression threshold settings [medium | high] that you basically change here:

0 and **16384** – means disabled. I recommend you to use **16384** instead of **0** as setting to zero doesn't instantly update the value.

<mark>Important info for nerds:</mark> Daz iRay has 2 texture compression methods (medium and high). Each one is triggered by the texture resolution set in the fields above. Texture resolution is the maximum of width or height of an image. If the image is 1024x2048, then its resolution is 2048. By my calculations, medium method compresses image by ~87%, high method compresses by ~91%. But the actual value is too vague, so do not trust me.

Maps:
Shows the number of maps used in the scene by visible objects. Also takes into account current Environment HDRI image. The number in MB is the **minimum** amount of VRAM that will be used by the renderer. The final VRAM usage depends on the texture compressor, the render resolution, denoiser and iRay mood.

Tris:

Shows the number of triangles and approximate VRAM usage. Here is how the script calculates it:

```
// Try to calculate mesh VRAM
var nVRAMMesh = 0;
// Vertex (index) - 16/32bit
nVRAMMesh += nVertices * 16;
// Vertex (x, y, z) - 12bit
nVRAMMesh += (nVertices * 3 * 12);
// Edge (index, index) - 16b/32bit
nVRAMMesh += nEdges * (2 * 16);
// Vertex Normal (index) 16/32bit
nVRAMMesh += nNormals * 16;
// Vertex Normal (x, y, z)
nVRAMMesh += nNormals * (3 * 12);
// +50% just because we don't know about other data
nVRAMMesh *= 1.5
// Convert Bytes to MB
var nMesh = Math.round(nVRAMMesh / (1024 * 1024 * 8));
```

Additional Buttons:
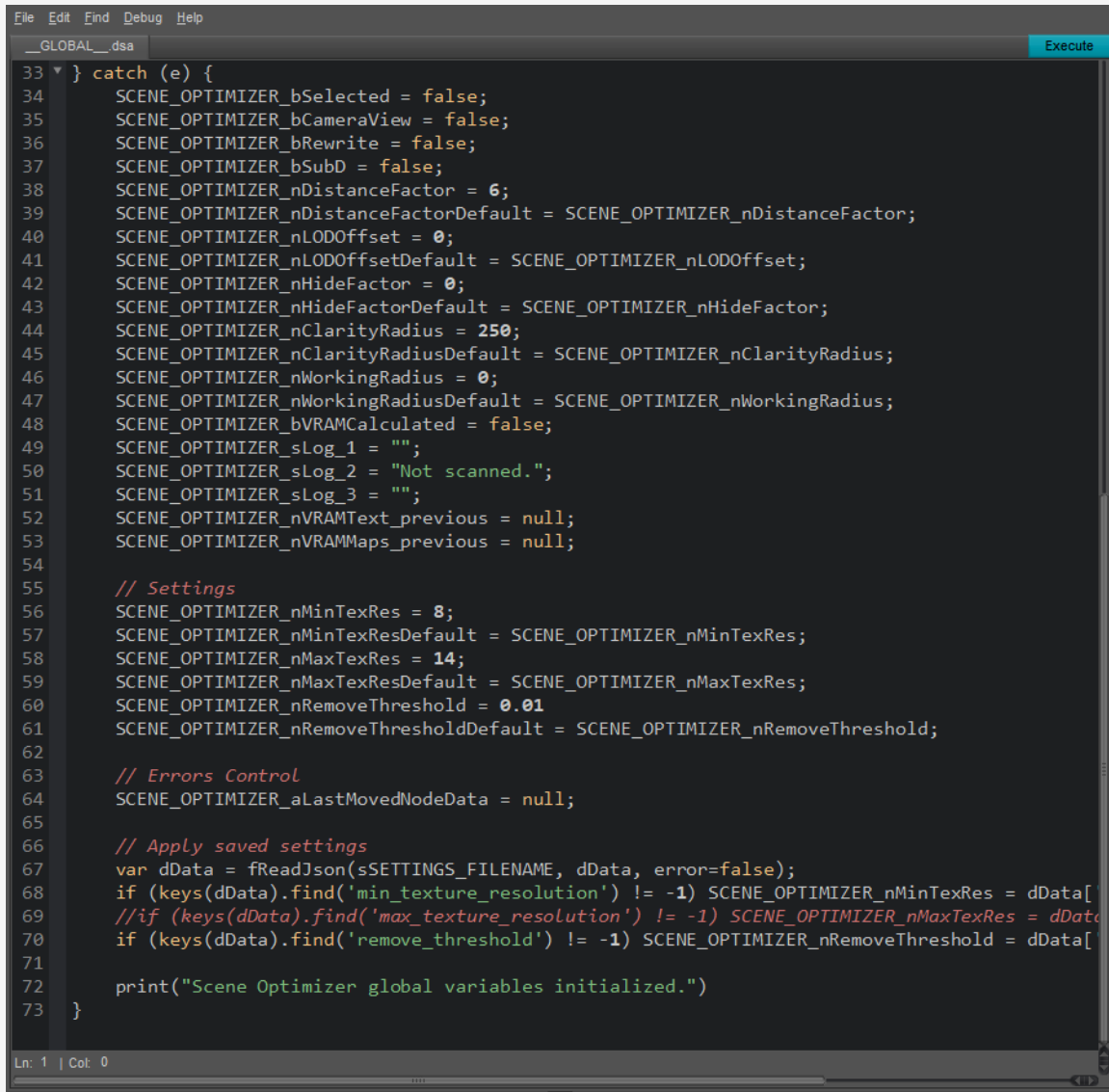
| ↩ Undo | → | Close |

Undo and Redo:

Hovering over each one displays the description of an action that will be returned.

Close:

Close.

Script Convention

Here you can see all the global variables that the Daz LOD System & Cleaner initializes and uses:

```
File  Edit  Find  Debug  Help
__GLOBAL__.dsa                                                                    Execute
33 ▼ } catch (e) {
34        SCENE_OPTIMIZER_bSelected = false;
35        SCENE_OPTIMIZER_bCameraView = false;
36        SCENE_OPTIMIZER_bRewrite = false;
37        SCENE_OPTIMIZER_bSubD = false;
38        SCENE_OPTIMIZER_nDistanceFactor = 6;
39        SCENE_OPTIMIZER_nDistanceFactorDefault = SCENE_OPTIMIZER_nDistanceFactor;
40        SCENE_OPTIMIZER_nLODOffset = 0;
41        SCENE_OPTIMIZER_nLODOffsetDefault = SCENE_OPTIMIZER_nLODOffset;
42        SCENE_OPTIMIZER_nHideFactor = 0;
43        SCENE_OPTIMIZER_nHideFactorDefault = SCENE_OPTIMIZER_nHideFactor;
44        SCENE_OPTIMIZER_nClarityRadius = 250;
45        SCENE_OPTIMIZER_nClarityRadiusDefault = SCENE_OPTIMIZER_nClarityRadius;
46        SCENE_OPTIMIZER_nWorkingRadius = 0;
47        SCENE_OPTIMIZER_nWorkingRadiusDefault = SCENE_OPTIMIZER_nWorkingRadius;
48        SCENE_OPTIMIZER_bVRAMCalculated = false;
49        SCENE_OPTIMIZER_sLog_1 = "";
50        SCENE_OPTIMIZER_sLog_2 = "Not scanned.";
51        SCENE_OPTIMIZER_sLog_3 = "";
52        SCENE_OPTIMIZER_nVRAMText_previous = null;
53        SCENE_OPTIMIZER_nVRAMMaps_previous = null;
54
55        // Settings
56        SCENE_OPTIMIZER_nMinTexRes = 8;
57        SCENE_OPTIMIZER_nMinTexResDefault = SCENE_OPTIMIZER_nMinTexRes;
58        SCENE_OPTIMIZER_nMaxTexRes = 14;
59        SCENE_OPTIMIZER_nMaxTexResDefault = SCENE_OPTIMIZER_nMaxTexRes;
60        SCENE_OPTIMIZER_nRemoveThreshold = 0.01
61        SCENE_OPTIMIZER_nRemoveThresholdDefault = SCENE_OPTIMIZER_nRemoveThreshold;
62
63        // Errors Control
64        SCENE_OPTIMIZER_aLastMovedNodeData = null;
65
66        // Apply saved settings
67        var dData = fReadJson(sSETTINGS_FILENAME, dData, error=false);
68        if (keys(dData).find('min_texture_resolution') != -1) SCENE_OPTIMIZER_nMinTexRes = dData[
69        //if (keys(dData).find('max_texture_resolution') != -1) SCENE_OPTIMIZER_nMaxTexRes = dData
70        if (keys(dData).find('remove_threshold') != -1) SCENE_OPTIMIZER_nRemoveThreshold = dData[
71
72        print("Scene Optimizer global variables initialized.")
73 }
Ln: 1 | Col: 0
```