LiDAR Point Cloud Plugin Overview

Use the LiDAR Point Cloud plugin to import, visualize, and edit point clouds in Unreal Engine.

With the **LiDAR Point Cloud plugin**, you can import, visualize, and edit point clouds in Unreal Engine. The plugin also supports a variety of coloration and shading techniques. Because of its multiple optimizations and dynamic LOD (Level of Detail) scaling, you can work with large datasets without any significant loss of performance, even at runtime.

Supported Point Cloud File Formats

A **point cloud** is a set of data points where each point is defined by its XYZ coordinates and, optionally, its color. Unreal Engine supports the following point cloud file formats:

Extension	Description
`*.xyz`, `*.pts`, `*.txt`	General types of ASCII point cloud file formats that can contain either: • Point coordinates (X Y Z for each point, expressed in meters) • Point coordinates plus colors, (X Y Z R G B for each point). Notation can be float (like 0.00892855) or scientific (like 8.92855E-03).
`*.las`, `*.laz`	LAS is a public file format for the interchange of three-dimensional point cloud data between users. Although developed primarily for the exchange of LiDAR point cloud data, this format supports the exchange of any three-dimensional X, Y, Z tuplet. This binary file format is an alternative to proprietary systems, as well as to generic ASCII file interchange systems, and is widely used. LAZ files are compressed LAS files. They are much smaller, but their import is also proportionally slower. Unreal Engine supports 8-bit, 12-bit, and 16-bit LAS / LAZ files.

Extension	Description
`*.e57`	E57 is a compact, open-source file format that stores and exchanges 3D imaging data such as point clouds, images, and metadata.

Importing and Exporting Point Clouds

To **import** a point cloud asset, you can use either of these methods:

- Drag a point cloud, saved in a supported file format, to your **Content Browser.**
- In the **Content Browser**, click **Add/Import**, then navigate to and select the desired file.

> On import, meters are converted to Unreal Units (UU), where 1 UU = 1 cm. To use a custom import scale, open the Project Settings for the LiDAR Point Cloud plugin, then change the **Import Scale** value.

You can **export** a point cloud asset to ASCII or LAS using the existing **Unreal Export** tool. In the **Content Browser**, right-click the asset, then select **Asset Actions > Export**.

> On export, Unreal Units are converted back to meters (multiplying the value by 0.01). To use a custom export scale, open the Project Settings for the LiDAR Point Cloud plugin, then change the **Export Scale** value.

Performance

Performance is primarily determined by the **global point budget**, which sets the maximum number of points that can be displayed at one time. Using a shared budget as opposed to a per-component budget allows for efficient rendering of many assets at the same time, as the system will select the most optimal points from all of the visible components. A higher point budget means higher quality with a higher performance cost. The global point budget saves VRAM and improves the frame rate, but does not reduce overall RAM usage.

You can set the global point budget using console variables. The higher the point budget, the denser the point cloud.

For more information, see the console variables section of the LiDAR Point Cloud Plugin Reference page.

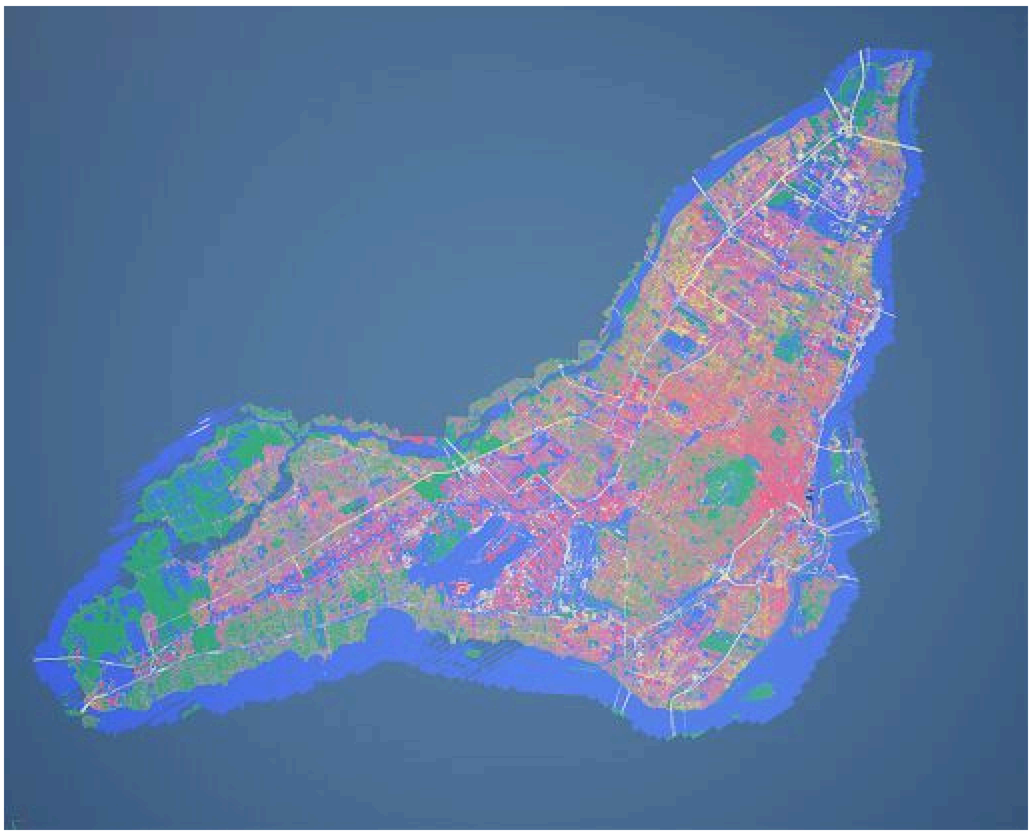
On-Demand Streaming

When opening a point cloud, Unreal Engine only loads the necessary header information and streams the actual bulk data as needed. This results in fast asset loading and lowers total

RAM consumption.

> The newly imported assets will be kept in system memory and won't be able to leverage streaming functionality until they are saved. Once they are saved, the editor will release the memory for that asset.

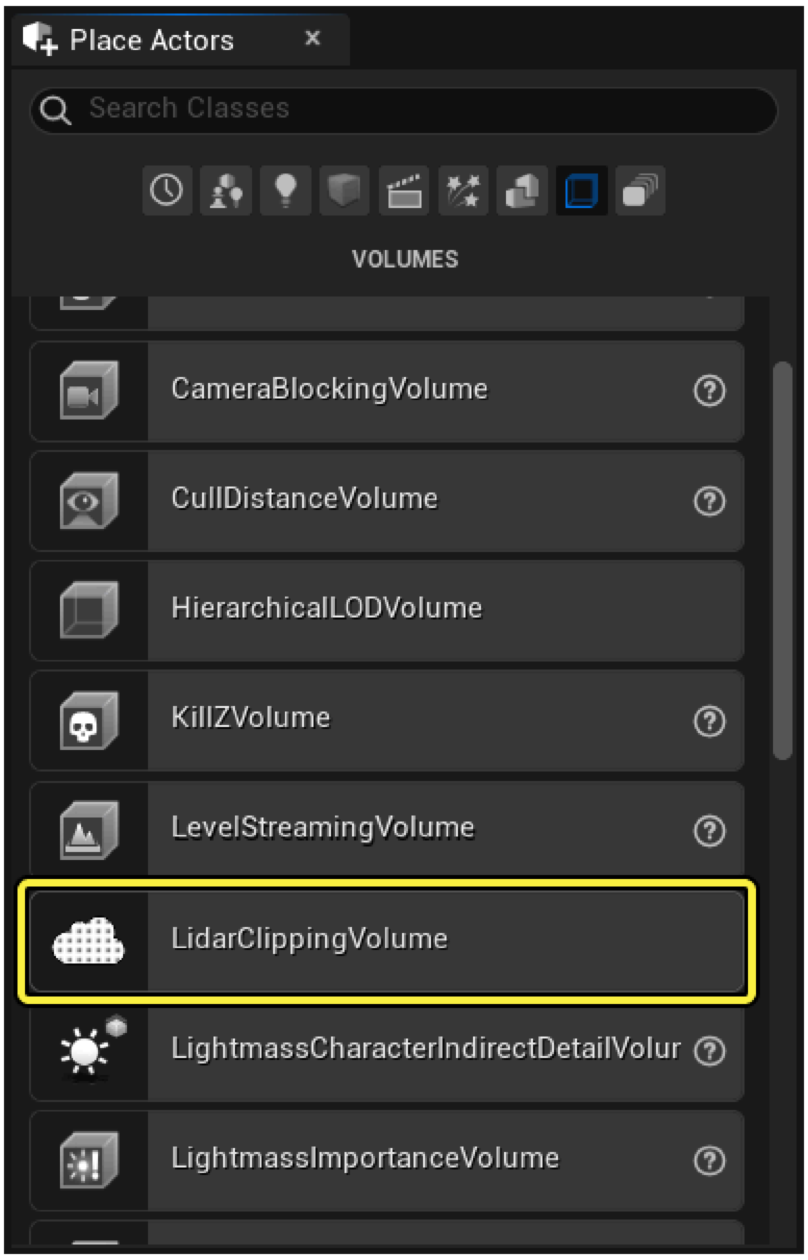
Loading large numbers of point cloud files in the engine still requires a significant amount of RAM when parsing point cloud data and processing it as Unreal Engine assets. For example, this publicly available LiDAR data from the city of Montreal loaded in Unreal Engine has the following performance indicators:



Individual LAS files	~684 tiles for a total of 253 GB on disk
Unreal RAM usage	~3.5 GB for a budget of 1 million points
Frame Rate	120 fps, with a global budget count of 1 million points
Total Amount of Points	Average 14.3 million points per 1 km x 1 km tile * 621 files = ~8.9 billion points

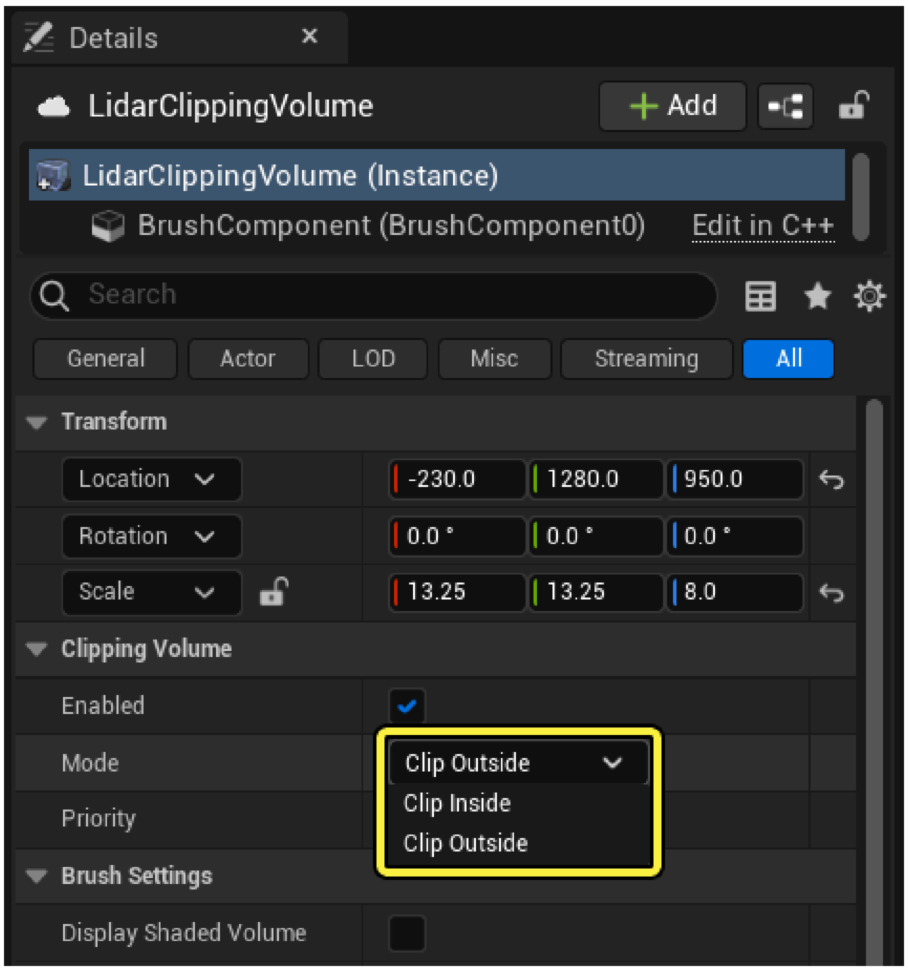
Clipping Volumes

If you only want to show a section of your data, but don't want to delete the rest of it, you can use a **Lidar Clipping Volume** Actor. With the LIDAR Point Cloud plugin enabled, you can find this Actor in the **Place Actor > Volume > Lidar Clipping Volume**.



Click image for full size.

You can configure the properties of a clipping volume from its **Details** panel.



Transform

Location	-230.0	1280.0	950.0
Rotation	0.0 °	0.0 °	0.0 °
Scale	13.25	13.25	8.0

Clipping Volume

Enabled	✓
Mode	Clip Outside (Clip Inside, Clip Outside)
Priority	

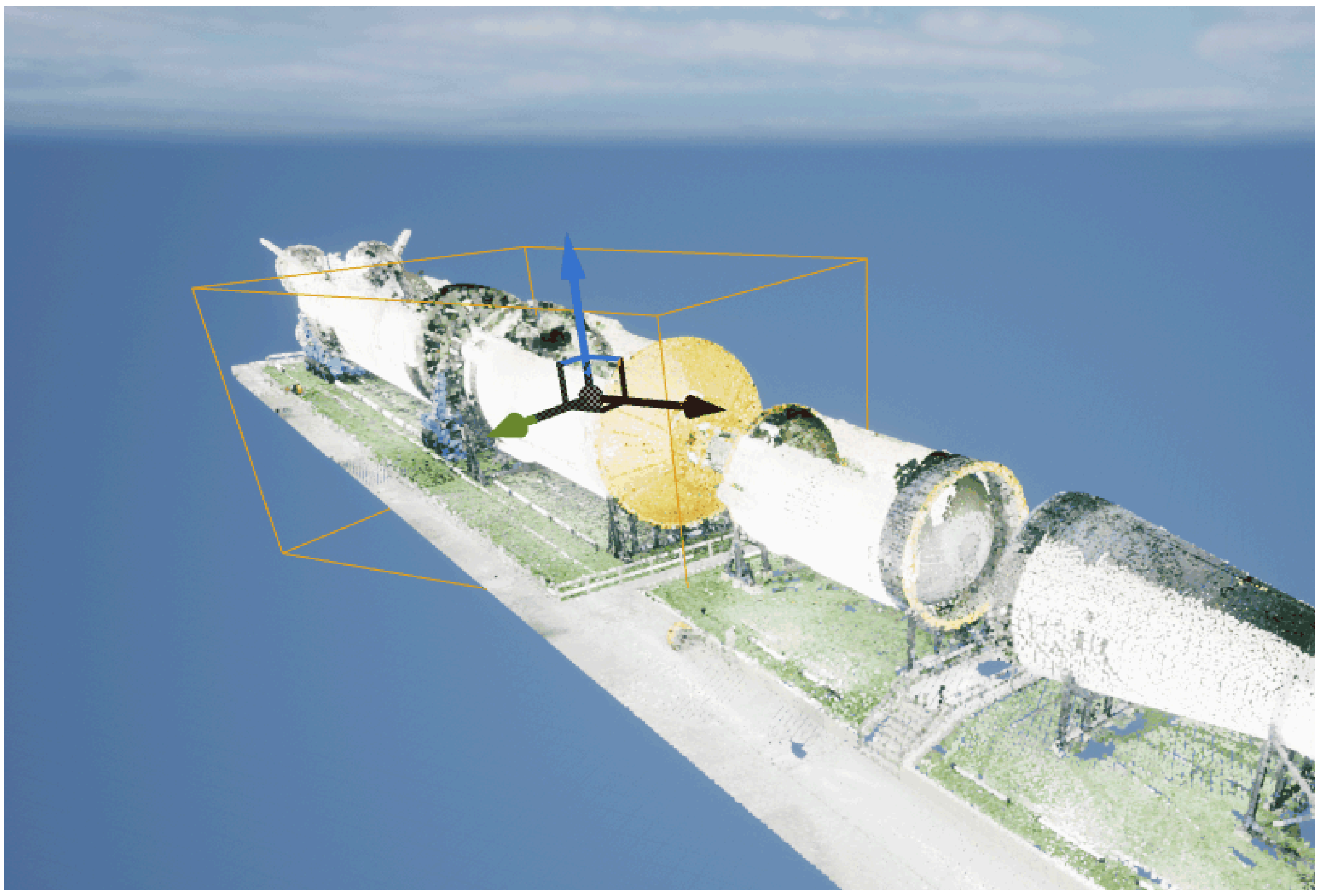
Brush Settings

Display Shaded Volume	

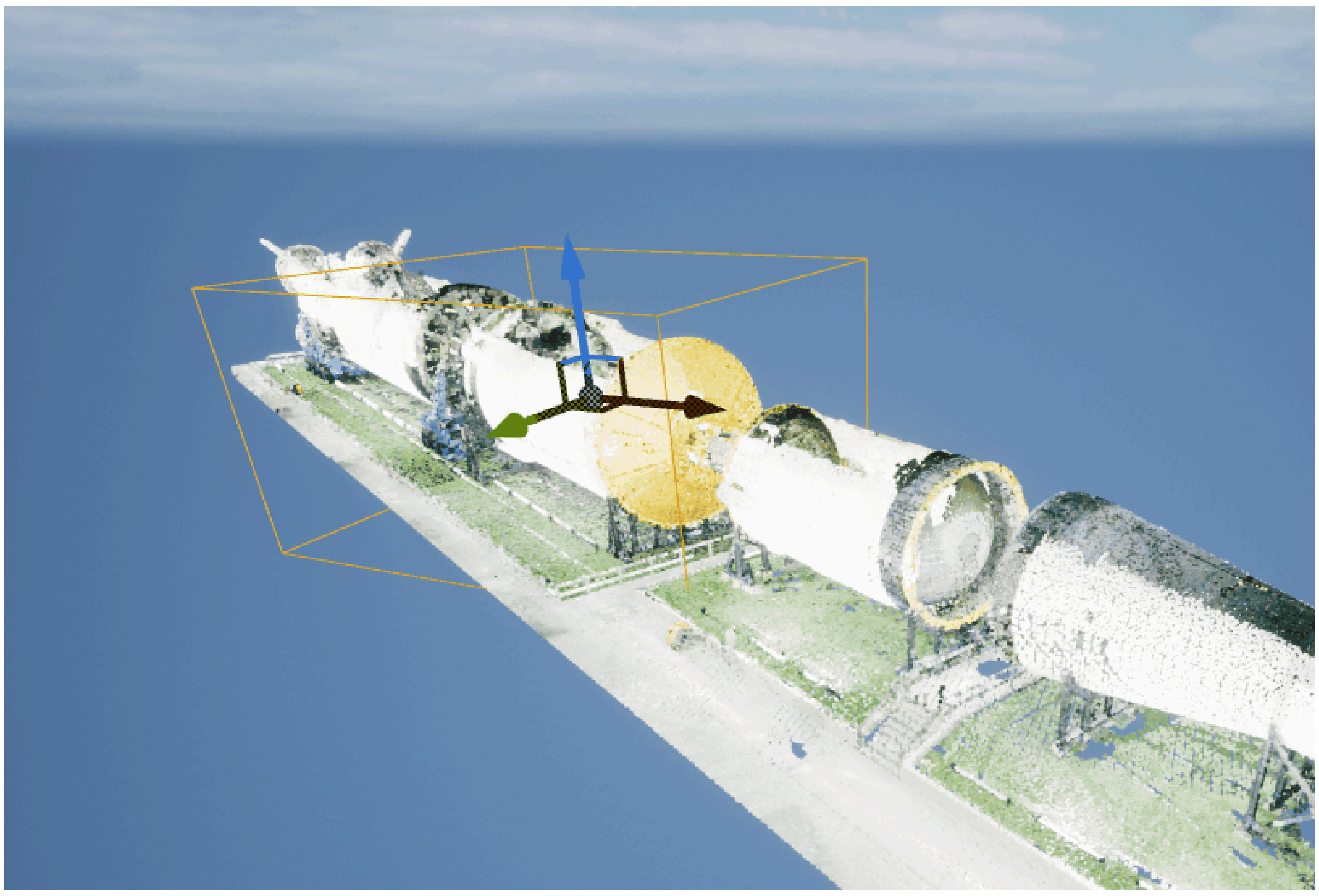
Click image for full size.

There are two **Modes** you can choose from:

- **Clip Outside** hides all the point cloud data outside the clipping volume.



- **Clip Inside** hides all the point cloud data inside the clipping volume.



You can use up to 16 clipping volumes in a Level. If you have overlapping volumes, use the **Priority** setting to determine which one takes priority.

Changing Data at Runtime

The LiDAR Point Cloud plugin supports inserting, removing, and modifying data at runtime, including in the packaged executable.

There are a few considerations to keep in mind when doing this:

- The inserted data has to be within asset bounds. Otherwise, it will interfere with the LOD system.
- Moving points doesn't relocate them to neighboring LODs, so it's recommended not to move points around too much.
- When inserting many points multiple times, LOD generation can become resource-intensive.

To mitigate the issues described above, you can initialize the point cloud asset with bounds that are much larger than necessary (for example, 100,000 in each axis) and essentially disable the LOD system altogether by changing the **Max Bucket Size** setting in the plugin's **Project Settings** to a very large number, like 1,000,000,000.