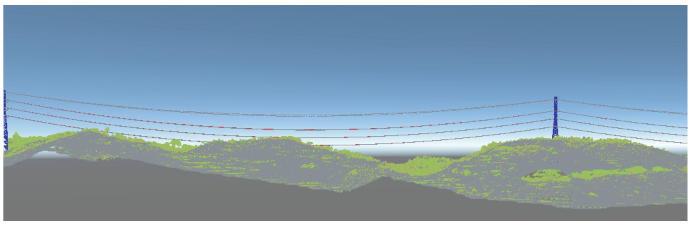
Visualization of Large Point Cloud Models on Unity

J.M. Santana, A. Trujillo and S. Ortega

CTIM CENTRO DE TECNOLOGÍAS DE LA IMAGEN

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Large point cloud rendering has become a very relevant topic on 3D graphics as scanners and other sources of 3D point data are nowadays available to companies and the general public. In this project, we propose an implementation of a point cloud viewer, designed for the full-detail visualization of virtually unlimited point clouds for their inspection on short ranges. This work presents the data structure and the LoD technique to achieve a real-time rendering of the model, making emphasis on the details of an initial prototype based on Unity.


Long power corridors are scanned by an airborned LiDAR scanner achieving a resolution of 50 p/m²

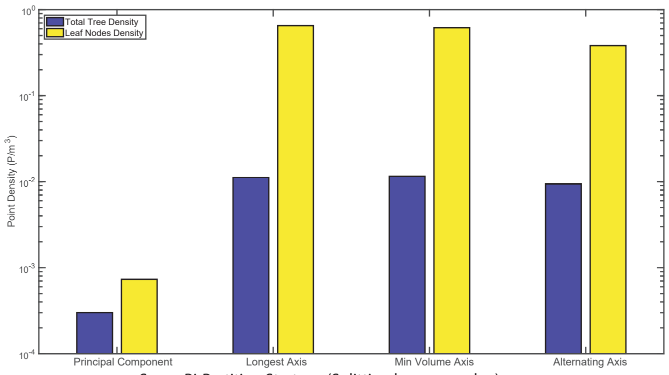
Project Motivation

The particular use case that motivated this project has been the visual inspection of airborne scans of long power line corridors, from which automatic and manual classification has been produced. The longitude of these corridors varies within a range of 100 - 200 km., with an average width of ~100 m. and a point density up to 50 p/m2. Hence, the multi-corridor models are usually encoded in several LAS files, adding up to hundreds of GBs. The final intention of the project is to accurately display the point cloud within the range of view, enabling a seamless navigation across the model, while allowing the inclusion of other 3D elements and the use of tools provided by the Unity framework. The use case imposes a local but holistic rendering of the point cloud, facilitating the spotting of undesired noise generated by the LiDAR scanner, as well as misclassified points.

Model Preparation

The size of these models (hundreds of millions of points) imposes a hierarchical partitioning that enables out-of-core rendering. The literature [Fra17, Sch16] covers a series of possible subdivisions of the point cloud model in order to serve manageable chunks to the GPU. In this project, we have opted to use a binary tree partition of the space, similar to the one proposed by Gobbetti and Marton [GM04]. However, as our model must be rendered with all its points at any distance, no coarser levels of detail have been generated in upper levels of the tree.

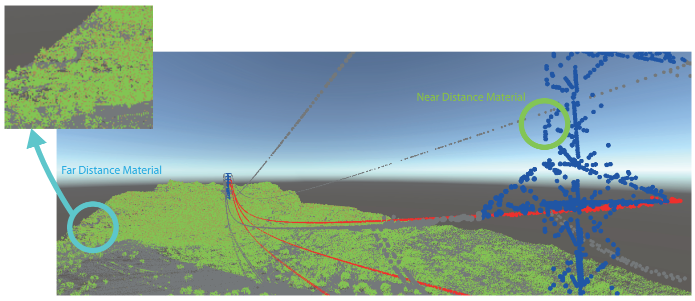
All the nodes of the tree are contained within a tight axis-aligned box, which is precomputed and serves the LoD test and point-picking strategies. Dividing the nodes at the mean value along their longest axis offered the best results, minimizing the overall bounding volume of the nodes. Leaf nodes contain xyz point offsets from the node center and clasification data in single precision. In Unity, the whole binary tree forms a hierarchy of GameObjects, and the LoD test uses the precomputed bounding boxes (as Bounds instances).



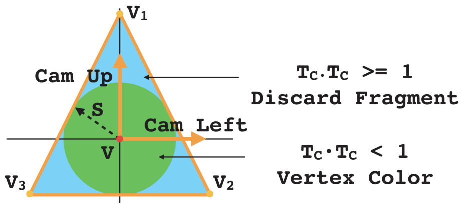
Space Bi-Partition Strategy (Splitting by mean value)

Discrete Level of Detail Strategy

In order to keep a high-performance rendering without removing points from the visible area of the model, a strategy for the rendering at different distances was devised. At a short distance, the goal is to show the points as rounded objects with a fixed physical size. At long distances, points must preserve a minimum screen-space size to remain visible and to avoid undesirable aliasing problems. However, Unity does not enable the user to establish a screen point size to preserve DirectX 11 as a target platform. Our solution consists in using two different materials, named Far Distance Material (FDM) and Near Distance Material (NDM), that are interchanged depending on the distance of the node to the viewer.



Points are rendered with a world-space size at a near distance and a screen pixel-size at far distances

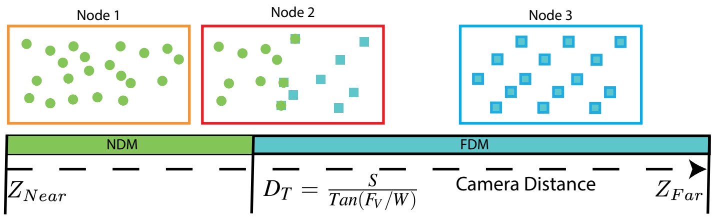


$$V_1 = V + 2S \cdot \overrightarrow{Cam_{Up}}$$
$$V_2 = V + S \cdot (\overrightarrow{Cam_{Up}} + \overrightarrow{Cam_{Left}})$$
$$V_3 = V + S \cdot (\overrightarrow{Cam_{Up}} - \overrightarrow{Cam_{Left}})$$
$$T_{C1} = \left\{0, \frac{1}{Sin(\pi/6)}\right\}$$
$$T_{C2} = \left\{\frac{1}{Tan(\pi/6)}, -1\right\}$$
$$T_{C3} = \left\{-\frac{1}{Tan(\pi/6)}, -1\right\}$$

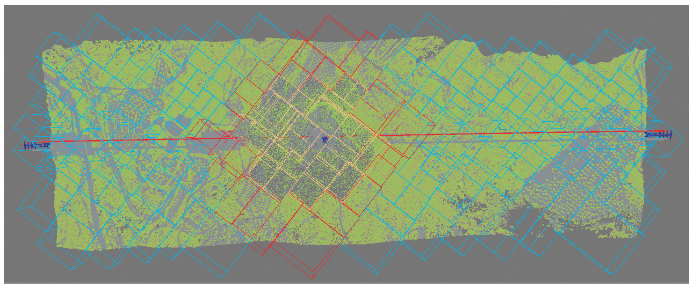
Near Distance Material - Geometry Shader Computations

Dynamic Material Rendering

By considering the vertical screen resolution (W) and the camera field of view (FV) provided by Unity, we establish the distance threshold at which the projected size of a NDM point equals one pixel. For any given node we can compute the distance to the furthest and closest point of its bounding box, which constrains the distance to any point within it.

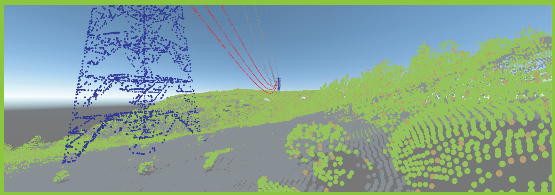


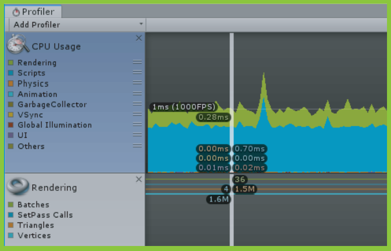
Distance-Based Material LoD Strategy


Blue nodes use the FDM, orange ones the NDM and red ones are rendered with both.

Performance

The prototype has been tested with model up to ~56 million points, achieving rendering times around 1 ms. on a desktop machine (Intel i5 Processor, Nvidia GTX 1060). The memory consumption and garbage collection times are also bounded by using object pooling techniques.





References

[Fra17] FRAISS S. M.: Rendering large point clouds in unity, Sept.2017. URL: https://www.cg.tuwien.ac.at/research/publications/2017/FRAISS-2017-PCU/
[GM04] GOBBETTI E., MARTON F.: Layered point clouds: a simple andefficient multiresolution structure for distributing and rendering gigantic point-sampled models. Computers & Graphics 28, 6 (2004), 815–826
[Sch16] SCHÜTZ M.: Potree: Rendering large point clouds in webbrowsers. Technische Universität Wien, Wiede´n (2016)