UnityPIC: Unity Point-Cloud Interactive Core

Y. Wu[1], **H. Vo**[1,2,3], **J. Gong**[4], & **Z. Zhu**[1,2]

1. *Department of Computer Science, The Graduate Center*
2. *Department of Computer Science, The City College of New York*
3. *Center of Urban Science and Progress, NYU*
4. *Civil and Environmental Engineering, Rutgers University*

Accepted at the Eurographics Symposium on Parallel Graphics and Visualization (EGPGV'21)

Introduction

Point cloud data is becoming increasing popular as a way to capture real-world environments and objects. There are many applications of point cloud, such as urban planning, disaster evaluation, facility modeling, virtual site visits, autonomous driving. Many of these applications recreate the captured real-world instance and allow users to interact with it virtually.

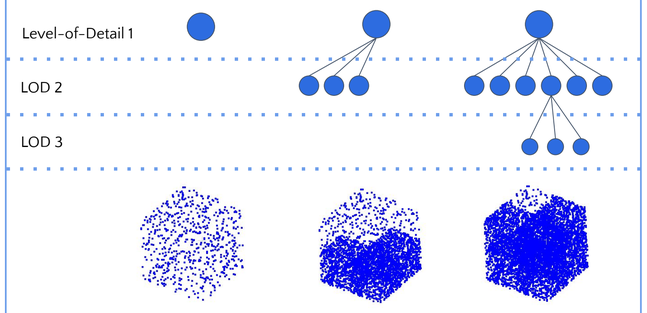
Challenge: Amongst these interactive applications, a scalable rendering pipeline for point clouds is a shared commonality. However, when creating these applications, the rendering pipeline is often re-invented to suit the specific application on a specific platform. In other words, the developed pipeline lacks portability across development and production platforms.

Solution: Motivated by the above, our work proposed the abstraction of the point-cloud rendering pipeline as a Unity Component, called UnityPIC. The goal of UnityPIC is to simplify the creation of point-cloud applications through the provision of a general and performant point-cloud rendering pipeline with high interactive visual quality on a highly accessible and portable development platform.

Key Contributions

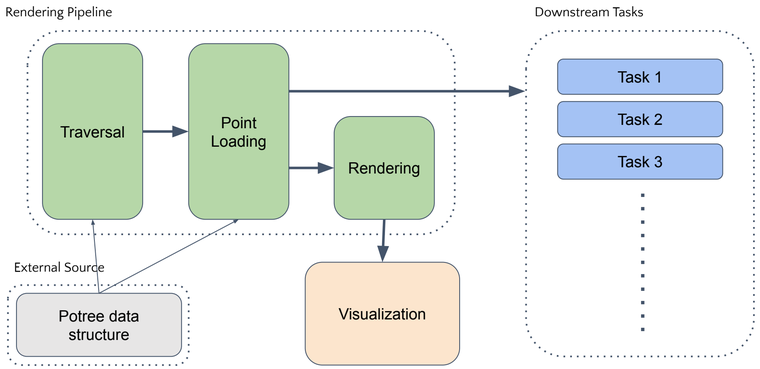
- **UnityPIC**, a Unity Component, is an abstraction of a scalable point cloud rendering pipeline.
- **Dynamic Batching** achieves optimal adaptive point-sizing. *The current state-of-the-art solution was proposed in Potree by [Schutz, 2016].*
- **Delayed Expansion** is an approximate rendering algorithm for rendering points. *To our knowledge, there exists no similar works in the current literature.*

Preliminaries



The raw data is often pre-formatted into an hierarchical acceleration structure. The structure can be traversed to efficiently determine the set of visible points during run-time. In addition, the hierarchical nature of the structure allows the selection of the appropriate level-of-detail to render.

General Application Pipeline



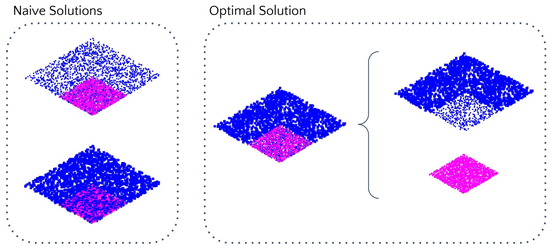
The rendering pipeline contains 3 main processes:

- **Traversal** - The determination of visible points.
- **Point loading** - The loading of points from an external source.
- **Rendering** - The transfer of points from CPU to GPU and the rendering of those points.

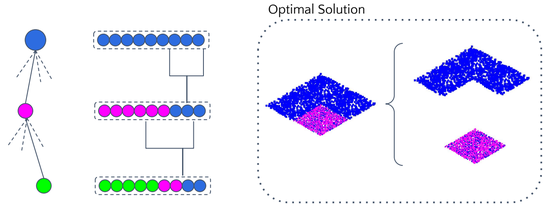
The loaded points are available to be processed in other downstream tasks, if any.

Dynamic Batching

Adaptive point-sizing is an existing challenge when rendering hierarchical point-clouds. Achieving the optimal result requires per-point density information, however, only per-node density information is available. The state-of-the-art method is proposed by [Schutz, 2016], where point sizes are computed on the GPU.



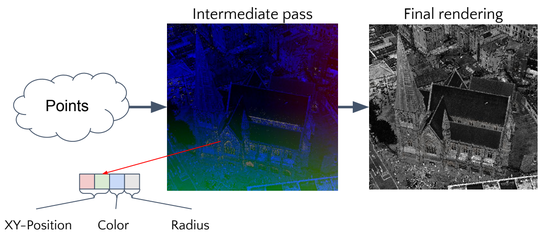
We proposed a inheritance scheme for the hierarchical structure which achieves the optimal result using only per-node density information.



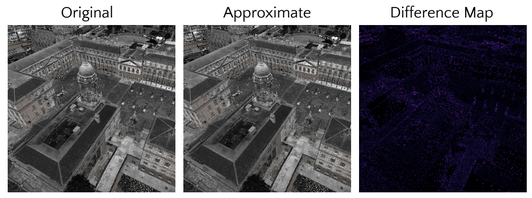
[Schutz, 2016] Schütz, Markus. (2016). Potree: Rendering Large Point Clouds in Web Browsers.

Delayed Expansion

We proposed an approximate rendering algorithm for points which significantly improves GPU performance. The proposed algorithm first performs occlusion culling by rendering pixel sized points in an intermediate pass. The point attributes are stored in the RGBA components of the intermediate pass.

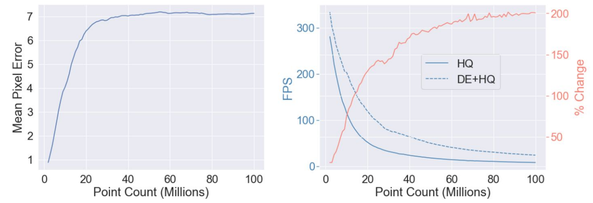


The final rendering suffers from small errors due to the occlusion culling and slight differences in rasterization. The errors can be visualized using an image difference evaluator as follows.



Experiments

Approximate Rendering - Errors & Performance



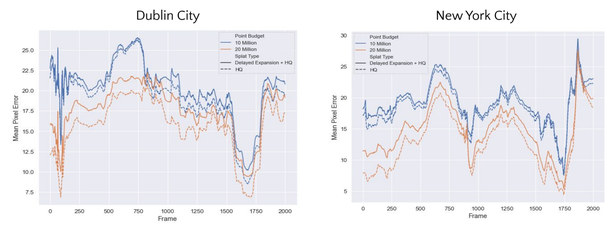
(a) *Approximation Error* **(b)** *FPS Difference*

On the right, HQ refers to the baseline rendering method and DE+HQ refers to the approximated version of that method.

Interactive Visual Performance

Two different point cloud datasets were used: Dublin City (1.4 billion points), and New York City (22.8 billion). The experiments are run on Windows, with an Intel(R) Core i9-9900K CPU and Nvidia TITAN Xp GPU.

The interactive results are compared against a baseline rendered with 300 million points. Two point budgets are used: 10 million (blue), and 20 million (orange). Non-approximate (dotted) and approximate (solid) are compared. The pixel differences are at most about 11.7% (30/255).



Acknowledgements

- The National Science Foundation Awards #1827505 and #1737533.
- Alfred P. Sloan Foundation Award G-2018-11069.
- Air Force Office for Scientific Research (Award #FA9550-21-1-0082)
- The Intelligence Community Center of Academic Excellence (IC CAE) at Rutgers University (Awards #HHM402-19-1-0003 and #HHM402-18-1-0007)
- Vingroup Innovation Award VINIF.2019.20.