Topics in Computer Graphics

Final project report

Ben Benchaya

Asaf Chelouche

Abstract

In this project, we take a cloud that consists of millions of points in 3D , and render them on screen in an interactive environment. The input file ("point cloud") is a proprietary file of a commercial company, and is the output of a high-accuracy laser scanner. On this cloud, 3D of objects are rendered, superimposed on the cloud. The objects are classified into categories, so the user has the ability to display the objects in a selective manner.

Technical information

This project is intended to work on Windows based machines, using Visual Studio.
The core function of the app is implemented in C++ via two open source libraries: OpenFrameworks and PCL (Point Cloud Library).
All of the input files (the clouds, their respective transformations and the 3D models) are provided by a commercial company, which scanned the streets of Vancouver, British Columbia, Canada.

Implementation

We received preliminary app that loaded a single cloud into the origin of the global space. We then cleaned up any unneeded or incomplete functionality.
We then added the functionality to load an arbitrary number of clouds, so that whole streets could be displayed. Afterwards, we parse the given transformations (in `.csv` format) files for all of the clouds and all of the models.
When the user selects a cloud to load, the cloud is automatically associated to its respective transformation, according to the cloud's filename.
Afterwards, cultural 3D objects are associated to the cloud according to:

a) the cloud's center;
b) the model's center;
c) a distance threshold that is defined in the system (default: 65 world space units).

Each loaded cloud is always rendered and the user can select to display filtered clouds: these are clouds that contain only five percent of the original cloud's points, which are selected using even distribution.
The purpose of the filtered clouds is to improve responsiveness to user input, especially when several clouds are rendered concurrently.
When the user chooses to display the models, they are all displayed initially, and he can select which categories he wishes to toggle on and off for display.

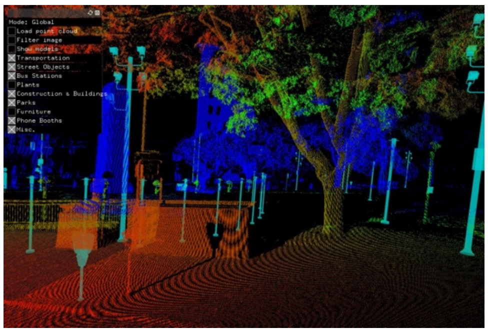
User Interface

The camera can be moved around the space in the same way as an FPS video game: moving the mouse while left-clicking changes orientation, and the following keys move the camera itself:

Key	**Direction of movement**	**Key**	**Direction of movement**
w	Forward	W	Up
a	Left	A	Pan clockwise
s	Backward	S	Down
d	right	D	Pan counter-clockwise

Mouse scroll also moves forwards and backwards.

Sample screenshots



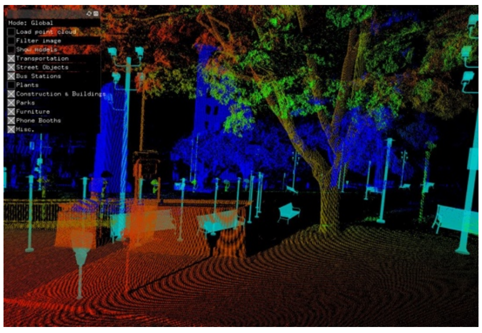


Fig. 1: Model category selection – furniture deselected on the left; selected on the right

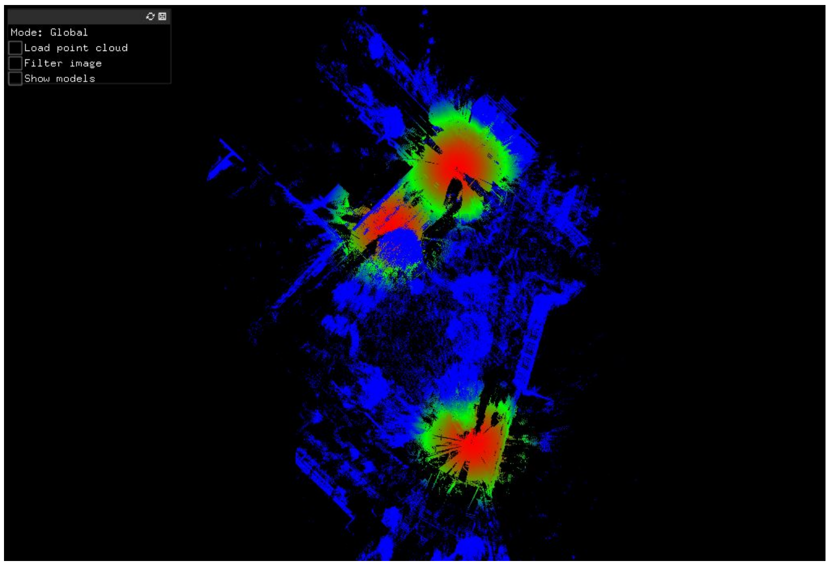


Fig. 2: 3 clouds rendered concurrently, bird's-eye view

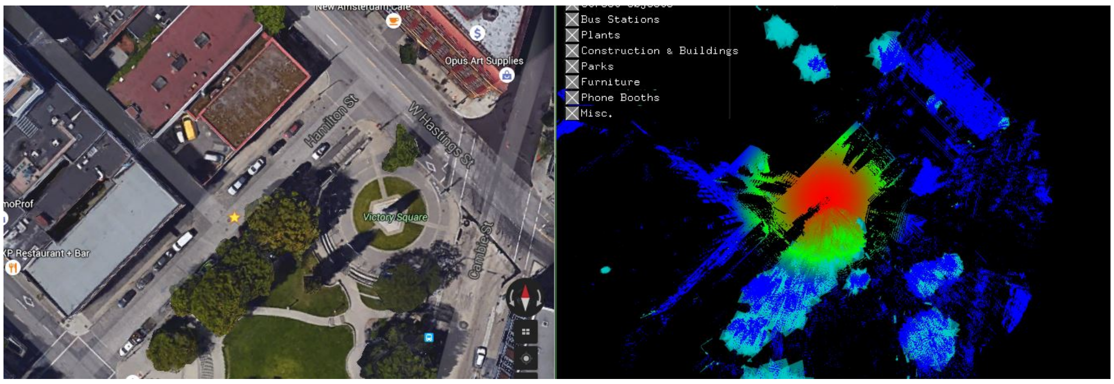


Fig. 3: Bird's-eye view comparison between Google Maps and the application

461 Hamilton St., Vancouver, BC

Sources:

Codebase - https://github.com/asafch/pointcloud_viewer

PCL – http://pointclouds.org/

OpenFrameworks – http://openframeworks.cc/