Dynamic loading of 3D models

PROJ-H-402 Project Report

Tim Lenertz, INFO MA1, ULB

February 24, 2014

Résumé

résumé en français

Abstract

english abstract

Contents

1 Introduction

Point clouds are a way of digitally representing three-dimensional models using only a set of points located on the object's surfaces[1]. Each point consists of 3 coordinates (x, y, z) on an Euclidian coordinate system defined for the model, and can be attributed with additional information such as color, surface normal vector, and others.

This data is typically produced by 3D scanners, which today can capture surfaces at a very high level of detail and thus yield huge quantities of points. Full representations of large objects or environments can be achieved by combining the outputs of multiple scans from different view points. For example to get a full point cloud model of an archeological site, aerial scans may be combined with close-up scans of individual landmarks. The resulting files can easily reach several GB in size and contain over 10^8 points, and so they can no longer be processed or visualized efficiently as a whole. Instead, subsets of points around an area of interest are extracted.

The goal of this project is to develop a system which dynamically loads visible subsets of points to render while a user moves through a large scale point cloud. This involves methods to extract the right subsets from the point cloud, the data structure which the full point cloud is stored in, and the file format using which this data structure is serialized on secondary storage. The process should appear seamless to the user.

To this end, different data structures and file formats were compared for their possibilities to extract the right subsets of points in an efficient way, and a program was developped which converts a given point cloud file into such a data structure (preprocessing stage), and then allows the user to explore the point cloud, by dynamically loading chunks of it from the preprocessed data structure file.

[1] Only non-volumetric point clouds are considered for this project. In volumetric point clouds, points are not only located on surfaces, but on the insides of objects as well.

2 Filtering the point cloud

This chapter describes the methods used to compute a smaller set of points based on the full point cloud, which visually represent the model as seen from a given view point. The data structure used to store the point cloud is not considered in this chapter.

2.1 Definitions

The following definitions are used throughout this report. A *point cloud* is an unordered set of points with an Euclidian coordinate system. Each *point* $p = \langle x, y, z, r, g, b \rangle$ consists of its coordinates x, y, z, and RGB color information. The *model* P is the full point cloud used as input. The *point capacity* C is the maximal number of points that can be outputted to the renderer.

The *view-projection matrix* $\boldsymbol{M} = \boldsymbol{V} \times \boldsymbol{P}$ is a 4x4 matrix that defines the view frustum of the camera. The 6 planes of the frustum can be derived from the matrix as described in [1]. The *view matrix* $\boldsymbol{V}$ transforms the points' coordinate system into one centered around the camera at its current position (x_c, y_c, z_c) and orientation $(\alpha_c, \beta_c, \gamma_c)$, while the *projection matrix* $\boldsymbol{P}$ is used to project the points to their two-dimensional screen coordinates. $\boldsymbol{P}$ might define both a parallel projection or a perspective projection with a given *field of view* λ.

The *filtering function* $f_P(\boldsymbol{M})$ computes a set of rendered points P' from and model P, the matrix $\boldsymbol{M}$, and additional parameters. Its main constraint is that $|P'| \leq C$ (whereas $|P|$ may be much larger than C). P' does not need to be a subset of P: Some methods (such as uniform downsampling) will add points into P' that are not in P, in order to achieve a better visual quality.

The criteria for quality of the filtering function is that the 2D projection of P' at the current view point $\boldsymbol{M}$ looks similar to that of P, that is there should be no loss of important details and no obvious visual artifacts or discontinuities. Techniques such as hidden surface removal can actually improve the appearance of P' compared to that of P.

The function f_P described in this chapter is an idealized version that operates on a set of points. The next chapters describe algorithms that calculate an approximation of $f_P(\boldsymbol{M})$ using a specific data structure for P, and with additional time complexity constraints.

2.2 Projection

When the point cloud is rendered, the points p are projected from their three-dimensional virtual space onto the two-dimensional screen, using the view frustum defined by $\boldsymbol{M}$. This can be described as a function $\text{proj}_{\boldsymbol{M}}(x, y, z) = (x', y')$, where $x_S \in [0, w[$ and $y_S \in [0, h[$, with w and h being the width and height of the screen in pixels.

2.3 Frustum culling

2.4 Downsampling

2.5 Occlusion culling

3 Data structures

Bibliography

[1] Klaus Hartmann Gil Gribb. Fast extraction of viewing frustum planes from the world-view-projection matrix. 2001.