Evaluation of an Appearance-Preserving Mesh Simplification Scheme for Configura AB

Rasmus Hedin   `<rashe877@student.liu.se>`
Erik S. V. Jansson   `<erija578@student.liu.se>`

LI.U LINKÖPING UNIVERSITY

Contents

Contents

1   **Introduction**

For many years the field of computer graphics has been an important part in many industries, but especially in the entertainment industry (for instance video games and motion pictures). These industries generate a lot of money, and are quickly growing in size. A recent survey by *Kroon and Nilsson* [3] from *Dataspelsbranschen* have shown that the video games industry in Sweden generated €1325 M in revenue in 2016, a steep increase from the €392 M in 2012. Also, most movies nowadays use to some extent 3-D computer graphics in scenes where the cost would be to large to reproduce in reality, be too risky for actors, or simply be impossible.

The rendering of meshes (a collection of polygons describing a surface) is one of the main activities in computer graphics (usually, a collection of meshes; a so called scene description). In many cases, these meshes are very detailed, and require a large amount of polygons to fully describe a surface. This is problematic, since the rendering time of a scene depends on the number of polygons it has. Therefore, it is important to reduce the number of polygons in a mesh as much as possible. This is especially true in video games, where the scene needs to be rendered in real-time. However, if the number of polygons are reduced too much, it will degrade the visual quality of a mesh, giving a progressively flatter surface than intended and removing small surface details. This destroys the intended *geometrical appearance* of the mesh.

1.1   Motivation

While the geometrical appearance of a mesh is important, it is not the only factor which gives the final appearance of a mesh when rendering. According to *Cohen et al.* [1], both the surface curvature and color are equally as important contributors. *Textured appearance* will be used as the common name for these since surface properties are usually specified with a texture map.

In computer graphics, the process to reduce the number of polygons in a mesh based on some metric is called a *mesh simplification algorithm*, as seen in *Talton's survey* [5] on the subject. Historically, these have been mostly concerned with minimizing the geometrical deviation of a mesh when applying it. Somewhat recently, methods for minimizing the texture deviation when simplifying a mesh have also appeared. They attempt to reduce the texture deviation and stretching caused when removing polygons from a mesh, as described in *Hoppe et al.* [2].

By simultaneously taking into account the geometrical and texture deviation, one can preserve the *visual appearance* of a mesh when simplifying it. If polygons can be removed without affecting this appearance significantly, the rendering time can be reduced for "free".

1.2 Aim

To survey the field for state-of-the-art mesh simplification algorithms that preserve the visual appearance of a mesh, and integrate these into *Configura's* (see Section 1.5) graphics pipeline. This will enable Configura to generate better *Level of Detail* (LoD) meshes for speeding up their rendering time. Currently, Configura only takes into account the geometrical deviation when simplifying, with no regard for the textures (e.g. diffuse or normal) on top of the mesh.

Thereafter, we plan to evaluate each of these solutions by measuring their performance and ability to preserve the meshes' original appearance. In the end, the goal is to find the mesh simplification algorithm which both performs and preserves the mesh appearance well.

1.3 Research Questions

After implementing and measuring the performance of these mesh simplification algorithms, answers to the questions below should have been obtained. These will be used to decide on a suitable alternative for Configura and also other systems with a similar set of requirements.

1. What alternative *mesh simplification schemes* exist that *preserves the appearance* of a mesh?

2. Which of these alternatives give the best *performance* and *appearance preservation*? When:

    a) Measuring the algorithm's *computation time* while targeting an *appearance threshold*?

    b) Measuring the algorithm's *memory usage* while targeting some *appearance threshold*?

    c) Measuring the real-time *rendering time* of the LoD-hierarchy of the *simplified mesh*? (by using the meshes generated according to the target *appearance threshold* above)

3. Which of the alternatives gives the best *appearance preservation* for a target *polygon count*?

1.4 Delimitations

Since there are many mesh simplification algorithms in previous work, a proper literature review would have to be done to find possible candidate solutions. Since our thesis' goals are mostly concerned with implementing and evaluating each solution, we have decided to base our choices on existing surveys and literature reviews to skip doing some of these ourselves.
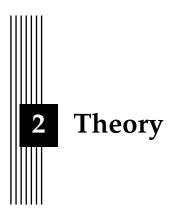
Also, since implementing and doing measurements on all algorithms would take too long, we have decided to only pick an interesting subset of the algorithms presented in the surveys. More precisely, we have chosen to pick three different mesh simplification algorithms, one that does not take texture deviation into account and two that do. In addition, we will also compare them to Configura's existing mesh simplifier scheme; for a total of four algorithms.

1.5 Background

This thesis was requested by Configura AB, a company in Linköping which provides space planning software. Their main product, *CET designer*, lets companies plan, create and render 3-D office spaces (among other things). These scenes can have a large amount of polygons that need to be rendered in real-time for customers to evaluate their creations in CET designer.

To allow larger scenes to be rendered with higher frame-rates (e.g. needed when exploring environments in *Virtual Reality* (VR), to prevent motion sickness), it would be beneficial to reduce the amount of polygons as much as possible. The meshes in these scenes usually have textures applied to them, and it is therefore important to keep the quality as high as possible.

While Configura already has a mesh simplification in their pipeline, it only accounts for surface simplifications, and doesn not take into account the texture appearance that might be degraded when applying mesh simplification. Hence, the given task was to integrate a new mesh simplification scheme that takes into account texture quality when simplifying a mesh.

2 Theory

Since several mesh simplification algorithms are being considered, Section 2.1 presents the most notable schemes found (through peer-reviewed surveys) in previous work. An outline of the algorithm and the results found by authors are given for the reader's convenience, and also to be used as a guideline when implementing the solutions into Configura's CG pipeline.

Afterwards, in Section 2.2, we discuss the different metrics that can be used to measure the appearance preservation after a simplification has been done. This will later be used to evaluate the solutions and provide an empirical way to answer research questions 2 and 3 by giving a concrete metric for appearance thresholds and the amount of appearance deviation.

Finally, in Section 2.3, the methods and common practices for measuring performance of an algorithm are discussed. Based on existing industry practices, we show how to measure the computation time and memory usage of the algorithms. Since these measurements can be noisy, statistical methods will need to be used to truthfully answer our research questions.

2.1 Mesh Simplification

According to *David Leubke's survey* [4] on the subject, mesh simplification is a technique which transforms a geometrically complex mesh (with a lot of polygons) to a simpler representation by removing unimportant geometrical details (reducing the number of polygons). It does this by assuming that some meshes are small, distant, or have areas which are unimportant to the visual quality of the rendered image. For example, if the camera in a scene always faces towards a certain direction, then removing details from the backside of a mesh won't affect the final rendered result, since they will never be seen by the camera anyway.

Quadric-Based Error Metric

Appearance-Preserving Simplification

In order to preserve the appearance of a model when it is simplified, *Cohen et. al* [1] defines a new *texture deviation metric*. This metric takes three attributes into account: Surface position, surface curvature, and surface color. To properly sample these attributes from the input surface, the surface position is decoupled from the color and normals stored in texture and

3

normal maps, respecticely. The metric guarantees that the maps will not shift more than a user-specified number of pixels on the screen ϵ.

Approximation of the surface position is done offline with simplification operations such as edge collapsing and vertex removals. At run-time, the color and normals are sampled in pixel-coordinated with mip-mapping techniques. Here the decoupled representation is useful since the texture deviation metric can be used to bound how much the mapped attributes value's positions deviate from the positions of the original mesh. This gurantees that the sampling and mapping to screen-space of the attributes is done in an appropriate way.

Before any simplification can be made, a parametrization of the surface is required in order to store the color and normals in maps. If the input mesh does not have a parametrization, it is created and stored per-vertex in texture and normal maps. Next, the surface and maps are fed into the simplification algorithm which decides which simplification operations to use and in what order. The deviation caused by each operation is measured with the texture deviation metric. A *progressive mesh* (PM) with error bounds for each operation is returned by the algorithm which can then be used to create a set of LOD with error bounds. Using the error bounds, the tesselation of the model can be adjusted to meet the user-specified error ϵ.

Texture Mapped Progressive Meshing

Given an arbitrary mesh, *Sander et. al* [**sander2001texture**] presents a method to construct a PM where a texture parametrization is shared between all meshes in a PM sequence. In order to create a texture mapping for a simplified mesh, the original mesh's attributes, e.g normals, is sampled. This method was developed with two goals taken into consideration:

- Minimize *texture stretch*: When a mesh is simplified the texture may be stretched in some areas which decrease the quality of the appearance. Since the texture parametrization determines the sampling density, a balanced parametrization is prefered over one that samples with different density in different areas. The balanced parametrization is obtained by minimizing the largest texture stretch over all points in the domain. No point in the domain will therefore not be too stretched and thus making no point undersampled.

- Minimize *texture deviation*: Conventional methods use geometric error for the mesh simplification. According to the authors this is not appropriate when a mesh is textured. The stricter texture deviation error metric, where the geometric error is measured according to the parametrization, is more appropriate. This is the metric by *Cohen et. al* [1] explained in 2.1. By plotting a graph of the texture deviation vs the number of faces, the goal is to minimize the height of this graph.

Cohen et. al [1] stored an error bound for each vertex in a PM. *Sander et. al* [**sander2001texture**] instead tries to find an atlas parametrization that minimizes both texture deviation and texture stretch for all meshes in the PM.

2.2 Metrics for Appearance Preservation

Previously 2.12.1, the metrics texture deviation and texture stretch have been defined. But to measure more exactly how much the visual appearance of a simplified mesh deviate from the original mesh another metric would be better. *Lindstrom and Turk* [**lindstrom2000image**] defines *Image-driven simplification* which captures images from different angles of the mesh. The difference between the images of the original and simplified mesh are computed in order to measure how well the appearance is preserved. This metric is more general and can be applied to all meshes.

The *image metric* is defined as a function taking two images and gives the distance between them. To measure the distance the authors use root mean square of the luminance values of
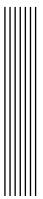
two images Y^0 and Y^1 with dimensions $m \times n$ pixels. It is defined as:

$$d_{RMS}(Y^0, Y^1) = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (y_{ij}^0 - y_{ij}^1)^2}$$

To evaluate the quality of the simplified mesh the authors capture images from 24 different camera positions. The positions are defined as the vertices of a rhombicuboctahedron. Two sets of l images $Y^0 = Y_h^0$ and $Y^1 = Y_h^1$ with dimensions $m \times n$ is rendered and the RMS is then computed as:

$$d_{RMS}(Y^0, Y^1) = \sqrt{\frac{1}{lmn} \sum_{h=1}^{l} \sum_{i=1}^{m} \sum_{j=1}^{n} (y_{ij}^0 - y_{ij}^1)^2}$$

2.3 Measuring Algorithmic Performance

Bibliography

[1] Jonathan Cohen, Marc Olano, and Dinesh Manocha. "Appearance-preserving simplification". In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM. 1998, pp. 115–122.

[2] Hugues Hoppe. "Progressive meshes". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 99–108.

[3] Jacob Kroon and Anna Nilsson. "Game developer index 2017". In: *Based on 2017 Annual Reports. Dataspelbranchen* (2017).

[4] David P Luebke. "A developer's survey of polygonal simplification algorithms". In: *IEEE Computer Graphics and Applications* 21.3 (2001), pp. 24–35.

[5] Jerry O Talton. "A short survey of mesh simplification algorithms". In: *University of Illinois at Urbana-Champaign* (2004).