Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Skeleton to Quad Dominant Mesh

Master Thesis

2013

Michal Piovarči

Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Skeleton to Quad Dominant Mesh

Master Thesis

Study programme:	Computer Science
Study field:	9.2.1 Computer Science, informatics
Department:	Department of Computer Science
Supervisor:	RNDr. Martin Madaras

Bratislava, 2013

Michal Piovarči

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:
Study programme:

Field of Study:
Type of Thesis:
Language of Thesis:

Title:

Aim:

Supervisor:
Department:

Assigned:

Approved:

Guarantor of Study Programme

.....
Student

.....
Supervisor

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:
Študijný program:

Študijný odbor:

Typ záverečnej práce:
Jazyk záverečnej práce:

Názov:

Cieľ:

Vedúci:
Katedra:
Vedúci katedry:

Dátum zadania:

Dátum schválenia:

garant študijného programu

.....

študent

.....

vedúci práce

Acknowledgement

I would like to thank my supervisor RNDr. Martin Madaras for his help and advices.

Abstract

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Keywords: lorem, ipsum, consectetur

Abstrakt

Lorem Ipsum je fiktívny text, používaný pri návrhu tlačovín a typografie. Lorem Ipsum je štandardným výplňovým textom už od 16. storočia, keď neznámy tlačiar zobral sadzobnicu plnú tlačových znakov a pomiešal ich, aby tak vytvoril vzorkovú knihu. Prežil nielen päť storočí, ale aj skok do elektronickej sadzby, a pritom zostal v podstate nezmenený. Spopularizovaný bol v 60-tych rokoch 20. storočia, vydaním hárkov Letraset, ktoré obsahovali pasáže Lorem Ipsum, a neskôr aj publikačným softvérom ako Aldus PageMaker, ktorý obsahoval verzie Lorem Ipsum.

Kľúčové slová: lorem, ipsum, consectetur

Preamble

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc tristique, sem et feugiat ornare, lorem eros mattis odio, et tempus lectus ipsum nec ante. Phasellus interdum nunc ut sapien semper porttitor. Nam mi erat, faucibus in fermentum eu, varius eu velit. Integer egestas iaculis varius. In pulvinar, ligula eget adipiscing suscipit, nisl ipsum aliquet arcu, eget tristique felis leo vitae magna. Nulla et magna sed justo accumsan ultrices a in leo. Suspendisse tincidunt malesuada leo, eget rhoncus ipsum fringilla at. Integer et tortor vitae nisl fermentum vestibulum. Fusce eu dui neque, a egestas nunc. Vivamus condimentum mi non arcu lacinia et aliquam risus euismod. Nunc ut risus nec elit luctus aliquet et sit amet magna. Vestibulum vehicula enim eget erat fermentum a lacinia purus varius.

Duis tempus sem sit amet elit accumsan ultricies. Curabitur a nibh ante, vitae pharetra nulla. Suspendisse non risus elit, in aliquam felis. Maecenas suscipit placerat commodo. Vivamus et molestie odio. Quisque ut augue mi. Quisque aliquam luctus est, ac dignissim ante adipiscing eget. Quisque volutpat, sem vitae placerat condimentum, nunc lorem malesuada leo, sit amet pretium nisi felis nec lorem. Pellentesque nisi ipsum, vestibulum sed lacinia sed, condimentum a turpis.

In posuere convallis lectus vel hendrerit. Cras suscipit mi risus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec ante nunc, cursus ac vulputate at, bibendum eget nisi. Nunc eget nunc sed massa blandit posuere id vel quam. Duis bibendum orci vel ligula tempor condimentum. Nulla pharetra tortor at risus dignissim fringilla. Nullam ac massa et nibh auctor vestibulum quis vitae ligula. Suspendisse ultrices eros sit amet lectus dictum dapibus. Sed congue, turpis nec aliquam fermentum, diam nisi cursus nibh, id vulputate massa tellus sit amet turpis.

Contents

List of Figures

List of Tables

Introduction

Chapter 1

Implementation

The main parts of the Skeleton to Quad Dominant Mesh algorithm, were discussed in previous chapter. Now we are going to describe each part in greater detail as well as our enhancements.

1.1 Skeleton Straightening

Skeleton straightening is a preprocessing which serves to simplify the third phase of Skeleton to Quad Dominant Mesh algorithm, that is joining of Branch Node Polyhedrons(BNP). Joining straight lines is easier than joining of arbitrary poly-lines. After this step each node is collinear with its parent node. In practice we rotate child nodes. The angle and axis of rotation are determined from the angle between two vectors **from** and **to**. **From** vector is the vector from parent to child node and **to** vector is the vector from parents parent to parent node. After the rotation of a BNP node all his child nodes are also rotated. This step serves to preserve the original skeleton structure.

During this step of preprocessing we also create and store skinning matrices. This matrices are used in last step of the algorithm named final vertex placement. This is an enhancement that allows us to finish the generated mesh in programmable shaders. Using skinning also allows us to prevent undesired mesh deformation that may occur with simple straightening.

Skinning matrices are computed from the original and straightened skeletons. Straightened skeleton represents the bind pose of the final generated mesh. Degree of freedom (DoF) of each node is calculated as the quaternion that is needed to rotate from bind pose

back to original skeleton. When we calculate DoFs in this way bind pose will always have zero values in its DoFs. So skinning matrices of bind pose will be identity matrices and we can omit them in calculation of subsequent skinning matrices.

1.2 BNP Generation

For each BNP we calculate the intersections of its associated sphere with paths that connect the BNP with its childes. We then triangulate the resulting points to create a mesh of the polyhedron. The triangulation is done using Delaunay triangulation in a spherical domain. We adopted the algorithm used in the original article.

Each BNP is then tessellated by inserting a vertex in the middle of each face and edge. We create a list of all faces that are not split yet and all edges that are already split. For each face we insert a vertex in its center and check its edges. We also insert a vertex in the middle of each edge that was not split yet. After this insertions the face is not a triangle. Next we connect each point of the face with the newly inserted central vertex so the face is triangulated. Finally we project all newly inserted vertices onto the sphere associated with current node. This serves to increase the volume of each node. Without it some BNPs will have little to no volume. For example BNPs created with only three intersection would be two faces stitched together with zero volume.

1.3 BNP Refinement

During the joining phase of the algorithm we want to join BNPs that are connected via a path. This connection is not possible to consist purely of quadrilaterals if the corresponding vertices have different valency (number of vertices in one-ring neighbourhood). Thus the purpose of BNP refinement is to ensure that each pair of connected vertices has equal valency. Link Intersection Edges (LIEs) is a set of edges which belong to the link of two intersection vertices. We increase the valency by splitting edges in a LIE. This split increases the valency of two intersection vertices simultaneously. We always split a representative edge and the we apply various smoothing schemes to equalize the lengths in a LIE. The smoothing schemes are: quaternion smoothing, neighbour avaraging smoothing, one-ring area weighted Laplacian smoothing and valency weighted Laplacian smoothing.

The algorithm loops through all BNPs. First we calculate the difference in valencies of each pair of vertices and mark it into a table. Intersection vertex with nodes parent is specially marked to not be subdivided more than is needed. Without this requirement a subdivision in child BNP could cause a need to subdivide parents BNP and thus create a possibly infinite loop.

Next we generate all LIEs of the BNP. For each LIE we store how many times it was refined, first and last vertex of LIE and quaternion describing the rotation from the first vertex of a LIE to its last vertex. This quaternion will be used to smooth inserted vertices if quaternion smoothing is enabled. We generate LIEs as follows. We loop through all intersection vertices. For each vertex we take an arbitrary vertex from its one-ring neighbourhood. We move backwards around the neighbourhood to find the first vertex of a LIE. For each vertex of one-ring neighbourhood we know its corresponding intersection vertices. If we fix two intersection vertices, then the first vertex of a LIE is the last vertex corresponding to these two intersection vertices while moving backwards. From the other side the last vertex of a LIE is the last vertex corresponding to the two selected intersection edges while moving forward.

In this fashion we can find the first vertex of the first LIE and after that we can construct all LIEs corresponding to a intersection vertex just by moving forward from the first vertex. When we find the first and last vertex of a LIE we try to construct quaternion representing rotation between them. We represent each of the points as a unit vector from corresponding node position and the vertex position. The axis of rotation is the cross product of these two vectors and the angle is their dot product. A problem may occur if the two vectors are linearly dependent. Then we can not decide the correct axis of rotation. In this case instead of using the first and last vertex of a LIE we use the first two vertices of a LIE to calculate the quaternion rotation. A LIE will always have at least two edges and three vertices due to the subdivision phase.

Generated LIEs are finally mapped to their corresponding intersection vertices, along with the number of splits required. Next we loop through all the intersection vertices. If the need to split is greater than zero, that is the valency of the intersection vertex is greater that the valency of its pair vertex. We then loop through all the LIEs of the intersection vertex and try

to find the best LIE to split. We pick the LIE witch has the biggest need to be split and was least refined. We prefer to pick a LIE that corresponds to a leaf node if there is no need to refine LIEs corresponding to other BNPs to avoid excessive subdivision. Intersection vertex corresponding to parent is picked first for splitting and after that it is forbidden to split it any further. This ensures that we wont force subdivision of parents BNP from its child.

1.3.1 Smoothing algorithms

Since we only subdivide the first edge of each LIE we need to smooth the resulting polyhedron. We have developed four smoothing methods: quaternion smoothing, neighbour avaraging smoothing, one-ring area weighted Laplacian smoothing and valency weighted Laplacian smoothing. In the end we selected quaternion smoothing as the most appropriate method, due to its speed and quality of output.

For quaternion smoothing we use the quaternions calculated during LIE generation phase. First and last point of each LIE are fixed so the angle and between them and their axis of rotation are also stable. We first count the number of vertices between first and last vertex of the LIE and then divide the angle by this number. Then we rotate the first point by a quaternion that is formed from the calculated angle and fixed axis of rotation. In the end the points are projected onto the sphere corresponding to current node. This method produces LIEs that lie on small circles of their corresponding sphere. The spacing between vertices is regular and thus is very suitable for our needs.

Averaging smoothing consists of averaging a vertex with its one-ring neighbourhood. We move from the last vertex of a LIE backwards towards the first vertex. We move each vertex of a LIE (except first and last) to the barycentre of its one-ring neighbourhood. This is a iterative approach that will potentially spread all vertices of a LIE evenly. We have found that one iteration of the algorithm is usually enough.

Laplacian smoothing consist of calculating standard Laplacian smoothing with different weight settings and then projecting smoothed nodes onto the sphere associated with current node. We used two weight settings. Valency weighted Laplacian used the valency of each vertex as the input weight. One-ring area weighted Laplacian used the one-ring area of each vertex as the input weight. Both of these methods produced similar results.

1.4 BNP Joining

All vertices that are connected via a path now have equal valency. For each intersection vertex we remove all faces of his one-ring neighbourhood. Then we select all the vertices of his one-ring neighbourhood and a ring of vertices that is parallel to them but is offset to the center of the next node. Then we can create quadrilaterals that will represent our mesh. We repeat this process for each node that is not a leaf or BNP node. In BNPs we instead of creating new set of vertices just connect with the vertices already existing in the BNP after we remove the unnecessary triangles. For leaf nodes we need to finish the mesh somehow. Various techniques can be used but the most useful for our purposes seem to be a ending in a point or in a capsule.

1.4.1 Point ending

Point ending is the simplest way of finishing the mesh. All the vertices of the last ring will simply form triangles with a vertex given by nodes position. Although relatively simple this approach offers the most control over the resulting mesh.

1.4.2 Capsule ending

Capsule ending means that leaf node will create a hemisphere on leaf nodes. There are various approaches to solve this problem. We could calculate the number of rings needed directly. But the algorithm is capable of creating tubular structures with varying radius. So the best approach seem to be to insert new nodes into the skeleton tree in a pre-process phase and let the algorithm create capsules on its own. This is very useful because we don't need to alter the algorithm itself and generated capsules will also work with tessellation.

After each capsule node we insert several new nodes into the skeleton. The number of nodes is depends on capsules radius. We have found that it is best that the number of new nodes is equal to the radius of the capsule, but to be at least five nodes. The inserted nodes are collinear with capsule node and its parent. They lie forward (direction from parent to capsule node position) from capsule nodes center. Inserted nodes are distributed according to a bezier curve that approximates the curvature of a unit sphere. for each node its radius is calculated according to formula:

$$newRadius = \sqrt{nodeRadius^2 * (1 - step^2)}$$

1.5 Final vertex placement

We now should apply the inverse of the rotation that we used to straighten the mesh to receive mesh in original pose. However a simple rotation could cause self intersections in the mesh and thus create non manifold meshes. In order to avoid this we used skinning to transform the vertices. The extraction of skinning matrices has been described in section 1.1. We use linear matrix combination. This method proves sufficient for our needs but other more sophisticated methods as quaternions or dual quaternions could be used.

Chapter 2

Example

2.1 Tables

In this section you can see example of tables.

1	2	3
4	5	6
7	8	9

Table 2.1: Numbers

And another one

A	B	C
D	E	F
G	H	I

Table 2.2: Letters

2.2 Figures

In this section you can see example of figures.

Figure 2.1: Johann Amos Comenius

2.3 Cross reference

In this chapter we used table 2.1 with numbers and table 2.2 with letters on page 8. Also, we used figure 2.1 with Johann Amos Comenius on page 9.

2.4 Citation

Appendix A

TEX

LATEX, TEX

Bibliography