


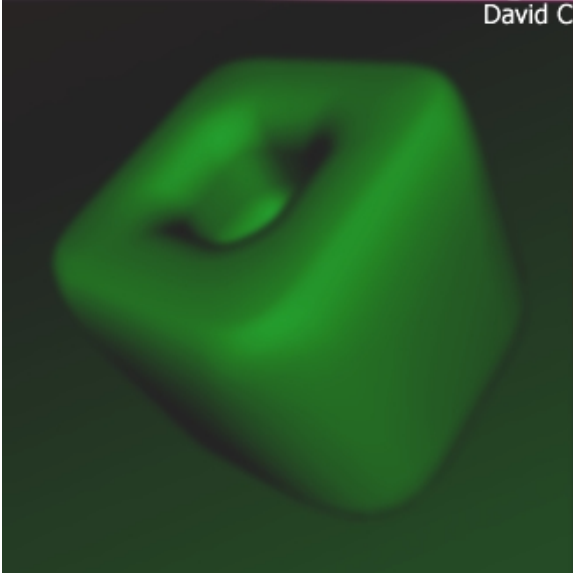


**JUST IN TIME**

**VRML97**

**VOLUME ROUNDING** with  
**POLYHEDRON BEVELLING**

David C. W. Poon



A thesis submitted in partial fulfilment of  
the requirements for the degree of  
Master of Science in Computer Science  
University of Auckland, 1998

Polyhedra are widely used as scene components in computer graphics. However, a problem with true polyhedra, as compared to real-world polyhedral objects, is that they have perfectly sharp edges and corners. A solution is to round the edges and corners to produce a softer more natural look. This thesis presents *Polyhedron Beveling*, a new fast polygon-based volume rounding method with a simple roundness control.

The thesis begins with a user survey of how people perceive *roundness*, and suggests requirements for an automatic volume rounding method. Two variants of polyhedron beveling are developed, both based on recursive mesh subdivision techniques. The first method introduces a partitioning scheme to Catmull-Clark/Doo-Sabin subdivision to indirectly control the roundness of the resultant shape. Although the method generates analytically-smooth surfaces, it suffers from several construction limitations and stability problems if the desired roundness is large. The second method makes use of straight skeletons, modifying the actual recursive mesh subdivision to accomplish more directly the task of preserving planar regions. The result is a more stable algorithm. Both methods produce excellent results on a wide range of input meshes both in terms of execution speed and resultant quality.

The final contribution of the thesis is the development of a Just-In-Time (JIT) VRML volume rounding extension using the polyhedron beveling method. With this extension to VRML, rounded scene objects can be described as an initial coarse object representation together with a rounding specification. Polyhedron beveling is executed just before rendering in real time. The file size required for this JIT approach is typically only 1 - 2% of the size of a standard polygon-mesh representation of the rounded object. The compactness of the design is a great advantage as VRML is designed to work on the Internet, while today's scarce bandwidth severely constrains the feasible scene complexity.

# Acknowledgements

---

Someone once mentioned writing a thesis is both a physical and a mental torture. I agree totally. The satisfaction it brings, however, is also tremendous. I am delighted with the work I have done and there are, of course, many people that I wish to acknowledge. In particular, I should like to thank my supervisor Dr Richard Lobb for his intellectual stimulation and guidance. Congratulations also on his recently achieved teaching award. If this is not enough to prove how good he is, I should say, without his advice and encouragement, I beg I cannot get anywhere near to what I have achieved.

I should also like to acknowledge the assistance by Associate Professor Don Sheridan, the director of the Management Teaching and Technologies Unit, and also my boss. Not to mention his various helps like arranging the colour printing and his permission for conducting the survey on human recognition of roundness using CECIL. I am deeply in debt because of his consideration and understanding by allowing me to adjust my workload during the time I conduct this research.

Then, it is my girlfriend, Molly, and my family. Their loves, supports, and many other qualities that I cannot describe in full are truly appreciated.

Finally I should like to thank you all my friends for participating in the survey and various other aspects of the research.

# Table of Contents

---

TABLE OF CONTENTS	VII
LIST OF FIGURES	X
CHAPTER 1 INTRODUCTION	1
1.1 COMPUTER GRAPHICS MODELLING .....	1
1.2 THREE-DIMENSIONAL VOLUME ROUNDING .....	2
1.3 VIRTUAL REALITY MODELLING LANGUAGE .....	3
1.4 JUST-IN-TIME VRML VOLUME ROUNDING .....	3
1.5 GOALS AND CONTRIBUTIONS .....	4
1.6 ORGANISATION OF THE THESIS .....	5
CHAPTER 2 MEANING OF VOLUME ROUNDING	7
2.1 THREE-DIMENSIONAL VOLUME ROUNDING: WHAT DOES IT MEAN? .....	7
2.2 QUANTIFYING ROUNDNESS .....	7
2.2.1 ROUNDNESS VS. SMOOTHNESS .....	8
2.2.2 INTERPRETATION OF CURVATURE PLOT .....	8
2.3 CONSTRAINTS ON THE ROUNDING OPERATION .....	10
2.3.1 PRESERVATION OF TOPOLOGY .....	10
2.3.2 PRESERVATION OF PLANAR SURFACE .....	11
2.3.3 MONOTONIC REDUCTION OF PLANAR REGIONS .....	11
2.3.4 SHRINKING FROM BOUNDARY .....	12
2.4 DEGREE OF ROUNDNESS AND LIMIT BEHAVIOUR .....	13
2.4.1 WELL-DEFINED LIMIT .....	13
2.4.2 CONTINUITY IN CONTROLLABLE ROUNDNESS .....	13
2.5 DIFFERENT WAYS OF CONTROLLING THE ROUNDED REGIONS .....	14
2.6 SURVEY ON HUMAN PERCEPTION OF ROUNDNESS .....	14
2.6.1 AIMS AND DESIGN .....	15
2.6.2 SURVEY RESULT .....	17
2.6.3 OBSERVATIONS .....	18
2.7 QUALITY CRITERIA OF VOLUME ROUNDING .....	20
2.8 CHAPTER CONCLUSION AND SUMMARY .....	20
CHAPTER 3 EXISTING VOLUME ROUNDING METHODS	21
3.1 IMPLICIT BLENDING METHODS .....	21
3.1.1 CONVOLUTIONAL SMOOTHING .....	21
3.1.2 ROLLING-BALL BLENDS .....	22
3.2 PARAMETRIC BLENDING METHODS .....	22
3.3 RECURSIVE MESH SUBDIVISION METHODS .....	23
3.4 THE FULFILMENT OF QUALITY CRITERIA .....	24
3.5 CHAPTER CONCLUSION AND SUMMARY .....	26
CHAPTER 4 RECURSIVE MESH SUBDIVISION	27
4.1 UNIFORM B-SPLINE SURFACE REFINEMENT .....	27
4.2 CATMULL-CLARK REFINEMENT .....	30

4.3	SMOOTHNESS OF CATMULL-CLARK SURFACE AND EXTRAORDINARY POINTS.....	32
4.4	DOO-SABIN REFINEMENT.....	32
4.5	CHAPTER CONCLUSION AND SUMMARY.....	33
<b>CHAPTER 5</b>	<b>POLYHEDRON BEVELLING BY MESH PRE-PARTITIONING</b>	<b>35</b>
5.1	ARCHITECTURE.....	35
5.2	TWO-DIMENSIONAL ANALOGY.....	36
5.2.1	CHAIKIN REFINEMENT SCHEME.....	36
5.2.2	MESH PRE-PARTITIONING POLYHEDRON BEVELLING IN TWO DIMENSIONS.....	37
5.2.3	PROBLEM AND REMEDY IN TWO DIMENSIONS.....	39
5.3	LOCALITY, CENTROID INTERPOLATION AND THE CONVEX HULL PROPERTY.....	40
5.4	SIMPLE MESH PRE-PARTITIONING POLYHEDRON BEVELLING.....	41
5.5	JUSTIFICATION OF THE PRE-PARTITIONING SCHEME.....	44
5.6	MAPPING FROM THE ROUNDNESS CONTROL VALUE TO THE ROUNDEDNESS.....	44
5.7	RESULTS.....	46
5.8	CHAPTER CONCLUSION AND DISCUSSION.....	46
<b>CHAPTER 6</b>	<b>OPTIMISATION METHODS FOR MESH PRE-PARTITIONING POLYHEDRON BEVELLING</b>	<b>49</b>
6.1	OPTIMISATION BY EXTRACTING PLANAR REGIONS.....	49
6.1.1	COPLANAR NEIGHBOUR REMOVAL.....	50
6.1.2	PLANAR REGION FITTING.....	50
6.1.3	RESULT OF THE OPTIMISATION BY EXTRACTING PLANAR REGIONS.....	51
6.2	OPTIMISATION BY CORNERS SUBDIVISION AND RECONSTRUCTION.....	52
6.2.1	ARCHITECTURE OF CORNERS SUBDIVISION AND RECONSTRUCTION.....	52
6.2.2	VOLUME RECONSTRUCTION OF CORNERS SUBDIVISION AND RECONSTRUCTION.....	53
6.2.3	RESULT OF CORNERS SUBDIVISION AND RECONSTRUCTION.....	55
6.3	CHAPTER SUMMARY AND CONCLUSION.....	56
<b>CHAPTER 7</b>	<b>POLYHEDRON BEVELLING BY TARGET-DRIVEN SUBDIVISION</b>	<b>57</b>
7.1	RATIONALE.....	57
7.2	TWO-DIMENSIONAL ANALOGY – A NEW CONCEPT OF “CENTROIDS”.....	58
7.2.1	MESH SUBDIVISION AS MOVING VERTICES TOWARDS CENTROIDS.....	58
7.2.2	ROUNDING BY MOVING VERTICES TOWARDS TARGET POINTS.....	59
7.3	TREATMENT IN THREE-DIMENSIONS.....	61
7.4	DEFINING THE TARGET FUNCTION - THE SKELETAL TARGET.....	62
7.4.1	BACKGROUND ON SKELETON.....	63
7.4.2	INTRODUCTION TO STRAIGHT SKELETON.....	63
7.4.3	FUNCTIONAL DEFINITION OF TARGETS USING THE STRAIGHT SKELETON.....	64
7.5	RESULTS AND DISCUSSION.....	66
7.6	COMPARISON WITH THE MESH PRE-PARTITIONING APPROACH.....	68
7.7	CHAPTER SUMMARY AND CONCLUSION.....	69
<b>CHAPTER 8</b>	<b>THE JUST-IN-TIME IMPLEMENTATION</b>	<b>71</b>
8.1	SCOPE AND OBJECTIVE.....	71
8.2	EXTENSIBILITY OF VRML.....	72
8.2.1	PROTOTYPES.....	72
8.2.2	SCRIPTS.....	73
8.3	THE LANGUAGE EXTENSION.....	73
8.4	IMPLEMENTATION AND ARCHITECTURE OF THE JUST-IN-TIME MODIFICATION.....	75
8.5	RESULT.....	77
8.6	CHAPTER SUMMARY AND CONCLUSION.....	78

<b>CHAPTER 9</b>	<b>CONCLUSIONS</b>	<b>79</b>
9.1	<i>RESEARCH OVERVIEW</i> .....	79
9.2	<i>LIMITATIONS AND DIRECTIONS OF FUTURE RESEARCH</i> .....	80
<b>APPENDIX A</b>	<b>MATHEMATICAL PROOFS</b>	<b>81</b>
A.1	<i>PROOF OF LINEAR BOUNDARY PROPERTY</i> .....	81
A.2	<i>MAPPING <math>\rho</math> TO <math>\alpha</math> FOR REGULAR <math>N</math>-GONS</i> .....	85
<b>APPENDIX B</b>	<b>IMPLEMENTATION OF STRAIGHT SKELETON</b>	<b>87</b>
<b>BIBLIOGRAPHY</b>		<b>89</b>

# List of Figures

Figure 2.1. Normal section.....	9
Figure 2.2. Curvature plots of a sharp edge and a rounded edge.....	9
Figure 2.2. The $\epsilon$ -boundary planar regions of a rounded cube.....	12
Figure 2.3. Four possible ways of controlling the region of roundness.....	14
Figure 2.4. Images used in the survey on human perception of roundness.....	16
Figure 2.5. Survey results.....	19
Figure 3.1. Convolutional smoothing of a polygon by a circular filter.....	21
Figure 3.2. Rolling-ball blend defined on the same polygon as in figure 3.1.....	22
Figure 3.3. Examples of blending surface of N-sided vertex where (a) N=3 (b) N=5 (c) N=6.....	23
Figure 3.4. Fulfillment of the quality criteria for existing volume rounding methods.....	25
Figure 4.1. (a) A quadratic B-spline patch with its control mesh (b) The control mesh after a step of refinement (c) The four sub-patches generate B-spline surfaces. Each of them is equivalent to a quarter of the B-spline surface generated from the initial control mesh.....	28
Figure 4.2. Three different types of faces in quadratic Catmull-Clark refinement.....	31
Figure 4.3. Catmull-Clark (left) and Doo-Sabin (right) quadratic surface defined with a unit cube control mesh.....	33
Figure 5.1. Architecture of mesh pre-partitioning polyhedron bevelling.....	35
Figure 5.2. The Chaikin refinement scheme. (a) The control polygon and the resultant quadratic b-spline curve (b) New control points generated on the first iteration (c) Control polygons from the first two iterations superimposed on the original control polygon.....	37
Figure 5.3. Mesh pre-partitioning polyhedron bevelling in 2-dimensions. (a) Original polygon with the desired final straight-line segments marked red (b) Extra control points are added so that the endpoints of the final straight-line segments are at the midpoints of those corner segments (c) A quadratic b-spline curve defined on the pre-partitioned control polygon.....	37
Figure 5.4. Artefacts produced by mesh pre-partitioning polyhedron bevelling in two dimensions.....	39
Figure 5.5. Partitioning of a convex polygonal face.....	42
Figure 5.6. Problem with partitioning scheme using absolute displacement.....	42
Figure 5.7. The nomenclature for a pre-partitioned face.....	43
Figure 5.8. Different partitioning result in different roundedness.....	43
Figure 5.9. Using different $\alpha$ values on different faces.....	45
Figure 5.10. Five mesh pre-partitioning polyhedron bevelled cubes with different roundedness.....	46
Figure 5.11. The final rounded stapler with the original in the background.....	46
Figure 5.12. Other examples of mesh pre-partitioning polyhedron bevelled objects.....	46
Figure 6.1. The high fragmentation problem illustrated by a rounded cube with randomly coloured faces.....	49
Figure 6.2. Architecture of optimisation with planar regions extracted.....	49
Figure 6.3. A cube: (a) after mesh pre-partitioning with all the centroid faces removed (b) after mesh subdivision, both with randomly coloured faces to illustrate the partitioning.....	50
Figure 6.4. Resultant cube rounded with the optimised method by first extracting the planar regions.....	51
Figure 6.5. Resultant cube rounded by the optimised method that perform mesh subdivision only on corner faces.....	52
Figure 6.6. Immediate result of each step of the corners subdivision & reconstruction method.....	53

Figure 6.7. Topological information used in the corner subdivision and reconstruction method .....	54
Figure 6.8. The topological labelling on each curved edge .....	55
Figure 6.9. Five cubes rounded using the corners subdivision and reconstruction optimisation method with roundedness that normally are unachievable using the normal method .....	56
Figure 7.1. An example of artefacts produced on a input mesh with concave faces by the Doo-Sabin refinement scheme .....	57
Figure 7.2. An example of partitioning of a concave face with one of the special partitioning scheme explored .....	58
Figure 7.3. The nomenclature for a step of Chaikin refinement .....	58
Figure 7.4. Target-driven polyhedron bevelling in 2-dimensions: (a) initial polygon and nomenclature (b) after one step of subdivision .....	60
Figure 7.5. Polyhedron bevelling by target-driven subdivision in 3D. (a) initial mesh with initial centroids defined. (b) the mesh after the first step of subdivision. (c) the E and V-targets are re-computed and the new control points are moving towards the corresponding destination points in the subsequent refinements .....	62
Figure 7.6. Medial axis transform .....	63
Figure 7.7. Straight skeleton .....	63
Figure 7.8. Joint points of straight skeleton .....	64
Figure 7.9. Preserved planar regions defined using the straight skeleton (upper row) and mesh pre-partitioning (lower row) .....	65
Figure 7.10. Problem of the initial skeletal target .....	65
Figure 7.11. The initial and the first iteration mesh of target-driven polyhedron bevelling using skeletal targets .....	66
Figure 7.11. Examples of target-driven polyhedron bevelling using skeletal targets .....	66
Figure 7.12. Fragmentation of a target-driven polyhedron bevelled object .....	67
Figure 7.13. Staplers rounded with the two different polyhedron bevelling methods .....	67
Figure 8.1. Conceptual model of the JIT modification .....	71
Figure 8.2. A portion of the metamodel of VRML using the notation of the Unified Modelling Language [FS97] .....	72
Figure 8.3. VRML code for a rounded scene object and its corresponding non-rounded version .....	75
Figure 8.4. The integration of JIT VRML plug-in and VRML browser .....	76
Figure 8.5. Network transparency of the JIT modification .....	76
Figure 8.6. The prototype definition of the BevelledPolyhedron node type .....	77
Figure 8.7. A rounded stapler in VRML browser .....	78
Figure A.1. Linearity of centroids of the three consecutive boundary faces .....	81
Figure A.2. Partitioning of a triangular face .....	82
Figure A.3. Partitioning of a quadrilateral face .....	82
Figure A.4. Linearity of the boundary segment between two consecutive boundary centroids .....	83
Figure A.5. The nomenclature used in deriving the roundness control from the roundedness .....	85
Figure B.1. A divide-and-conquer method for computing straight skeleton .....	87
Figure B.2. Angle-bisecting vectors .....	88
Figure B.3. Calculating hit time between an angle-bisecting vector and a wavefront .....	88





---

## Chapter 1 Introduction

Three-dimensional volume rounding is the process of replacing sharp edges and corners with rounded surfaces for computer graphics scene modelling. This research presents a new class of fast volume rounding methods called *Polyhedron Beveling*. Our discussion starts with a general overview on computer graphics modelling.

### 1.1 Computer Graphics Modelling

The trends in computer graphics are constantly changing, but creating and rendering three-dimensional objects remain unarguably fundamental. The three-dimensional object representation influences both the creation and rendering. Hence its design is driven by the demands from both sides. Depending upon the target application, on one hand, the modelling format should be primitive, direct, and efficient enough for fast rendering. On the other hand, the modelling format must be rich, abstract and flexible enough for easy creation and expression of three-dimensional objects. This application-oriented nature leads to a wide range of modelling formats forming a continuous spectrum from richness to run-time efficiency.

Despite the diversity of modelling formats, most fall into one of two categories. They are either surface-based techniques or solid-based techniques. As the names already imply, surface-based techniques concern the modelling of merely the surface of the objects and ignore the fact that the objects are solid in nature. Solid-based techniques, of course, are the opposite. They are concerned with spaces inside or outside of an object. We will cover some of the most common representations below. The emphasis here will be mainly on surface-based techniques.

Surface-based techniques are usually at the run-time efficiency end of the modelling spectrum. For extreme efficiency, the surfaces of the objects are approximated by a set of polygons or even only triangles. This representation is called a polygon mesh or polyhedron. A polygon mesh is the most commonly used general-purpose modelling format, largely because of the existence of fast and efficient polygon renderers, software libraries and standards.

One obvious fault of a polygon mesh is its inability to represent curved surfaces. Piecewise linear approximation by polygon meshes offers a finite degree of accuracy that cannot be altered after creation. What is even worse is that the size of the representation grows rapidly as the required accuracy increases.

With interpolated or smooth shading techniques like Gourand shading or Phong shading [FV96], a smooth appearance on a curved surface can be obtained with a relatively rough polyhedral approximation. However, interpolated shading is not a perfect solution – for example, the silhouette edge of the polygon mesh is still obviously polygonal. As a result, a large quantity of vertex and normal information is still required for a convincing level of realism.

Higher-degree surface polynomials can be used for a better approximation of curved surfaces with comparatively less storage cost. Quadratic and cubic surfaces are common. Such polynomials are usually given in parametric form. Common examples of parametric surfaces are Bézier and B-spline [FV96] surfaces.

Implicit surfaces lie on the boundary between surface-based and solid-based modelling. Surfaces are defined implicitly as all points  $p$  in the 3-dimensional Euclidean space such that an implicit equation  $f(p) = 0$  is satisfied. Implicit surfaces often use the sign of  $f(p)$  to classify space as inside or outside of an object.

Constructive Solid Geometry (CSG) is a common example of a solid-based modelling technique. CSG objects are defined as combinations of simple primitives by means of boolean set operations such as set union and intersection. CSG primitives are usually simple geometric shapes including spheres, cylinders, cubes and, sometimes, half-spaces.

Other examples of solid-based modelling techniques include primitive instancing, sweeps and spatial partitioning. Readers interested in a survey of solid-based modelling can refer to Foley, van Dam et al [FV96 pp. 533-563].

## 1.2 Three-dimensional Volume Rounding

Although real-world scene objects always have at least slightly rounded corners and edges, such fine details are usually ignored in computer graphics scene modelling. Human eyes however can easily notice the lack of highlights and shadows resulting from the absence of the small curvature.

Modelling such details produces high realism, but at the same time greatly increases the required modelling time. The usual manual approach, using a Computer Aided Design (CAD) package, is to trim every planar surface back a little from the edges and then fit smooth surfaces to connect the adjacent planar faces. This process is usually referred to as blending. The CAD system then produces a polygonisation of the resultant shape. However, the process is still tedious, time-consuming and requires a fairly high level of expertise. Automation is clearly sought.

Three-dimensional volume rounding is concerning with automatically rounding sharp edges and corners, given an initial coarse object representation and a rounding specification. Much research effort has been put into different volume rounding techniques, which will be briefly reviewed in Chapter 3.

This thesis introduces a new class of fast volume rounding methods – *Polyhedron Beveling*. While most of the existing volume-rounding methods emphasise on the quality of the resultant shape or the complex control of roundness, Polyhedron beveling, on the other hand, differentiates itself by its fast execution speed and the real time usage.

### 1.3 Virtual Reality Modelling Language

This section briefly introduces the history and nature of our target implementation modelling language – Virtual Reality Modelling Language (VRML). The term *modelling language* generally refers to the file format used for describing computer graphics scene model. VRML is one of the popular scene description languages used to describe both static or animated computer graphics scenes. It is designed for use mainly on the Internet.

VRML was conceived in the spring of 1994 at the first annual World Wide Web (WWW) Conference. The term *Virtual Reality Markup Language* was coined at that time. The word 'Markup' was later changed to 'Modelling' to reflect the graphical nature of VRML. VRML is quickly being adopted as a standard for describing interactive three-dimensional scene objects and virtual worlds. VRML is also intended to be a universal interchange format for integrated three-dimensional graphics and multimedia. VRML 2.0, the second release of VRML, added significantly more interactive and animation capabilities. In December 1997, VRML97 replaced VRML 2.0 as the formal International Standard. See [VRML97].

VRML files are usually located on a remote server. The file gets transferred across the network and rendered locally with an VRML-enabled browser only on users' requests. Downloading speed is hence one of the major concerns in VRML. These factors severely constrain the scene complexity that is feasible with today's scarce bandwidth.

### 1.4 Just-In-Time VRML Volume Rounding

Although solid-based modelling is gaining popularity in recent years, polygon meshes or polyhedra remain the most commonly used representation of three-dimensional objects in computer graphics. This is largely because of the existence of fast and efficient polygon renderers. VRML, for example, is largely a polygon-based technology.

As noted earlier, one obvious fault of the polyhedron representation is the inability in representing curved surfaces. Consequently, it requires large amounts of data to approximate objects with curved surfaces. Objects with rounded edges and corners, for example, need large numbers of tiny polygons to approximate. Hence files describing the original "unrounded" volume and those with rounded edges and corners can easily differ in size by an order of magnitude.

The difference is significant especially in the context of VRML, as VRML files are mostly hosted on a remote machine and transferred over the network on request. The difference in



file size may mean a huge cost in scarce network resources. This imposes an extra challenge for three-dimensional volume rounding in VRML.

Compression serves as a solution here. Pre-compressed scene descriptions can be generated, transferred and even manipulated directly. Decompression takes place only before rendering. Data transfer is therefore minimised.

Unarguably one of the most natural and effective ways of compressing rounded three-dimensional object is to actually represent it by its semantic. Instead of describing the rounded volume as a large collection of small polygons, it can be represented as the geometry of the original object and a rounding specification. Rounding algorithm can be applied just before rendering. We will call this approach *Just-In-Time* (JIT) modification.

The benefits of JIT modification can be easily appreciated. A rounded equivalent of a computer graphics scene of less than a hundred kilobytes can be easily larger than the original by several megabytes of data. Assuming an average home-use Internet connection, we are talking about minutes or even hours more on the required download time. In contrast, given the huge and continuously increasing processing power available on desktop computing, a fast volume-rounding algorithm can be executed within seconds.

Enhanced readability and maintainability further justify the JIT approach. With the pre-constructed polygon mesh approach, once the shape is generated it is hard to modify and hence reusability is limited. In comparison, changing the roundness of the volume with the just-in-time approach can be as easy as altering one scalar value in the specification.

## 1.5 Goals and Contributions

This research is initiated from two orthogonal paths: academic and practical. On the academic side, we set out to investigate a new method to perform fast volume rounding. On the practical side, we want to develop a VRML volume-rounding tool, which produces rounded equivalents of the existing input VRML scenes. The focus here is more on the user interface design and the actual implementation.

The major contribution is the development of a new approach to volume rounding - *Polyhedron Beveling*. Polyhedron bevelling methods are based on recursive mesh subdivision methods, which are used for fast construction of a smooth surface from an initial polygon mesh. Recursive mesh subdivision methods will be reviewed in Chapter 4. Polyhedron bevelling methods extend recursive mesh subdivision methods to allow user specification of the degree of roundness. Two approaches have been tried and, as a result, two polyhedron bevelling methods are developed. They are *mesh pre-partitioning* and *target-driven* polyhedron bevelling.

The second contribution is the development of a just-in-time extension in VRML.

## 1.6 Organisation of the Thesis

The remainder of this research is organised as followed.

Chapter two discusses the subjective meaning of volume rounding. It attempts to determine what is meant by “rounding” of a three-dimensional object. The objective is to come up a set of quality criteria, which can be used to evaluate different volume rounding methods.

Chapter three gives an overview of existing volume rounding methods. The emphasis is largely on the relative strengths and weaknesses of each of the methods. We run through each of the methods against the set of quality criteria developed in chapter two. Understanding of the current development on the field helps identifying the potential area of research.

Chapter four reviews various recursive mesh subdivision methods, the bases of our research. The focus for the chapter will be on the mathematical formulation and detail explanation of the methods.

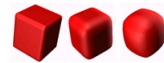
Chapter five presents the first attempt of our volume rounding method: polyhedron bevelling by mesh pre-partitioning. The idea of mesh pre-partitioning polyhedron bevelling is to pre-process the initial coarse polyhedron representation before applying the recursive mesh subdivision method so that the desired shape is obtained after the mesh subdivision. Different degrees of roundness are achieved by varying the pre-processing phase.

Chapter six describes two optimisation methods for the mesh pre-partitioning method. The optimisation methods lead to both a faster algorithm and less fragmentation.

Chapter seven introduces polyhedron bevelling by target-driven subdivision. Instead of adding a pre-processing phase to the recursive mesh subdivision process, the method actually modifies the recursive mesh subdivision scheme itself. The chapter discusses the advantages and disadvantages of both methods.

Chapter eight covers the implementation details of the just-in-time volume rounding extension in VRML using the target-driven subdivision method.

Chapter nine is the conclusion of this research; it also discusses the limitation and the future research directions.



## Chapter 2 Meaning of Volume Rounding

Most real world objects are slightly rounded. Modelling of rounded objects in computer graphics however is tedious and time-consuming. Three-dimensional volume rounding is concerning with automatically rounding sharp edges and corners of an object.

In this chapter, we wish to discuss what we actually mean by “*rounding*” of a 3-dimensional object. This is important both for evaluating the pros and cons of various existing volume-rounding methods and for setting up the goals and directions of our own method.

Since the topic is subjective in nature, we will try to back up our arguments with the result of a survey on how humans perceive the concepts of *roundness* and *rounding*. The details of the survey can be found at the end of this chapter. The result of this discussion will form the basis for evaluating the existing volume rounding methods in the next chapter.

### 2.1 Three-dimensional Volume Rounding: What does it mean?

The meaning of *rounding* of a geometric object is intuitively obvious, but very imprecise. Two questions are essential to the understanding of the truth meaning of a rounding operation.

First, a simple-minded interpretation describes a rounding operation as making the object rounder. The focus here is on the word “*rounder*”. The first question of concern is therefore: “*How to compare or quantify roundness?*” To answer the question, we need to study what characteristics of an object contribute to the human perception of its roundness. Section 2.2 is devoted to approach this question.

Assuming the roundness of objects is measurable, a rounding operation cannot be interpreted as simply making an object rounder. Some additional constraints, which govern the relationship between the original volume to the resultant shape, must be satisfied. Moreover, one will, for example, not expect rounding a rectangular box will yield a cucumber. The essential nature of the original shape must be unaltered during the rounding process. We will concern ourselves with the second question: “*What are the constraints that govern the resultant shape of a rounding operation?*” In section 2.3, we will address this question and attempt to develop a set of constraints that limit a rounding operation to produce only “*reasonable*” outputs.

### 2.2 Quantifying Roundness

To ensure a uniformity of terminology, we will first distinguish between two related but distinct concepts: smoothness and roundness.

### 2.2.1 Roundness vs. Smoothness

Quantifying *smoothness* is a well-researched area. Smoothness is defined as the continuity of points over a surface. Parametric and geometric continuity are two generally accepted concepts.

For a parameterised surface, the parametric continuity  $C^k$  is used to quantify smoothness, denoting a surface that is continuous up to the  $k$ -derivative. Parametric continuity is not invariant with respect to re-parameterisation. When a parametric representation is not present or re-parameterisation is necessary, geometric continuity can instead be used to quantify smoothness. Geometric continuity,  $GC^k$  or  $G^k$ , measures smoothness in terms of order of contact. For example, two surfaces have zero order of contact means the two are touching each other. Higher order of contact means continuous in gradient, curvature, torsion and so forth. Geometric continuity is usually less restrictive than parametric continuity. For example, two curves with  $GC^1$  continuity requires only that the tangent vectors have the same direction at the joint point, while  $C^1$  continuity requires the tangent vectors match in both direction and magnitude. However, geometric continuity does offer comparative advantages like geometric interpretation and parameterisation independence.

Unfortunately, on the other hand, *roundness* is not mathematically well defined. People do not give quantitative measurement for roundness in practice. Roundness describes how closely a shape resembles a circle or a sphere in two and three-space respectively. A sphere is completely rounded while a cube is not. Linguistic quantifiers such as *very*, *moderately* are used to describe something in between.

The qualitative nature of roundness is obviously undesirable for our application. What is even worse is that the roundness of a three-dimensional object can be a very subjective interpretation. How a person perceives roundness of an object can be different from one to another.

### 2.2.2 Interpretation of Curvature Plot

Despite the qualitative and subjective nature of roundness, curvature at the rounded edges and corners is unarguably the single most influential factor that affects how people perceive roundness. Although curvature has an intuitive meaning to people, we will give the definition here to avoid unnecessary confusion.

In two dimensions, the radius of curvature can be interpreted geometrically as the radius of the circle whose first and second derivatives agree with the curve at the given point and the curvature is given as the inverse of the radius of curvature.

Curvature in three-dimension is more complicated. A *normal section* plane which is tangential to the surface at point  $p$  must be chosen. See figure 2.1.

The normal curvature  $\kappa_n$  at point  $p$  can then be measured like the two-dimensional case along the intersection curve of the normal section plane and the surface. Clearly, the curvature  $\kappa_n$  varies as this normal section rotates along the normal vector  $N$ . The most important, however, are the two principal curvatures,  $\kappa_1$  and  $\kappa_2$ , which are the two extreme values among all  $\kappa_n$ , since any in-between normal curvature  $\kappa_n$  at  $p$  can be obtained by:

$$\kappa_n = \kappa_1 \cos^2 \phi + \kappa_2 \sin^2 \phi,$$

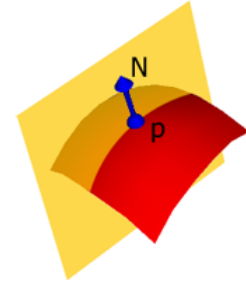


Figure 2.1. Normal section.

where  $\phi$  is the angle between the normal section and the direction of the first principal curvature [HL97 pp.45].

Clearly the larger the radius of curvature on the edges and corners the rounder the object. Since the radius of curvature of edges and corners will change with scaling, that implies scaling of an object will alter its roundness. This, however, contradicts the result of the survey, covered later in section 2.6.3, in which most people think that the roundness of an object is invariant to scaling.

Investigating the curvature plots will give a deeper insight into how the perception of roundness relates to curvature. Curvature plots are obtained from plotting the curvature along a particular path on the object surface. Figure 2.2 depicts curvature plots across a sharp edge from a cube and a rounded edge from a rounded cube.

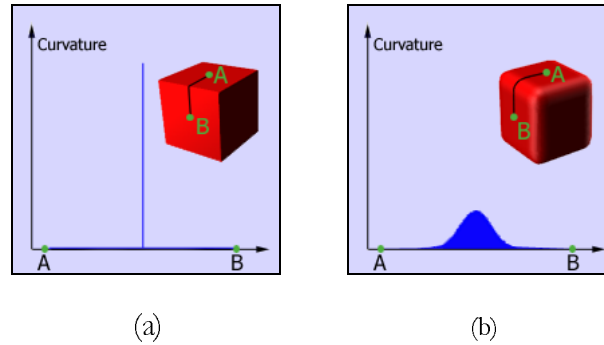


Figure 2.2. Curvature plots of a sharp edge and a rounded edge

Notice that in the curvature plot of the sharp edge illustrated on figure 2.2(a), the curvature remains at zero except for a very sharp discontinuity at the turning point. In comparison, figure 2.2(b) shows that on a rounded edge the sharp discontinuity is spread out over a wider range with a lower peak curvature.

We can see that curvature relates to the degree of roundness in multiple aspects namely: the *peak curvature*, the *shape* of the curvature plot and the *spread* of the curvature.

- The *peak curvature*, given that the size of the object is fixed, controls how sharp the turning point is.



- The *shape* of the curvature plot governs the smoothness of the surface and the shape of the rounded region. While it is possible to construct a rounded object with a rough surface, rounded objects are generally smoother. There are many methods to control the general shape of the rounded region and we will cover some of them later.
- The *spread* of the curvature on the edges and corners is sometimes hard to conceptualise. It can be better understood if viewed as the ratio between the surface area of the planar regions and that of the rounded regions. Planar regions are identified by their zero curvature. The spread of curvature turns out to be the major factor affecting human recognition of degree of roundness. This factor matches several phenomena observed from the survey. These phenomena include 1) roundness is invariant under scaling and 2) the roundness of objects varies with their sizes even if the curvature of all their edges and corners are identical.

The discussion so far has focused on the degree of roundness of an individual edge or corner. One can expect the overall roundness to be some combination of the different rounded parts of an object. Some interesting questions can be raised. For example, what will people think of the roundness of a teardrop shape (a sharp and pointy end but round and smooth everywhere else)? This research is, however, primarily concerned with a global rounding operation: a rounding operation that can be applied everywhere equally to a target object. Hence the degree of roundness of an individual edge or corner will serve as a fair estimation of the overall degree of roundness.

## 2.3 Constraints on the Rounding Operation

The previous section was concerned with comparing and quantifying roundness. In the current section, we will have a look at the other question raised in section 2.1: “*what are the constraints that govern the resultant shape of a rounding operation?*”

We begin by claiming that it is necessary to maintain the geometric similarity between the original and the rounded shape. We will successively refine the requirements or constraints in this section.

### 2.3.1 Preservation of Topology

The similarity in shapes is studied in the theory of homotopy [Sier92]. This is usually related to the sharing of the same topology. By topology, we are referring to the connectedness of shape, including the number of disjoint components and the number of holes in each of them.

Our survey indicated that most people believe comparing roundness of two objects with different topology is not sensible. Although we do not claim the result to be conclusive, this observation provides evidence that the *preservation of topology* is a necessary constraint on a rounding operation.

However, preservation of topology by itself cannot be sufficient. Topology contributes little to the general appearance. For example, a cucumber and a rectangular box will have the same topology and therefore some additional constraints are clearly required.

### 2.3.2 Preservation of Planar Surface

We mentioned in section 2.2 that the degree of roundness of an object is highly related to the ratio of the area of its rounded regions and planar regions. Hence rounding can be described as a process of reducing the area of the planar regions. This implies the rounded object will preserve, to some extent, the planar regions of the original object.

We will first define some notation.

Planar regions are all point  $p$  on the surface of an object such that the two principal curvatures of  $p$  are both zero. Otherwise the point is a member of the rounded regions.

Consider a rounding operation  $R$  with a single scalar control of roundness  $\rho \in [0, \infty)$  and let  $\tau$  be the topological space of the three-dimensional Euclidean space. In other words,  $\tau$  is a set of all possible subsets of the three-space.

The signature of the rounding operation can hence be written as  $R: \tau \times [0, \infty) \rightarrow \tau$ . This simply means the rounding operation takes a shape and a degree of roundness and maps it to another (or possibly the same) shape.

Also let  $S: \tau \rightarrow \tau$  be a function that maps any given volume to its planar surfaces. That is  $S(t)$  is all points  $p$ , such that  $p$  is on the boundary of  $t$  and the two principal curvatures at  $p$  are both zero.

Now we can describe the constraint that governs the preservation of planar surface, for any given initial volume  $t$ , as:

$$|S(R(t, \rho_2))| \geq |S(R(t, \rho_1))| \Leftrightarrow \rho_1 \geq \rho_2$$

where  $|x|$  denotes the cardinality or size of the set  $x$ .

In words, this means given any two rounded versions of the same initial volume, the total area of the planar surfaces of the more rounded one will be less than that of the less rounded one.

### 2.3.3 Monotonic Reduction of Planar Regions

Not only do we want the area of the planar surface to reduce as the required degree of roundness increases but also we require a constraint to guarantee that no extra planar regions will be generated by the process.

In order to accomplish this, we will strengthen the constraint from section 2.3.2, while keeping the same notation, as:

$$S(R(t, \rho_1)) \subseteq S(R(t, \rho_2)) \Leftrightarrow \rho_1 \geq \rho_2$$

This means that given any two rounded versions of the same initial volume, the set of planar surfaces of the more rounded one is required to be a subset of that of the less rounded one.

### 2.3.4 Shrinking from Boundary

Even with the above constraints, we still cannot avoid undesired behaviour such as introducing curvature in the middle of the planar regions. If we regard rounding as a process of shrinking planar regions of an object, we need to restrict that the shrinking processing takes place only on the boundary of the planar regions.

Again, we will start with defining some more notations.

The boundary constraint means that planar region can be reduced only if it is a *boundary planar region*, which is defined as all point  $p$ , such that  $p$  is planar while one or more immediate neighbour of  $p$  is non-planar.

The  $\epsilon$ -boundary planar region includes all point  $p$ , such that  $p$  is a member of the planar regions and there exists a point  $q$  on the boundary planar region and  $q$  is a member of the  $\epsilon$ -neighbourhood of  $p$ .

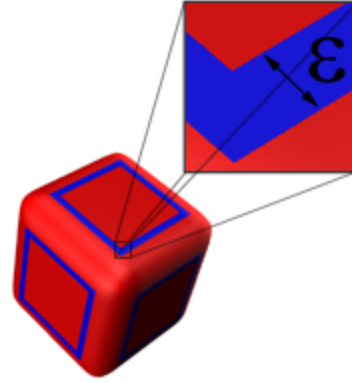


Figure 2.2. The  $\epsilon$ -boundary planar regions of a rounded cube

The  $\epsilon$ -neighbourhood of a point  $p$  in a metric space, which is a three-dimensional Euclidean space in this case, is a set of all points within a distance of  $\epsilon$  from  $p$ .

Figure 2.2 shows an  $\epsilon$ -boundary planar regions of a rounded cube.

Let  $B: \tau \times [0, \infty) \rightarrow \tau$  be the function that map a shape  $t \in \tau$  and a distance measure  $\epsilon \in [0, \infty)$  into the  $\epsilon$ -boundary planar region of  $t$ .

Now, the boundary constraint can be written as:

for any arbitrary  $t \in \tau, \rho \in [0, \infty)$

$\exists f: [0, \infty) \rightarrow [0, \infty)$  and  $f$  is continuous, such that

$$\lim_{x \rightarrow 0^+} f(x) = 0 \text{ and}$$

$$S(R(t, \rho)) \setminus S(R(t, \rho + \Delta\rho)) \subseteq B(R(t, \rho), f(\Delta\rho))$$

A function,  $f$ , is continuous if  $x$  is a neighbour of  $y$  then  $f(x)$  is also a neighbour of  $f(y)$ .

With this extension, for every single connected planar surface  $S$  on the original volume, the corresponding planar surface(s)  $S'$  on the rounded version will have an additional property: the number of holes in  $S'$  must be less than or equals to that of  $S$ . This guarantees that no additional holes will be generated on the planar surfaces by the process.

We believe that by satisfying all the above constraints, most of the unreasonable outputs are eliminated. Further constraints are possible but greater effort is required to avoid the limitation on the design freedom of the rounding operation since the topic is, by and large, subjective.

## 2.4 Degree of Roundness and Limit Behaviour

An essential aspect of a volume rounding operation is not stated explicitly in the discussion so far: the degree of roundness of the resultant shape must be user adjustable, which leads to the requirement of a rounding parameter. This, in turn, gives rise to two additional desired behaviours of any rounding method, namely: *a well-defined limit behaviour* and *continuity in controllable roundness*.

### 2.4.1 Well-defined Limit

It is our belief that the *well-defined limit behaviour* suggests two criteria.

First, there must be a *zero rounding* primitive. That is, there is a certain parameterisation such that the rounding operation is identical to the identity mapping.

$$\exists \rho (\forall t (R(t, \rho) \equiv t))$$

Second, there must be at least one parameterisation that always maps a given input to a *limit of rounding*. By *limit of rounding*, we mean an object that is already maximally rounded; attempting to apply further rounding will result in the same object. Let's define  $L \subset \tau$  as the collection of all limits of rounding, then by definition:

$$t \in L \Leftrightarrow \forall \rho (R(t, \rho) \equiv t)$$

Thus we can write the second criteria as:

$$\exists \rho (\forall t (R(t, \rho) \in L))$$

### 2.4.2 Continuity in Controllable Roundness

The *continuity in controllable roundness* requires the rounding operation itself to be continuous. If the rounding parameter  $\rho_1$  is a neighbour of  $\rho_2$ , then, given any arbitrary input shape  $t$ , the resultant shape of the rounding operation  $R(t, \rho_1)$  must be a neighbour of  $R(t, \rho_2)$ . We regard two shapes  $s$  and  $t$  as neighbours if and only if every points in  $s$  has a neighbour in  $t$  and vice versa.

## 2.5 Different ways of Controlling the Rounded Regions

Our discussion, so far has primarily focused on the preserved planar surfaces of a rounded object. One cannot really finish a discussion on rounding without mentioning the other half of a rounded object: the rounded regions. They are blends that smoothly join up the preserved planar regions. There are many acceptable ways of controlling the shape of the rounded regions depending on the blending method employed. Four common ways of controlling the rounded regions are depicted in figure 2.3 [HL97]. They are:

- (a) *Unbounded blending*, where the blends globally influence the shape of the objects. The resultant surface usually lies slightly inside or outside of the control volume and hence no planar region of the control volume will be preserved. No obvious measurement of roundness is associated with this type of blending.
- (b) *Volume-bounded blending*, where the spread of the blend is controlled by a prescribed volume and is usually subject to some sort of smoothness conditions. Therefore the degree of roundness of this blend is obviously controlled by means of the prescribed volume. The larger the prescribed volume the rounder the resultant shape.
- (c) *Range-bounded blending*, where the spread of blend is controlled by a given distance measure from the edge (or vertex). Note that a single value scalar measurement is not sufficient to describe the degree of roundness of this sort of blending. In a two-dimensional case, one can adjust the distance measures of each of the straight-line segments joining the corner. In a three-dimensional case, there is one control from each edge on each face so that the degrees of freedom rise to  $2 \times n$ , where  $n$  is the valence of the corner of concern.
- (d) *Curvature-bounded blending*, where the radius of curvature of the blend is fixed. The degree of roundness of this sort of blending is controlled by the given curvature. The roundness of a corner increases as the curvature decreases.

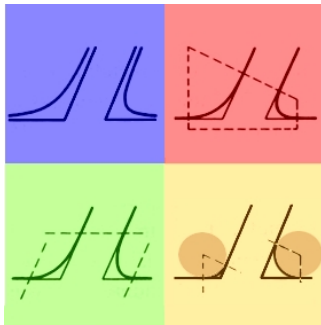


Figure 2.3. Four possible ways of controlling the region of roundness.

- (a) Unbounded
- (b) Volume-bounded
- (c) Range-bounded
- (d) Curvature-bounded

(from: [HL97] pp374 fig 14.1)

## 2.6 Survey on Human Perception of Roundness

We have already mentioned several times the survey we performed on how humans perceive the concepts of *roundness* and *rounding*. The details are covered in this section. Firstly the aim and the design of the survey will be reviewed. Next, the results will be presented, along with the interesting observations.

### 2.6.1 Aims and Design

The aim of the survey is to investigate how humans perceive roundness. In particular, we are interested in examining the relevance of the factors that we thought would alter the perceived roundness of an object. The questions we are going to explore are:

- How the scaling of an object (thereby altering the curvature of the edge) affects its perceived roundness.
- The effect of the size of an object on perceived roundness given that the curvature of the rounded regions of the object remains unchanged.
- How people think about comparing the roundness of objects with different topology.
- How people think about comparing the roundness of objects with very different shape.

The survey is made available on the World Wide Web using the online testing module of the *CECIL* project [CECIL] – a custom-built computer-supported learning facility within the University of Auckland. Participants can perform the survey anywhere in the world via the Internet. The responses from each participant are recorded in a database for easy manipulation and analysis.

Random sampling is very hard to achieve in this situation. Internet-based surveys usually suffer from both *selection* bias and *self-selection* bias. There can be selection bias because only the people with Internet access are “selected”, which may result in an unrepresentative subgroup of the population. Also there can be self-selection bias as surveys are made available to large numbers of people (all Internet users) while people themselves decide whether or not to be surveyed. However, given the resources available, it is infeasible for us to perform a proper random sampled survey and, at the same time, obtain a convincing survey size and response rate. Hence, the results are helpful to provide us a good insight but no conclusive information should be drawn merely from the findings.

There are two versions of the survey and each contains six questions. The two versions contain similar questions except that the wording is slightly altered so that one of them emphasises the *operation* of rounding and the other is concerned with the pure comparison of roundness.

The result of each version is obtained from an independent group of participants. We enforced that by discarding any results if the participant had attempted either of the two versions before. Duplicated attempts are identified by the participant’s IP address and the persistent client status called “cookies”. Participants are also asked if they have attempted the other version before. Even with all the precautions, we cannot guarantee the complete non-existence of duplicated attempts in an online environment.

Each question contains an image of two objects as shown in figure 2.4. Every question from the same version has the same wording and options. In the first version of the survey, the question is “*Which of the objects shown in the image is rounder?*” and, in the second version, the

question is “*The two objects shown are cubes that have had a rounding operation applied to them. Which do you think is more rounded?*”

In both versions, the participant is asked to choose from the four given options below:

1. The one on the left
2. The one on the right
3. They are the same
4. It is not sensible to compare

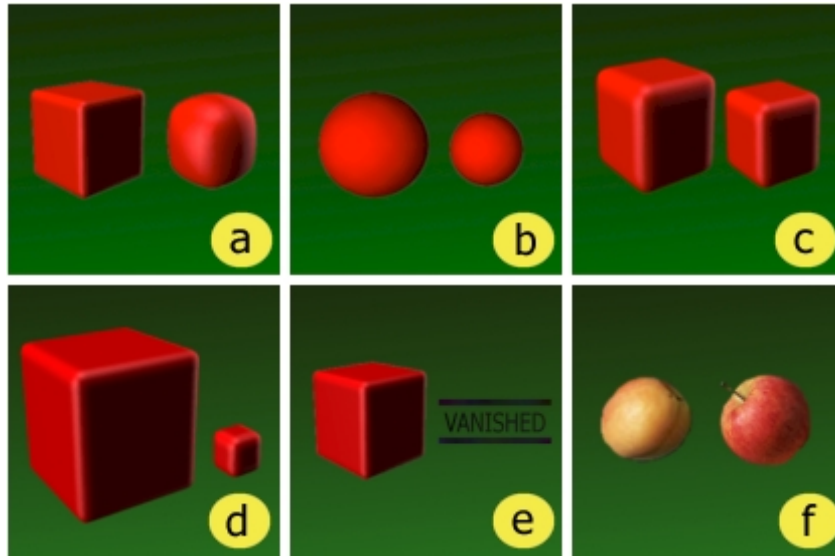


Figure 2.4. Images used in the survey on human perception of roundness

Question 1 shows two rounded cubes with very different degrees of roundness. See figure 2.4(a). This question is intentionally made obvious and serves as a control question. Results from participants whose fail in answering this question “correctly” are regarded as outliers and discarded.

The presentation order of questions 2 to 6 is randomised by the system at runtime to minimise the potential effect on the result from a particular ordering. Participants are, however, given the ability to go back to a particular question during the survey and make changes before they finally submit their survey result. The system records not only the result of the survey but also every aspect on the survey. Hence, we can ask interesting questions such as what percentage of people answer a particular question instantly, say, within two seconds, or what percentage of people change their mind on a particular question after seeing another question. However, we did not notice any interesting patterns from that.

Question 2 shows two spheres of different size. See figure 2.4(b). We believe that spheres are *the most rounded* objects possible, hence their degrees of roundness are equivalent. This question explores whether people will consider either one is actually rounder.

Question 3 again explores the effect of scaling on the human perception of roundness. However the spheres are replaced with two rounded cubes of different sizes. See figure 2.4(c).

Question 4 explores the effect of curvature on edges and corners. The larger the radius of curvature on the edges and the corners, the rounder the object. Participants are asked to compare the roundness of two rounded cubes that are very different in size but with the same curvature of the edges and corners. See figure 2.4(d).

In question 5, participants are asked to compare the roundness of a rounded cube and an empty space. See figure 2.4(e). An empty space is chosen, as we are particularly interested in whether people can accept the idea that applying a very heavy rounding leads to a completely vanishing of the object. The concept of an empty space is very hard to describe in the context of an image. An extra written explanation is included. However, feedback from various participants indicated that this question is still confusing.

Question 6 explores how people compare the roundness of objects with very different shapes but with the same topology. Participants are asked to compare the roundness of a peach and an apple. See figure 2.4(f).

### 2.6.2 Survey Result

The results of the survey were collected during the period from 4<sup>th</sup> May 1998 to 15<sup>th</sup> May 1998. Totals of 44 and 24 participants were surveyed for the first and the second versions respectively. Two out of the 44 and four out of the 24 participants were identified as outliers by failing to answer the first question “correctly”. That leaves 42 and 20 valid responses. The fact that a rather large proportion (16.7%) of the participant failed to correctly answer the obvious question for the second version may be due to the small sample size or confusing wording.

Not only are we interested in the results of the survey but also the potential errors. In order to obtain an estimation of the errors, we need to know the distribution of the population we sample from. Obviously we cannot have any knowledge of that, but by the *Central Limit Theorem*, we will assume the sample is normally distributed. The validity of this assumption is, however, arguable with a small sample size of 20. Most elementary statistics books say the assumption is good if the sample size is around 30 or more. In practice, the central limit effect works so well that even if the initial distribution is very skewed, it will converge towards a reasonable normal distribution with a sample size as small as 10. So we assume a normal distribution, but the reader should notice the possibility of an inadequate approximation here.

Since we also estimate the standard deviation using the standard error, the student-t distribution will be used with the degree of freedom of  $n - 1$  for a better estimation than the normal distribution, where  $n$  is the size of the sample.

The results are plotted in figure 2.5 with the 95% confidence intervals.



### 2.6.3 Observations

The second version of the survey has relatively wider confidence intervals because of the smaller sample size. However, the general distributions of both versions of the survey are very similar indicating that the effect of the different wordings is not significant. At least, we cannot reject the hypothesis that the two are statistically identical. The dominant options of each question in both versions are the same except for question 6. We focus our discussion on the first version of the survey because of their similarity of both surveys and the larger sample size of version one.

All questions except question 6 have the dominant option chosen by the majority of the population.

The result of question 2 shows that most people think that two spheres of different sizes have the same degree of roundness.

The result of question 3 reveals that most people think a rounded cube is of the same degree of roundness as a shrunk version of the same rounded cube. This result is consistent with the last question. Hence, this gives further evidence that the perceived roundness is invariant to scaling.

The result of question 4, however, shows that given two rounded cubes that are very different in size but with the same curvature on the edges and corners, most people think that the smaller one is actually rounder (more rounded). This indicates that the curvature on the edges and corners is not the only effect that governs the perceived roundness, but the ratio of area of the planar regions to area of the rounded regions is in fact more important.

The result of question 5 reveals that most people think that it is not sensible to compare the roundness of a rounded cube with an empty space. The feedback from our participants suggested that this question is relatively confusing. That may contribute to the large proportion of participants selecting the option “it is not sensible to compare”. While we cannot draw firm conclusions about preservation of topology and comparison of roundness, the result is still indicative that rapid change in the topology is not intuitive in the rounding process.

There is no majority option in question 6. This indicates that while most people think that it is sensible to compare the roundness of two objects with very different shape, the judgement of roundness is, however, very subjective in this complex situation. It is also interesting to note that while only 10% of the participants (version one) think that it is not sensible to compare roundness of two objects with very different shape, the percentage increases to 40% when talking about a rounding operation (version two).

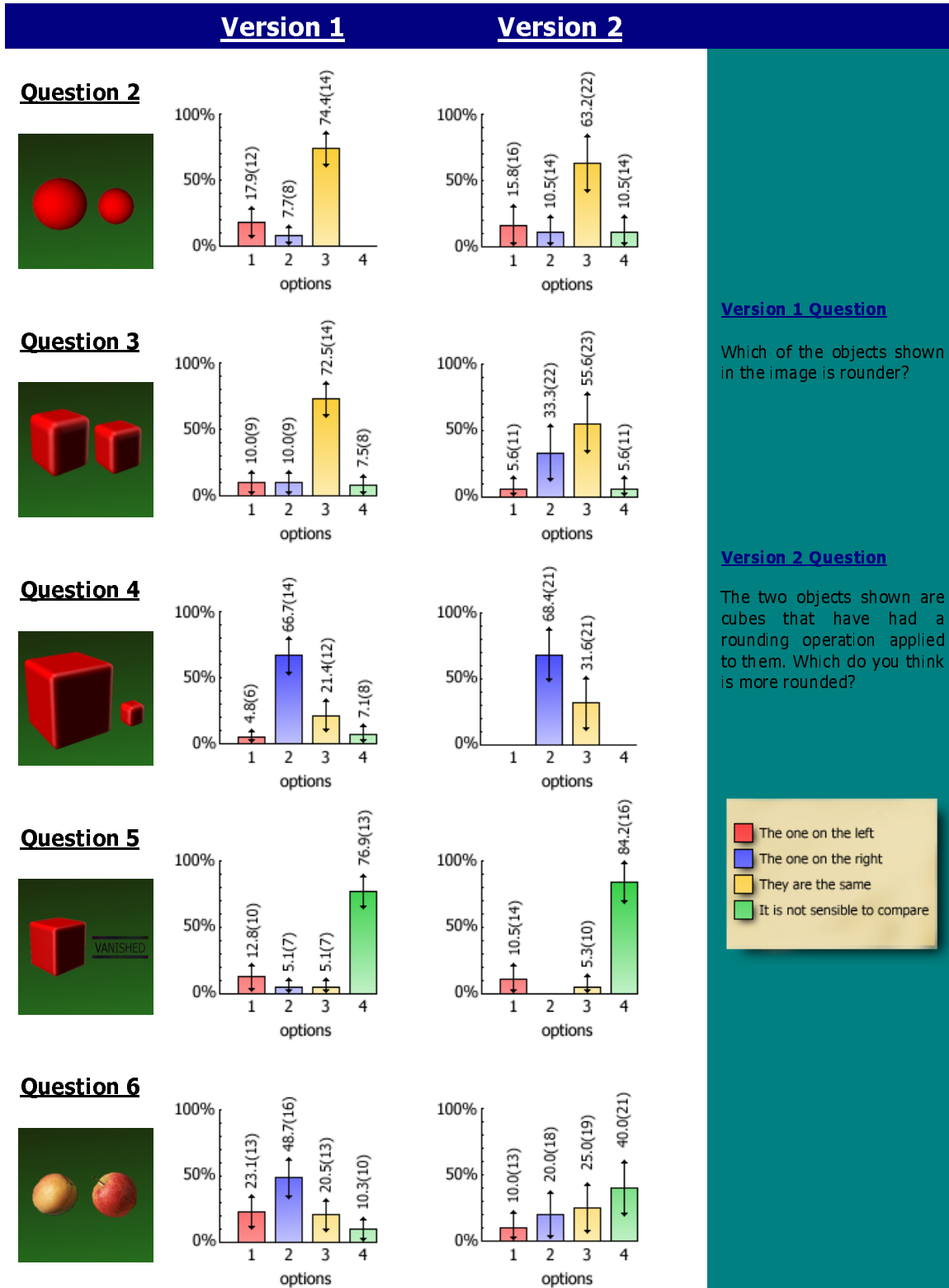


Figure 2.5. Survey results

## 2.7 Quality Criteria of Volume Rounding

At the beginning of the chapter, we mentioned that the discussion in this chapter is going to be used as a basis for evaluating various existing volume rounding methods. To fulfil this objective, we will define in this section, a set of quality criteria of general-purpose volume rounding methods. To a large extent, the quality criteria are based upon those desired behaviours we covered before. In addition, criteria evaluating the algorithm itself such as the execution speed and the stability are included. The quality criteria are summarised below:

- ⇒ Algorithm Execution Speed
- ⇒ Algorithm Stability for Arbitrary Input Configuration
- ⇒ Preservation of Topology
- ⇒ Preservation of Planar Region
  - ◆ Reduction of Planar Area
  - ◆ Monotonic Reduction
  - ◆ Shrinking from Boundary
- ⇒ Smoothness or Continuity
- ⇒ User Controllable Degree of Roundness
  - ◆ Continuity in Roundness Control
  - ◆ Flexibility in Roundness Control
  - ◆ Well-defined Limit Behaviour
    - Existence of Limit Rounding
    - Existence of Zero Rounding

## 2.8 Chapter Conclusion and Summary

The chapter has explored the conceptual view of the problem – what exactly is the meaning of volume rounding to humans. Although some may challenge whether it is worth putting effort into something that contributes so little to the final algorithm, it is our belief that it is always a good practice to fully understand the problem before rushing into the solution.

Our focus in this chapter has been mainly on “*How to compare or quantify roundness?*” and “*What are the constraints that govern the resultant shape of a rounding operation?*” Since the topic is subjective, our discussions were largely based on the result of a survey.

The most important finding is that instead of the curvature of the rounded edges and corners, the ratio of the preserved planar area to the total surface area of the unrounded object is a more intuitive and significant measure of the “roundedness”. We therefore recognise the importance of the preservation of planar regions and hence the “essential polyhedral nature” as the major goal of our volume-rounding method.

Finally we concluded the chapter with a list of quality criteria. These criteria will then become our evaluation tool for the existing volume rounding methods.



## Chapter 3 Existing Volume Rounding Methods

This chapter is an overview of volume-rounding methods. There are three main classes of method namely *implicit blending*, *parametric blending* and *recursive mesh subdivision*. We will first briefly discuss each of them. At the end of this chapter, we will have a look at how the various methods fulfil the quality criteria we presented in the previous chapter.

### 3.1 Implicit Blending Methods

Implicit blending methods involve describing the rounded object in the form of an implicit surface. There are many variants of implicit blending, which differentiate themselves by how the rounded surface is defined. Convolutional smoothing and rolling-ball blends are the two most common examples of implicit blending.

#### 3.1.1 Convolutional smoothing

Convolution is often used for smoothing images in the field of image processing. The idea is to blur out high frequency sharp edges of an image by averaging. Colburn [Colb90] applied a three-dimensional Gaussian filter to an octree representation of a solid volume. The surface is defined as the locus of the centre of a given gaussian (or solid spherical) filter where one half of the weighted volume is inside and the other half is outside the original solid volume. See figure 3.1. The result is a nicely smoothed surface with high frequency corners and edges being rounded out. One great advantage of this approach is that the degree of roundness of the resultant volume is controllable by simply adjusting the size of the gaussian filter.

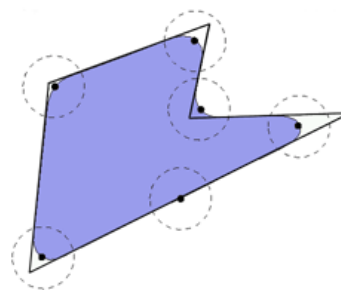


Figure 3.1. Convolutional smoothing of a polygon by a circular filter.

from: [Lobb96]

Convolutional smoothing, although mathematically simple, is computationally expensive. It involves the computation of the exact weighted volume of intersection of the scene and the filtering sphere. Lobb [Lobb96] presented quasi-convolutional smoothing (QCS) as a fast approximation of convolutional smoothing. Quasi-convolutional smoothing requires the input polyhedra in a format of Constructive Solid Geometry with the limitation that only the half-space primitive is allowed. Arithmetic addition, subtraction and multiplication are then used to approximate set union, difference and intersection of half-spaces respectively.

Dansted [Dans96] presented the algorithm for calculating exactly the volume of intersection of a polyhedron and a sphere. However he also concluded in his paper: that “the intersection

function is at least one, heading towards two orders of magnitude slower than the corresponding QCS operation.”

Rendering of the resultant volume of implicit blending methods is usually by means of ray tracing. If fast rendering is required, polygonisation is usually necessary. Wünsche [Wuns96] presented Triage polygonisation that is highly optimised for CS/QCS smoothed objects. Convolutionally or quasi-convolutionally smoothed objects usually contain large planar regions. Triage polygonisation takes advantage of this producing fast polygonisation with low fragmentation by extracting those regions right at the beginning. In comparison to general-purpose polygonisation schema like marching cubes [LC87], “*Triage polygonisation is 20-30 times faster and outputs only 1-2% of the polygons* (when applied to CS/QCS smoothed objects)” [Wüns96].

### 3.1.2 Rolling-ball Blends

Rolling-ball blends are also known as cylindrical fillets. Rolling-ball blends can be easily realised by rolling a sphere with a given radius along the inner surface of the control volume [HL97]. See figure 3.2.

To the best of our knowledge, Rossignac was the first to suggest the blending of this type [HL97]. The surface of a rolling-ball blend achieves  $GC^1$  continuity.

Simple rolling-ball blends are also called constant radius blends (CR-blends), since the radius of the rolling-ball is fixed. A variation of rolling-ball blends allow this radius to vary; these are called variable radius blends (VR-blends).

The blends are, therefore, defined as the *envelope* of a sphere with radius  $r(p)$ , whose centre  $p$  moves along a space called the *directrix*. The directrix contains all points  $p$  such that a sphere centred at  $p$  with radius  $r(p)$  fits completely inside the original solid volume [HL97]. For a rolling-ball with a fixed radius  $r$ , the directrix is the space bounded by the inner offset surface with a distance of  $r$  from the surface of the original volume.

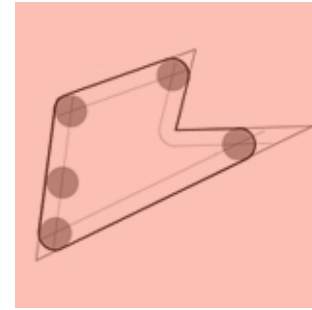


Figure 3.2. Rolling-ball blend defined on the same polygon as in figure 3.1.

Even with simple rolling-ball blends, the determination of the directrix and the blend can be very expensive in practice. For example, in [HL97] the CR-blend of two cylinders is described by an equation with 969 terms. The mathematical description of VR-blends is even more complicated.

## 3.2 Parametric Blending Methods

As the name implies, parametric blending describes in parametric form a surface that blends planar regions of a rounded object. Parametric blending can usually be separated into two distinct stages. They are the determination of trim curves and the fitting of blending

surfaces. Trim lines are the boundary on the preserved planar surfaces. Smooth, i.e., with  $C^1$  or  $GC^1$  continuity, parametric patches are then fitted onto those trim curves.

Trim curves can be defined both interactively and automatically. There are many automatic approaches. For example, one common approach as described in [HL97] is based first on the calculation of the intersections of the offset surfaces with the two given surfaces. Trim lines are then taken as the orthogonal projections of the intersection curve back onto the original surfaces.

Fitting a blending surface requires finding an explicit formula for a patch that satisfies some boundary and gradient conditions. The blending method depends on the form in which the trim lines are given. If the trim lines are defined in closed form, for example B-spline or Bézier curves, then the blend can be constructed as a tensor product surface. However, usually trim curves need to be defined as a set of points only. In that case, the blend is described as a set of curves. Notice that the correspondence of points in each trim curve needs to be specified.

While it is easy to fit a blending surface between two planes given boundary gradient and trim lines, the problem becomes exponentially harder to solve when considering blending surface that joins  $N$  faces. Common examples are the suitcase corner ( $N=3$ ) and the house corner ( $N=5$ ). Figure 3.3 shows some solutions to these  $N$ -sided vertex problems.

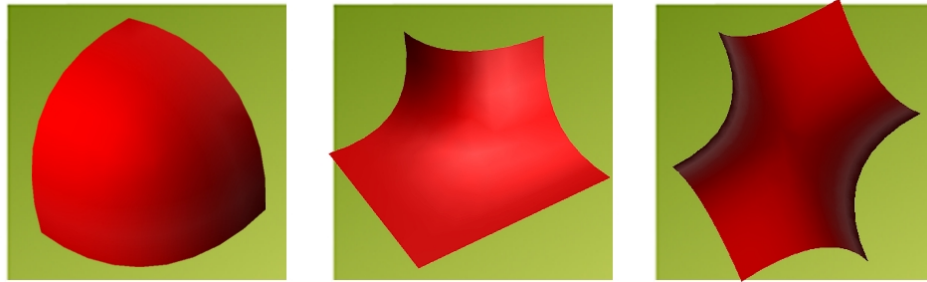


Figure 3.3. Examples of blending surface of  $N$ -sided vertex where (a)  $N=3$  (b)  $N=5$  (c)  $N=6$

### 3.3 Recursive Mesh Subdivision Methods

Recursive mesh subdivision methods are not volume rounding methods as such, in the sense that they do not produce rounded volumes with user specified roundness. Instead they generate smooth surfaces from a control mesh. A subdivision surface is defined by recursively applying a refinement procedure to a polyhedron-like configuration. The only difference of these configurations from normal polyhedra is that their polygonal faces do not need to be planar.

Chaikin [Chai74] introduced a recursive corner-cutting refinement to a list of control points to generate a curve that was shown equivalent to a uniform quadratic B-spline by Riesenfeld [Ries75]. Catmull and Clark [CC78] and Doo and Sabin [DS78] introduced the first popular subdivision refinement schemes for polygon meshes. They both produce biquadratic and bicubic tensor product B-spline surfaces from quadrilateral meshes. Loop [Loop87]

introduced a subdivision scheme based on quartic triangular B-splines. Joy and MacCracken [JM96] extended the work of Catmull-Clark to Catmull-Clark volumes.

Catmull-Clark's quadratic and cubic surface respectively achieve  $C^1$  and  $C^2$  continuity except on some extraordinary points. Extraordinary points are those vertices at which  $n$  edges join or the centroids of  $n$ -sided faces where  $n$  is not 4. Doo and Sabin [DS78] analysed the behaviour around extraordinary points using Fourier transforms and an eigen analysis. Fine-tuning of the quadratic refinement rules is presented in which the resultant surface exhibits the so-called ideal behaviour, namely  $GC^1$  continuity at extraordinary points.

Subdivision surface methods can be generally categorised into approximating and interpolating. The schemes mentioned so far are all approximating. Approximating schemes can be thought of as a recursive process of knocking off corners. The resultant limit surface shrinks inside the control mesh. Interpolating schemes, as implied by the name, generate surfaces that interpolate control points and normals. Some interpolating schemes interpolate only boundary control points while some interpolate all control points and normals.

Nasri [Nasr87] discussed boundary control with Doo-Sabin quadratic surfaces. He used the fact that each face of the control mesh converges towards its centroid during the refinement, and modified the topology of boundary faces to generate a subdivision surface that interpolates the boundary of its control mesh. Later [Nasr91], he presented a method to force quadratic Doo-Sabin surfaces to interpolate vertices and normals. Halstead, Kass and DeRose [HKD93] did the same for cubic Catmull-Clark surfaces.

Loop [Loop87], based on a generalisation of quartic triangular B-spline, suggested a subdivision scheme that works with arbitrary triangular meshes.

Dyn, Levin and Gregory [DLG90] presented the Butterfly scheme that produces a smooth surface that interpolates all vertices. The resultant surface exhibits  $C^1$  continuity over a topologically regular control mesh, where regular means that all vertices have valence of 6 for triangular patch. Zorin, Schröder and Sweldens [ZSS96] modified the Butterfly scheme to produce smooth interpolating surface even on an irregular control mesh. Like Loop's method, both the Butterfly scheme and the extended Butterfly scheme require inputs of triangular meshes.

Loop [Loop94] later presented another variant of surface subdivision. He applied one step of Doo-Sabin refinement to ensure all extraordinary points are isolated. He then derived directly a set of Bézier patches that achieve a global  $C^1$  continuity over the resultant volume.

Peters and Reif [PR97] presented the mid-edge subdivision method, a much simpler set of refinement rules, which after every second subdivision step, produces similar configurations to the Doo-Sabin method.

### 3.4 The Fulfilment of Quality Criteria

Implicit blending methods focus on the description of a rounded volume from an original solid volume. Parametric blending methods are mostly concerned with fitting blends

between different surfaces. The focus there is more on the continuity of the resultant surface. Recursive mesh subdivision methods, strictly speaking, cannot be regarded as volume rounding methods, as it is more correct to say that they produce smooth surfaces rather than saying that they produce rounded volumes. Despite the diversity of the foci, we will have a look at how the three types of existing volume rounding methods fulfil our quality criteria developed in section 2.7.

The candidates here are convolutional smoothing, quasi-convolutional smoothing, rolling-ball blends, parametric blending methods and recursive mesh subdivision methods. They are tabulated against the quality criteria in figure 3.4.

	Convolutional Smoothing	Quasi-Convolutional Smoothing	Rolling-ball Blends	Parametric Blending Methods	Recursive Mesh Subdivision Methods
<i>Algorithm Execution Speed</i>	Extremely slow	Varies <sup>α</sup>	Extremely slow	Varies <sup>β</sup>	Fast
<i>Algorithm Stability for Arbitrary Input</i>	Good	Good	Good	Fair <sup>δ</sup>	Good
<i>Preservation of Topology</i>	No <sup>ε</sup>	No <sup>φ</sup>	No <sup>γ</sup>	Yes	Yes
<i>Preservation of Planar Regions</i>	Yes	Yes	Yes	Yes	No
<i>Reduction of Planar Area</i>	Yes	Yes	Yes	Yes	complete vanish
<i>Monotonic Reduction</i>	Yes	Yes	Yes	not necessary	-
<i>Shrinking from Boundary</i>	Yes	Yes	Yes	Yes	-
<i>Smoothness or Continuity</i>	(unknown)	(unknown) <sup>η</sup>	GC <sup>1</sup>	GC <sup>1</sup> +	GC <sup>1</sup> +
<i>User Controllable Degree of Roundness</i>	Yes	Yes	Yes	Yes	No
<i>Continuity in Roundness Control</i>	Yes	Yes	Yes	Yes	-
<i>Flexibility in Control</i>	Simple	Simple	Simple	Multi-dimensional	-
<i>Well-defined Limit Behaviour</i>	Yes	Yes	Yes	No	-
<i>Existence of Limit Rounding</i>	Yes	Yes	Yes	No	-
<i>Existence of Zero Rounding</i>	Yes	Yes	Yes	Yes	-

Figure 3.4. Fulfillment of the quality criteria for existing volume rounding methods

One should notice that the summary highlights the lack of a fast volume-rounding method with a relative simple roundness control. Our research motivation is therefore provided by the fact that a fast volume-rounding method does not yet exist.

<sup>α</sup> The execution speed varies depending upon the choice of rendering primitive. Wunsche's polygonisation method, for example, can be much faster than the ray-tracing method proposed in the original paper [Lobb96].

<sup>β</sup> Fast on simple joint but the execution speed dramatically decreases when the complexity of input increases.

<sup>δ</sup> Usually undefined when trim curve is not on the surface of concern. i.e. too heavily rounded.

<sup>ε</sup> The whole volume vanishes with very heavy rounding. The volume may break into separate components in some circumstance.

<sup>φ</sup> Same as convolutional smoothing.

<sup>γ</sup> Object vanish if the rolling ball does not fit inside the volume.

<sup>η</sup> Same as convolutional smoothing in general but may produce visually unsmooth artefacts with some input configuration.



### 3.5 Chapter Conclusion and Summary

This chapter has introduced three main classes of volume rounding or related methods. They are implicit blending methods, parametric blending methods and recursive mesh subdivision methods.

Implicit blending methods define the rounding in the form of implicit surfaces. Unarguably, they are the simplest and the most elegantly defined in comparison to other classes of methods. Ironically, their implementations are the most computationally expensive.

Parametric blending methods represent volume rounding with the most complex controls. Rounded regions are described in the form of parametric surfaces. The exact trim lines or curves can be specified both automatically and manually. They are then joined smoothly with parametric surfaces. However, the high controllability does not come without a cost: they are harder to define, mathematical complex and slow to compute.

Recursive mesh subdivision methods are not volume rounding methods as such, in the sense that they do not produce rounded volumes with user specified roundness. They represent a class of fast methods that generate smooth surfaces from a control mesh. They serve as the basis of the polyhedron bevelling methods that will be presented in the remainder of this thesis.

The comparison between the various methods is summarised in figure 3.4, in which several existing key rounding methods are tabulated against the quality criteria developed in the previous chapter. The summary highlights the lack of a fast volume-rounding method with a relative simple roundness control.



## Chapter 4 Recursive Mesh Subdivision

Polyhedron bevelling is based heavily on recursive mesh subdivision methods especially the Catmull-Clark and Doo-Sabin refinements. Hence in this chapter we will cover in more detail how recursive mesh subdivision methods work.

First we will discuss how uniform B-spline surface refinement generates a B-spline surface from a series of repetitive refinement processes. The Catmull-Clark [CC78] surface, which is a generalisation of the uniform B-spline surface refinement scheme, will then be covered. Finally we will cover the smoothness condition for a Catmull-Clark surface and an improvement over Catmull-Clark refinement which was suggested by Doo and Sabin [DS78].

Doo-Sabin refinement scheme is used as a basis of our volume-rounding methods.

### 4.1 Uniform B-Spline Surface Refinement

Recursive mesh subdivision stems from the idea that a B-spline surface can be obtained by repetitively applying a refinement to its rectangular control mesh. At each level of refinement, a set of new control points is calculated based on a weighted sum of the original control points.

We will see how the refinement is obtained by considering a quadratic B-spline surface.

A tensor product quadratic B-spline surface can be expressed in matrix form as:

$$\mathcal{S}(u,v) = UMG^tV^t$$

where  $M$  is the basis matrix for a quadratic B-spline and  $G$  is the geometric constraint matrix.

$$M = \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \quad G = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

Each element  $p_{ij}$  in  $G$  is one of the control points arranged on a topologically rectangular patch according to the subscripts depicted in figure 4.1(a).

$U$  and  $V$  are the power bases and are given as:

$$U = [u^2 \ u \ 1] \text{ and } V = [v^2 \ v \ 1], \text{ where } 0 \leq u, v \leq 1.$$

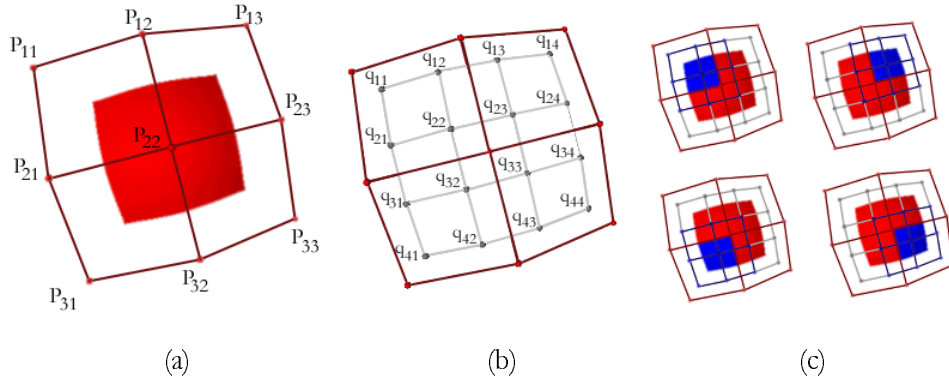


Figure 4.1. (a) A quadratic B-spline patch with its control mesh (b) The control mesh after a step of refinement (c) The four sub-patches generate B-spline surfaces. Each of them is equivalent to a quarter of the B-spline surface generated from the initial control mesh.

We will first define two terms that will be used later: the edges and faces of a mesh. Edges are straight lines connecting two adjacent control points in a row or a column of the geometric constraint matrix. Faces are area bounded connected edges that do not contain any other faces inside.

In the next step of the refinement, a control patch with 16 control points  $q_{ij}$  is generated from the original 9-point control mesh. See figure 4.1(b). Four B-spline surfaces can then be defined with the geometric constraint matrices given by:

$$G_0 = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \quad G_1 = \begin{bmatrix} q_{12} & q_{13} & q_{14} \\ q_{22} & q_{23} & q_{24} \\ q_{32} & q_{33} & q_{34} \end{bmatrix}$$

$$G_2 = \begin{bmatrix} q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \\ q_{41} & q_{42} & q_{43} \end{bmatrix} \quad G_3 = \begin{bmatrix} q_{22} & q_{23} & q_{24} \\ q_{32} & q_{33} & q_{34} \\ q_{42} & q_{43} & q_{44} \end{bmatrix}$$

The whole idea of refinement is that each of the B-spline surfaces defined by the geometric constraint matrices  $G_0$  to  $G_3$  is a quarter of the B-spline surface generated by  $G$ . See figure 4.1(c). Hence in each step of the refinement, four smaller quadratic B-spline control meshes are generated which together yield the same surface defined by the original control mesh. This together with the convex hull property<sup>1</sup> of B-spline shows that, by recursively applying the refinement process, the control mesh will converge to the B-spline surface itself at the limit.

The remaining question is how to calculate the set of new control points in each refinement.

---

<sup>1</sup> All points on the B-spline surface lie within the convex hull of its control patch.

Each refined sub-patch must be a quarter of the B-spline surface defined by the original control patch. For example, the quarter B-spline surface defined by  $G$ , where  $0 \leq u, v \leq \frac{1}{2}$  will be the same as the B-spline surface defined by  $G_0$ .

Hence we have:

$$\begin{bmatrix} \left(\frac{u}{2}\right)^2 & \frac{u}{2} & 1 \end{bmatrix} M G M^t \begin{bmatrix} \left(\frac{v}{2}\right)^2 & \frac{v}{2} & 1 \end{bmatrix} = U M G_0 M^t V^t$$

or

$$U S M G M^t S V^t = U M G_0 M^t V^t, \text{ where } S = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Since  $M$  is invertible,  $G_0 = [M^{-1} S M] G [M^t S M^{-t}] = H_0 G H_0^t$ , where

$$H_0 = M^{-1} S M = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix}$$

Therefore, we have

$$G_0 = \frac{1}{16} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

By simple algebra, we can obtain a linear map between the set of new and old control points as shown below:

$$\begin{bmatrix} q_{11} \\ q_{12} \\ q_{13} \\ q_{21} \\ q_{22} \\ q_{23} \\ q_{31} \\ q_{32} \\ q_{33} \end{bmatrix} = \frac{1}{16} \begin{bmatrix} 9 & 3 & 0 & 3 & 1 & 0 & 0 & 0 & 0 \\ 3 & 9 & 0 & 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 9 & 3 & 0 & 3 & 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 9 & 3 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 3 & 9 & 0 & 0 & 0 & 0 \\ 0 & 3 & 1 & 0 & 9 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 3 & 0 & 3 & 1 & 0 \\ 0 & 0 & 0 & 3 & 9 & 0 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 9 & 3 & 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix}$$

Equations for the rest of the control points can be obtained by considering the other three sub-patches. However, because of the symmetry of the B-spline basis, we will not present the details here.

Each new control point is given as a weighted sum of only four of the original control points. It can be verify that these four control points form a single face in the original control mesh, onto which the new control point lies. The weighting for the new control point are given as  $\frac{9}{16}$  for the corresponding (nearest) control point,  $\frac{3}{16}$  for the two adjacent control points and  $\frac{1}{16}$  for the opposite control point.

Refinements that generate higher order B-spline surfaces are also possible. Cubic refinement was presented in [CC78].

## 4.2 Catmull-Clark Refinement

The refinement rules presented in the last section are based on the assumption that the initial control mesh is rectangular. However, they can be generalised to work for any arbitrary rectangular mesh. Catmull and Clark [CC78] generalised the idea further and defined refinement rules which work for any arbitrary topology for both the cubic and quadratic case. Since only the quadratic formulation is employed in this thesis, we will cover only the quadratic refinement rules here.

Like the quadratic uniform B-spline surface refinement, Catmull-Clark refinement generates a new vertex for each control point on each face. However, the way to calculate the new vertex is generalised to work for faces that have arbitrary number of vertices. The formula is given as the average of the face point, two edge points and the vertex point. These points are defined as:

- The face point is the centroid of the face of concern.
- The two edge points are the midpoints of the edges that are incident on the original control point.
- The vertex point is the original control point.

Hence for an  $n$ -sided face, with the vertices labelled from  $p_1$  to  $p_n$  counter-clockwise, the formula for the  $i^{\text{th}}$  new vertex,  $q_i$ , can be written as:

$$q_i = \frac{1}{4} \left( \frac{\sum_{j=1}^n p_j}{n} + \frac{p_i + r_i}{2} + \frac{p_i + s_i}{2} + p_i \right)$$

where

$$r_i = \begin{cases} p_{i-1} & i \neq 1 \\ p_n & i = 1 \end{cases} \quad \text{and} \quad s_i = \begin{cases} p_{i+1} & i \neq n \\ p_1 & i = n \end{cases}$$

The  $r_i$  and  $s_i$  simply denote the previous and the next vertex in the faces. The extra complication is due to the cyclic nature of the labelling.

The equation is, more commonly, written as a weighted sum of the initial control points:

$$q_i = \sum_{j=1}^n W_{ij} p_j$$

where the weighting function  $W_{ij}$  is given as<sup>2</sup>:

$$W_{ij} = \begin{cases} \frac{4n+2}{8n} & , |i-j|=0 \\ \frac{n+2}{8n} & , |i-j|=1 \quad \text{or} \quad |i-j|=n-1 \\ \frac{2}{8n} & , |i-j|>1 \quad \text{and} \quad |i-j|<n-1 \end{cases}$$

The set of generated vertices are then reconnected to form the new control mesh. The way to reconnect the mesh is best illustrated as the construction of 3 different types of faces, namely F-faces, E-faces and V-faces.

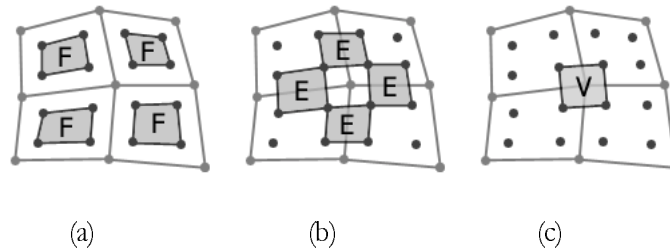


Figure 4.2. Three different types of faces in quadratic Catmull-Clark refinement

- Each F-face is constructed by linking up the new vertices generated from all vertices of one of the original faces. See figure 4.2(a).
- For each common edge in the original mesh, a four-sided E-face is constructed by linking up the two corresponding edges on each of the F-faces generated from the two original faces that share the edge. See figure 4.2(b).

<sup>2</sup> The formula is slightly different from that in the original paper, which ignored the cyclic nature of the labelling for the sake of simplicity.

- For each closed vertex of the original mesh, a V-face is constructed by linking up all the new vertices generated from that vertex. See figure 4.2(c).

### 4.3 Smoothness of Catmull-Clark Surface and Extraordinary Points

We have already proved that the uniform B-spline subdivision scheme produces a B-spline surface at the limit. The resultant surfaces have  $C^1$  and  $C^2$  continuity for the quadratic and cubic subdivision schemes respectively. Hence the Catmull-Clark surface will be smooth except at a finite number of extraordinary points.

For the quadratic formulation, the extraordinary points are found at the centroids of all  $n$ -sided original faces and all V-faces of  $n$ -valence vertices where  $n$  are not 4.

Fortunately, the number of extraordinary points is fixed after the first refinement step. The key is to realise that all E-faces are 4-sided. Every non-4-sided face will give rise to one and only one non-4-sided F-face. Four-sided E-faces however, always isolate this F-face. Hence the number of extraordinary points remains constant from iteration to iteration. On the other hand, a non-4-sided V-face will be generated if the number of edges incident on a vertex is not 4. However this situation can only happen on the first step of refinement, as every vertex on the refined mesh is always connected to an F-face, two E-faces and a V-face. Hence the valence of all vertices after the first refinement step is exactly 4.

The smoothness over extraordinary points is only  $C^0$ . Doo and Sabin [DS78] suggested a remedy, as described in the following section.

### 4.4 Doo-Sabin Refinement

A Catmull-Clark surface exhibits a problem in smoothness over a finite number of extraordinary points. Doo and Sabin [DS78] suggested improvements to both the Catmull-Clark cubic and quadratic refinement schemes. A Doo-Sabin quadratic surface exhibits ideal behaviour, which means it achieves  $GC^1$  continuity over extraordinary points. That means the Doo-Sabin quadratic surface is  $GC^1$  continuous globally while achieving a local  $C^1$  continuity except at extraordinary points.

The Doo-Sabin quadratic subdivision scheme is very much like the Catmull-Clark quadratic except in the way used to obtain new vertices. In a Doo-Sabin quadratic, new vertices can again be written as a weighted mean of old vertices:

$$q_i = \sum_{j=1}^n W_j p_j$$

but the weighting function  $W_j$  is modified to:

$$W_j = \begin{cases} \frac{n+5}{4n} & i = j \\ \frac{3 + 2\cos\left(\frac{2\pi(i-j)}{n}\right)}{4n} & i \neq j \end{cases}$$

The new computed vertices are then reconnected using the same method as in the quadratic Catmull-Clark refinement.

Doo and Sabin stated in their paper that:

*...the ideal behaviour may not be significantly better than the Catmull quadratic in normal use [DS78 pp360].*

However, we can see in figure 4.3 that Doo-Sabin quadratic surface is significantly smoother than Catmull-Clark quadratic over extraordinary points.

Another reason for using Doo-Sabin refinement is that although the weighting function used is somewhat more computationally expensive than the one in the Catmull-Clark quadratic, it can be pre-computed for common values of  $n$ ,  $i$  and  $j$  and table lookup can be used to retrieve those pre-computed values.

## 4.5 Chapter Conclusion and Summary

Polyhedron bevelling, to some extent, differentiates itself from other volume-rounding methods by its fast execution speed. The fast execution speed arise from the fact that it is based on recursive subdivision methods, which themselves are fast and efficient.

This chapter gave a deeper insight into several recursive subdivision schemes namely uniform B-spline refinement, the Catmull-Clark refinement [CC78] and the Doo-Sabin refinement [DS78]. The focus was primarily on the mathematical formulation and properties.

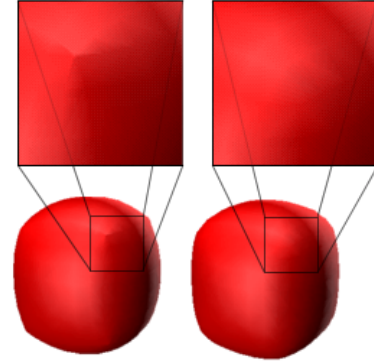


Figure 4.3. Catmull-Clark (left) and Doo-Sabin (right) quadratic surface defined with a unit cube control mesh.





## Chapter 5 Polyhedron Bevelling by Mesh Pre-partitioning

In this chapter we describe our first approach of volume rounding: the *mesh pre-partitioning* polyhedron bevelling method. The method is based on the Doo-Sabin [DS78] mesh subdivision refinement scheme to obtain a smooth surface with a control mesh. The major contribution of the method is that it allows a *user-defined degree of roundness* by providing an extra parameterisation.

The idea of mesh pre-partitioning is to modify the control mesh before applying the recursive mesh subdivision. The desired resultant surface is achieved after the subdivision. The beauty of this pre-processing approach is that the resultant surface is unaltered after the final mesh subdivision phase. The final result is therefore a Doo-Sabin surface. As a result, it inherits the well-defined limit behaviour and smoothness from a Doo-Sabin surface.

Our discussions in this chapter are limited to polyhedra with only convex faces. Any concave polygonal faces should be cut into convex polygons in advance. We will come back to the concavity problem in Chapter seven.

### 5.1 Architecture

The architecture of mesh pre-partitioning polyhedron bevelling is shown in figure 5.1.

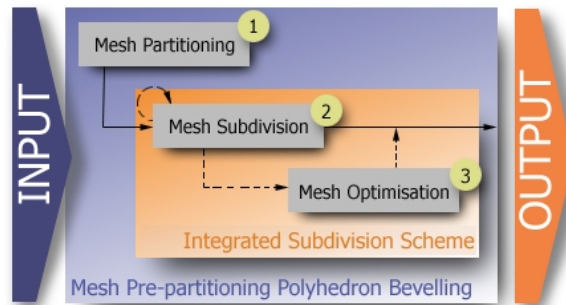


Figure 5.1. Architecture of mesh pre-partitioning polyhedron bevelling

In general, three separate phases can be identified as follows:

1. *Mesh partitioning* – The purpose of this phase is to introduce extra partitioning within the control mesh so that the user-specified roundness can be achieved when the subsequent subdivision is applied. Desired planar areas are buffered by the partitioning and hence further mesh subdivision will not affect their planarity.

2. *Mesh subdivision* – Doo-Sabin refinement is applied to the partitioned control mesh generated from the previous phase. In theory, this process is repeated ad infinitum until the limit surface is obtained. In practice, two or three refinement steps are usually enough to achieve a reasonably smooth polygon mesh. The recursive nature of this phase is indicated with a dotted loop as shown on figure 5.1.
3. *Mesh optimisation (optional)* – A vast number of unnecessary faces is produced in the previous phase. The number of faces grows very rapidly at each step of the recursion. For practical reasons, over-detailed surface representation must be approximated using a coarser polyhedron. Optional mesh simplification or optimisation reduces the number of primitives presented and enhances the rendering speed. They unfortunately add to the cost of the rounding process. Examples of general-purpose mesh optimisation techniques can be found in [HDD93][KT96]. They are, however, outside the scope of the thesis.

Over-applying mesh subdivision and then simplifying the resultant mesh is neither natural nor efficient. As an improvement, pre-subdivision mesh optimisation will be discussed in the next chapter. The optimisation method combines phase two and three of the algorithm as the so-called *integrated subdivision scheme*. The modification avoids some of the unnecessary subdivisions and hence increases the performance of the rounding method and also reduces fragmentation.

## 5.2 Two-dimensional Analogy

A better understanding of mesh pre-partitioning polyhedron bevelling can be achieved via a two-dimensional analogy, which is both simple and robust in comparison to the three-dimensional case.

The Chaikin refinement scheme produces a quadratic B-spline curve from a set of control points. It is, therefore, a logical equivalent of the Doo-Sabin refinement scheme for the mesh subdivision phase in two dimensions. The Chaikin refinement method is briefly covered below. It is followed by mesh pre-partitioning polyhedron bevelling in two-dimensions. Finally, the problem with the mesh pre-partitioning approach is identified and suggested remedy will be presented.

### 5.2.1 Chaikin Refinement Scheme

The Chaikin refinement scheme produces a quadratic B-spline curve in the limit. The proof is given by Riesenfeld [Ries75].

Figure 5.2(a) shows an example of a control polygon and the resultant quadratic B-spline curve. The input of the Chaikin refinement can also be a set of control points. In that case, the resultant curve will be open instead of closed.

Like Catmull-Clark or Doo-Sabin refinement, the method is recursive. At each step of the recursion, a better piecewise linear approximation of the resultant quadratic B-spline curve is produced. Two new control points,  $p_1'$  and  $p_2'$ , are generated between each of the two consecutive original control points, as illustrated in figure 5.2(b). The formulas are given as:

$$p_1' = \frac{3}{4}p_1 + \frac{1}{4}p_2 \text{ and } p_2' = \frac{1}{4}p_1 + \frac{3}{4}p_2$$

The original control points are then discarded and the new control points in each of the successive segment form the control polygon of the next iteration.

Figure 5.2(c) shows the resultant control polygons after the first and second iteration. In the limit, it will converge to the quadratic B-spline curve defined by the original control polygon. As shown in the figure, the control polygon after the second iteration already gives a very close approximation to the actual quadratic B-spline curve shown in figure 5.2(a).

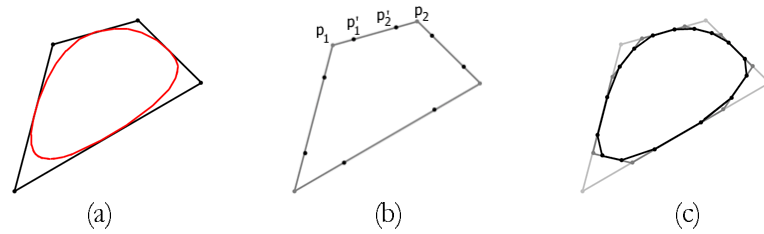


Figure 5.2. The Chaikin refinement scheme. (a) The control polygon and the resultant quadratic b-spline curve (b) New control points generated on the first iteration (c) Control polygons from the first two iterations superimposed on the original polygon.

### 5.2.2 Mesh Pre-partitioning Polyhedron Beveling in Two Dimensions

The preservation of straight-line segments of a polygon is obviously the two-dimensional analogue of the preservation of planar regions of a polygonal face. Hence, the two-dimensional problem can be described as fitting a quadratic B-spline curve over the linear segments to be preserved, each of which belongs to a single edge of a polygon. The segments to be preserved in a polygon are marked in red in figure 5.3(a).

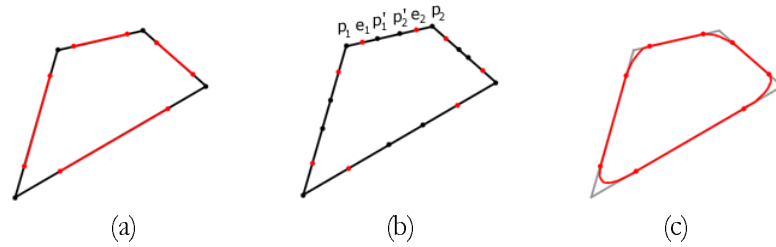


Figure 5.3. Mesh pre-partitioning polyhedron beveling in 2-dimensions. (a) Original polygon with the desired final straight-line segments marked red (b) Extra control points are added so that the endpoints of the final straight-line segments are at the midpoints of those corner segments (c) A quadratic b-spline curve defined on the pre-partitioned control polygon.

Before we describe the solution, we will briefly review three important properties of B-spline curves: locality, midpoint interpolation (quadratic B-spline only) and the convex hull property. The locality property ensures that each point on an  $m$ -order B-spline curve is defined solely by  $(m+1)$  instead of all of the control points. Take a quadratic B-spline as an example; the position of each portion of the quadratic B-spline curve is influenced exclusively by three consecutive control points. The midpoint interpolation property suggests that the portion of the curve controlled by the three control points starts at the midpoint of the first and the second control points and ends at the midpoint of the second and the third. Finally, the convex hull property guarantees that all points of that portion are located inside the convex hull created by the three control points.

Equipped with these three properties, we now describe and explain our solution to the problem mentioned. As indicated earlier, the essence of the mesh pre-partitioning polyhedron bevelling is to modify the control mesh such that the desired shape is obtained after applying the mesh subdivision. In a two-dimensional case, we simply add extra control points to partition the original edge of the control polygon before applying the Chaikin refinement. The term ‘*edge*’ refers to the straight-line segment between the two consecutive control points.

Figure 5.3(b) shows the segment  $e_1e_2$ , defined as the final straight-line segment to be preserved on  $p_1p_2$ . On each edge,  $p_1p_2$ , of the initial control polygon we add two new control point  $p_1'$  and  $p_2'$  such that  $e_1$  is located at the middle of  $p_1$  and  $p_1'$  and  $e_2$  is located at the middle of  $p_2$  and  $p_2'$ . Thus two new control points are simply given as:

$$p_1' = e_1 + (e_1 - p_1) \quad \text{and} \quad p_2' = e_2 + (e_2 - p_2)$$

The new control polygon after partitioning will hence have three times the number of control points of the original control polygon. They are shown as the black dots in figure 5.3(b). Each segment of the original control polygon is divided into three parts: one internal segment and two corner segments. Corner segments are the ones at both ends and the internal segment is the one in between.

Figure 5.3(c) shows the resultant quadratic B-spline curve after applying the Chaikin refinement scheme to the partitioned control polygon. The curve preserves the straight-line portions with the rounded corners smoothly joining those portions.

The following is the explanation of how the linear segment is preserved:

First, because of the midpoint interpolation property, the resultant quadratic B-spline curve interpolates all the midpoints of the corner segments of the partitioned control polygon. By construction, these midpoints are the endpoints of the straight-line segments to be preserved. Hence the resultant curve is a smooth quadratic B-spline curve which interpolates all the endpoints of those segments to be preserved.

Secondly, we need to show that portions of the curve in between those endpoints are actually straight-lines. Again by construction, there will be four collinear control points on every segments of the original control polygon: the two endpoints of the segment and the two new control points introduced in the partition. By the locality property of the B-spline,

the four consecutive control points will define a portion of the resultant quadratic B-spline curve starting from the midpoint of the first two to the midpoint of the last two. Since the four control points are collinear, the convex hull formed by the four control points will also be a straight-line. By the convex hull property, the portions of the curve in between those endpoints must therefore be linear.

### 5.2.3 Problem and Remedy in Two Dimensions

The method described in the last section will have a problem if the length of the portion to be preserved is very short in comparison to that of its original edge. The internal segments created by the partitioning will be flipped in such a case. Figure 5.4(a) depicts a partitioned control polygon with all the internal segments flipped, as emphasised by the numbering of the control points in the figure. Because of the flipped internal segments, the resultant quadratic B-spline curve is also flipped and does not end on the desired endpoints as shown in figure 5.4(b).

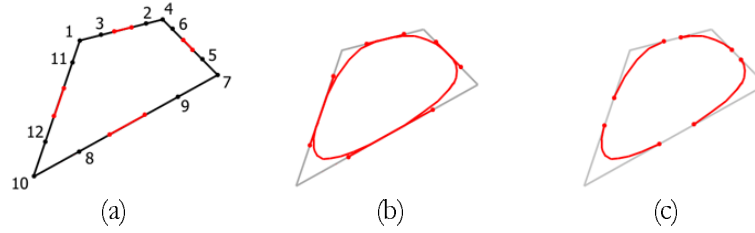


Figure 5.4. Artefacts produced by mesh pre-partitioning polyhedron beveling in two dimensions.

- (a) The desired final straight-line segments are shorter than half of the original edges which lead to overlapping of control polygon segments (b) A quadratic b-spline defined with a control polygon with overlapping (c) Quadratic b-spline segments produced by only the corner segments of the control polygon.

It is not hard to show that the “*flipping edge*” problem occurs when the length of the portion to be preserved is less than half of the length of its original edge. Again for two consecutive control points  $p_1$  and  $p_2$  of the original control polygon, we call the two newly introduced control points  $p_1'$  and  $p_2'$  and the endpoints of the desired final straight-line segment  $e_1$  and  $e_2$ . The length of the internal segment will be given as:

$$\begin{aligned} \text{Length} &= \|p_2' - p_1'\| = \|(e_2 + (e_2 - p_2)) - (e_1 + (e_1 - p_1))\| \\ &= \|2(e_2 - e_1) - (p_2 - p_1)\| = 2\|e_2 - e_1\| - \|p_2 - p_1\| \end{aligned}$$

Hence the length of the internal segment will be positive (no flipping) if the length of the desired final straight-line segment  $(e_2 - e_1)$  is greater than or equal to half of the length of the original edge  $(p_2 - p_1)$ .

Fortunately, there are many ways to solve the problem. Perhaps the simplest one is to produce those rounded portions of the quadratic b-spline curve by subdividing only the corner segments of the partitioned control polygon as shown in figure 5.4(c). We can then

simply join up those rounded corners with straight-lines, since those missing portions of the curve are linear, as explained earlier.

The method works well on any arbitrary input configuration. The other beauty is that it produces a more efficient implementation with less fragmentation because of the elimination of unnecessary subdivision on the straight-line segments. This idea leads to the optimisation scheme of mesh pre-partitioning polyhedron bevelling.

### 5.3 Locality, Centroid Interpolation and the Convex Hull Property

The planar region preservation is based heavily on the three properties of quadratic B-spline curve introduced earlier. Unlike the Chaikin algorithm, which produces an exact quadratic B-spline curve, Catmull-Clark and Doo-Sabin surfaces with non-quadrilateral control mesh are only approximations to the quadratic B-spline surfaces. Hence, before moving to the rounding algorithm in three dimensions, this section confirms that locality, centroid interpolation and the convex hull property hold for Doo-Sabin refined surfaces.

The locality property can be easily confirmed by the fact that each new control point on the  $i+1^{\text{th}}$  iteration mesh is computed by a linear combination of vertices only from one and only one face in the  $i^{\text{th}}$  iteration mesh. Hence the location of any points on the limit surface are solely controlled by some subset of control points in the initial mesh.

For the midpoint interpolation property, we consider an  $n$ -sided face with vertices  $p_i$  and the corresponding face after one step of the Doo-Sabin refinement with vertices  $q_j$ , where  $i \in [1...n]$ .

For the Doo-Sabin refinement, we know that:

$$q_i = \sum_{j=1}^n W_{ij} p_j, \text{ where } W_{ij} = \begin{cases} \frac{n+5}{4n} & i = j \\ \frac{3 + 2\cos\left(\frac{2\pi(i-j)}{n}\right)}{4n} & i \neq j \end{cases}$$

It is well known that a bi-quadratic B-spline surface interpolates the centre of each rectangular grid cell on the control mesh. To show that it is also true for the Doo-Sabin refinement, we need to show that the centroid of each face remains the same after one step of refinement.

The centroid of a new face is given as:

$$= \frac{1}{n} \sum_{i=1}^n q_i = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n W_{ij} p_j$$

$$\begin{aligned}
&= \frac{1}{4n^2} \sum_{i=1}^n \left[ (n+5)p_i + \sum_{j=1}^n \left( 3 + 2\cos\left(\frac{2\pi(i-j)}{n}\right) \right) p_i - \left( 3 + 2\cos\left(\frac{2\pi(i-i)}{n}\right) \right) p_i \right] \\
&= \frac{1}{4n^2} \sum_{i=1}^n \left[ (n+5)p_i - 5p_i + \sum_{j=1}^n \left( 3 + 2\cos\left(\frac{2\pi(i-j)}{n}\right) \right) p_i \right] \\
&= \frac{1}{4n^2} \sum_{i=1}^n \left[ np_i - \sum_{j=1}^n 3p_i + \sum_{j=1}^n 2\cos\left(\frac{2\pi(i-j)}{n}\right) p_i \right] \\
&= \frac{1}{n} \sum_{i=1}^n p_i + \frac{1}{2n^2} \sum_{i=1}^n p_i s_i \\
&\quad \text{where } s_i = \sum_{j=1}^n \cos\left(\frac{2\pi(i-j)}{n}\right)
\end{aligned}$$

It can be concluded that  $s_i$  always sum to zero<sup>3</sup>. The result shows that the new face centroid stay unchanged after each step of refinement and therefore confirms the centroid interpolation property for the Doo-Sabin subdivision.

The convex hull property can be proven by the fact that the weighting function,  $W_{ij}$ , always lies in between 0 and 1 as well as the sum of the weighting  $\sum_{j=1}^n W_{ij} = 1$ .

With the three properties reconfirmed, we can now move to the explanation and description of the algorithm.

## 5.4 Simple Mesh Pre-partitioning Polyhedron Beveling

The objective of the mesh partitioning phase is to construct a partitioning of the mesh given a scalar value roundness control, such that the preserved planar regions of the final shape are reduced as the roundness control increases. This parameterised mesh partitioning phase can be logically separated into two steps:

1. Definition of the planar regions based on the given roundness control.
2. Partition the initial mesh so that the planar regions are preserved after mesh subdivision.

We will first look at how the planar regions are defined using a roundness control. Based on the discussion in Chapter two, we identified that the ratio between the surface area of the planar regions to the rounded regions is an intuitive measure of roundness of an object. We will define the *roundedness*,  $\rho$ , for our rounding operation as:

---

<sup>3</sup> Proven by *Mathematica*<sup>®</sup>

$$\rho = 1 - \frac{A'_p}{A_p}$$

where  $A_p$  and  $A'_p$  are the area of the planar regions on the original polyhedron and the rounded polyhedron respectively. Thus  $\rho$  is a value between 0 and 1, with 0 representing a non-rounded polyhedron and 1 representing a completely rounded polyhedron with no remaining planar areas. We will also assume the percentage reduction of planar area is the same for every polygonal face.

The constraint of monotonic reduction of planar regions suggests the shrunk version of the polygonal face should lie completely within the original face. Even then, there are numerous ways to define the preserved region, and there is not a single “correct” answer. We will assert only that the preserved region of an arbitrary  $n$ -sided convex polygon should also be an  $n$ -sided polygon that sits entirely within the original face.

We have confirmed earlier that the limit surface interpolates the centroid. Hence, for an arbitrary  $n$ -sided convex polygonal face, we can construct, by partitioning,  $n$  “corner faces” with their centroids located at corners of the preserved region as depicted in figure 5.5.

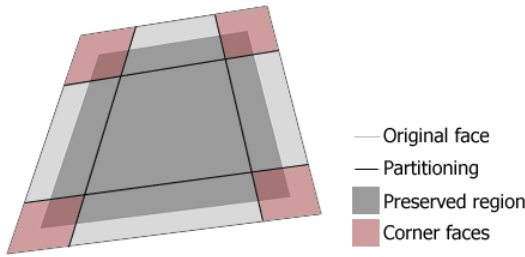


Figure 5.5. Partitioning of a convex polygonal face

This, however, still does not uniquely define the partitioning. As a result, we have gone through a series of trials and errors in order to find out the “optimal” partitioning.

The first “reasonable” approach defines the partitioning by simple displacing each edge of the polygonal face inwards by a certain amount such that resultant corner faces have their centroids interpolate the vertices of the shrunk face.

While this approach works well on small rounding with some input configurations, it suffers seriously from the stability problem in some other configurations. In addition to failing on concave faces, this partitioning scheme fails even with convex polygons with a very short edge as shown in figure 5.6.

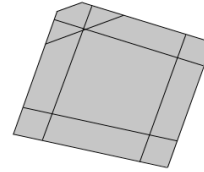


Figure 5.6. Problem with partitioning scheme using absolute displacement

The problem arises when the absolute displacement chosen is large in comparison to the length of an edge. This can easily be fixed by replacing the absolute displacement with a displacement that is relative to the length of the current edge. The modified partitioning scheme is described with a nomenclature shown in figure 5.7:



```

for each face  $F$  of original polyhedron
    for each edge  $e_i = (p_i, p_{(i+1) \bmod n})$  of  $F$ 
        Add vertex  $q_{2i} = p_i + \alpha (p_{(i+1) \bmod n} - p_i)$  to mesh
        Add vertex  $q_{2i+1} = p_{i+1} + \alpha (p_i - p_{(i+1) \bmod n})$  to mesh
    for each "edge"  $g_i = (q_{(2i-1) \bmod 2n}, q_{(2i+2) \bmod 2n})$ 
        Add to mesh the intersection point  $r_i$  of  $g_i$  and  $g_{(i-1) \bmod n}$ 
    Delete face  $F$  from mesh
    Add new face  $F' = \{r_0, r_1, \dots, r_{n-1}\}$  to mesh
    for each vertex  $p_i$  of  $F$ 
        Add new face  $\{p_i, q_{(i+1) \bmod n}, r_i, q_{2i}\}$  to mesh
    for each edge  $e_i = (p_i, p_{(i+1) \bmod n})$  of  $F$ 
        Add new face  $\{q_{2i}, r_i, r_{(i+1) \bmod n}, q_{2i+1}\}$  to mesh
    
```

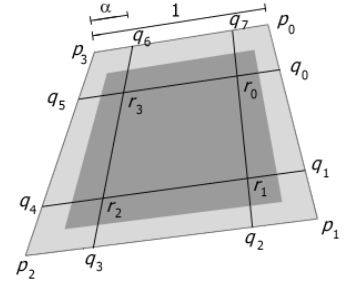


Figure 5.7. The nomenclature for a pre-partitioned face

The stability of this refined partitioning scheme is much better than that of the absolute displacement scheme, but it may still fail in constructing the partitioning if the roundness control,  $\alpha$ , is larger than a fundamental limit of roundness control  $\alpha_{\max}(F)$ , which varies from different input polygonal faces  $F$ . For example  $\alpha_{\max} = 0.5$  for a square face. This problem is related to the “flipping edge” problem as illustrated in the two dimensions, and remedy has been proposed. The remedy in the three dimensions is, however, more complicated. We will now accept the limitation and will come back to the solution later.

With the algorithm described, an  $n$ -sided polygonal face will be partitioned into  $2n+1$  faces of three types:

- *Centroid face* – Exactly one non-boundary  $n$ -sided centroid face will be produced, which is located at the centre and buffered up with the adjacent coplanar faces.
- *Corner face* –  $n$  corner faces are produced, each of which is located at the corner and sharing one of the initial vertices. See figure 5.5. Note also that they are always 4-sided by construction.
- *Edge face* –  $n$  edge faces are produced. They are located in between two corner faces along the initial boundary. Again they are always 4-sided by construction.

Both the corner and edge faces are also referred later as *boundary faces*.

Figure 5.8 shows an example of using the partitioning scheme to produce rounded cubes with different roundness.

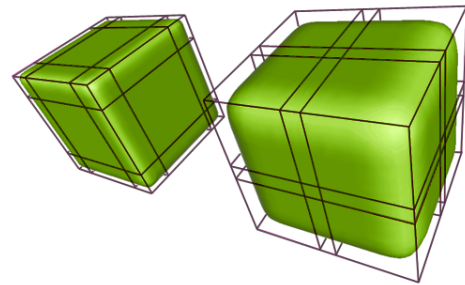


Figure 5.8. Different partitioning result in different roundness

## 5.5 Justification of the Pre-partitioning Scheme

Two justifications are required:

First, it must be shown that the preserved region is actually planar. This can be trivially proven by the locality and the convex hull properties as in the two-dimensional case. For any  $n$ -sided polygonal face,  $2n+1$  faces will be produced and by construction all those adjacent faces are coplanar. Since we have already mentioned that the two properties hold for Doo-Sabin subdivision, the same arguments can be applied as in the two-dimensional case. The (preserved) region of the resultant surface is solely controlled by the corresponding set of coplanar adjacent faces, and the region must lie within the convex hull of those coplanar faces and hence must be planar.

Second, as mentioned earlier, for an  $n$ -sided polygonal face, an  $n$ -sided planar region is expected. We have already shown that the limit surface of the Doo-Sabin refinement interpolates the centroid of each face. Hence we can tell that the resultant surface will land on the centroids of all the corner faces after partitioning. However, by construction,  $2n$  boundary faces are created for an  $n$ -sided polygonal face after partitioning. Consequently, in order to show that the preserved region is an  $n$ -sided polygon, we need to show that the centroid of each edge face is collinear with the two neighbouring corner faces. We must also verify that the boundaries in between the centroids of two successive boundary faces are actually linear. Although, unfortunately, both of the two criteria do not hold for arbitrary  $n$ -sided polygons, we are able to show that the straight boundary property of the preserved planar regions is satisfied for any arbitrary 3-or-4-sided polygonal faces, which are the two most common cases. The proof can be found in Appendix A1. For most other input configurations, although the straight boundary property is not necessarily satisfied in a mathematical sense, the curvature is not apparent visually.

## 5.6 Mapping from the Roundness Control Value to the Roundedness

Since we have claimed that the roundedness,  $\rho$ , is an intuitive measure for a rounding operation, we will attempt to relating the roundness control,  $\alpha$ , with it. Unfortunately, the mapping between the two is not trivial. There are in general two problems.

The first problem arises from the fact the computation of  $\alpha$  given  $\rho$  is very complicated. Even with the assumption that the straight boundary property is satisfied for all input faces, not only does the  $\alpha$  value depend on the number of size of a given face but also the actual geometry of the face.

We therefore believe that the cost involved to compute the exact  $\alpha$  value for every single face cannot justify the usefulness of the absolute accuracy. Since also the straight boundary property is only an approximation, a heuristic method is adopted. We assume that the mapping from  $\alpha$  to  $\rho$  depends only on the size of the polygonal face,  $n$ . Thereby we approximate the required  $\alpha$  value of an  $n$ -sided input face  $F$ , based on the assumption that  $F$  is a regular  $n$ -gon.

The mapping is therefore given as:

$$\alpha = \frac{1 - \sqrt{1 - \rho}}{1 + \cos \theta} \quad \text{where } \theta = \frac{(n-2)\pi}{n}$$

The proof of correctness is presented in Appendix A2.

The formula reveals the other limitation of the partitioning scheme. Since  $\alpha$  is bounded by the limit of 1 by construction, the fundamental limit on  $\rho$  is therefore:

$$\rho_{\max} = 1 - \cos^2 \theta$$

hence,

$$\lim_{n \rightarrow \infty} \rho_{\max} = 0$$

Thus we cannot, for example, obtain any roundedness on a circular face composed by infinite number of vertices using the partitioning scheme presented. For example, even with a very large  $\alpha$  value, the effect of rounding a polyhedral approximation of a cylinder, for which both end faces are circular, is unnoticeable. We will ignore the occurrence and assume input faces with a lot of vertices are cut into triangular or quadrilateral faces in advance.

The other problem that we encounter when mapping  $\rho$  to  $\alpha$  is that different  $\alpha$  values are required on different face of the input mesh. This is however not possible for the partitioning algorithm presented earlier. The subsequent mesh subdivision relies on the fact that vertices are shared along the boundary faces on both sides of every edge. Figure 5.9(a) illustrates that, when different  $\alpha$  values are used, boundary faces after partitioning will not share common vertices along the edges of the original mesh.

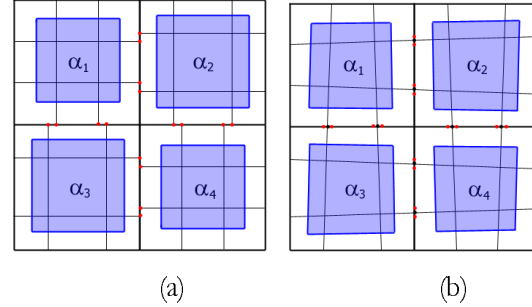


Figure 5.9. Using different  $\alpha$  values on different faces

Several approaches have been tried to deal with the problem. The simplest approach takes an  $\alpha$  value for each edge by averaging the  $\alpha$  values from the two adjacent faces. While this approach is straightforward and simple to implement, the major drawback is that the process alters the preserved planar regions. It is illustrated by the difference between the blue regions shown in figures 5.9(a) and (b). Various other more robust but complicated methods are possible.

Although the use of roundedness is, to some extent, more an academic interest rather than a practical requirement, methods that allow the use of different alpha values is important to the development of *non-uniform rounding*. However, since our study is primarily concerned with a global rounding operations, this side-issue will not be further discussed.

## 5.7 Results

With mesh pre-partitioning polyhedron bevelling, we are able to produce rounded volume with user specified degree of roundness. Figure 5.10 shows an image of five mesh pre-partitioning polyhedron bevelled cubes rounded with  $\alpha$  values range from 0.1 to 0.5.

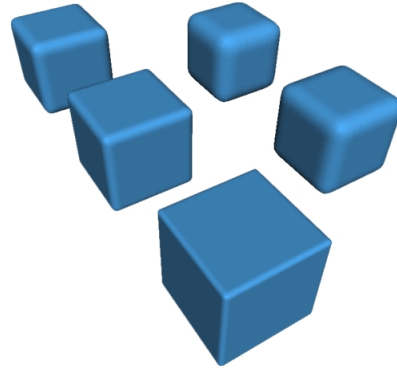


Figure 5.10. Five mesh pre-partitioning polyhedron bevelled cubes with different roundedness

Figure 5.11 shows an image of a stapler, smooth shaded, with the original non-rounded model aside for comparison. The stapler model is composed with five different separated components. Rounding on each component is performed separately with different roundness control value.

The algorithm also executes fast. Rounding of the entire stapler model takes less than 0.1 second with a Java implementation running on a 166MHz Pentium desktop computer performing three steps of Doo-Sabin refinement. This result is promising for our initial aim of real-time usage. Figure 5.12 depicts more examples of mesh pre-partitioning polyhedron bevelled objects. Note that all concave input faces, for example the stapler's base, are broken into smaller convex faces in advance.

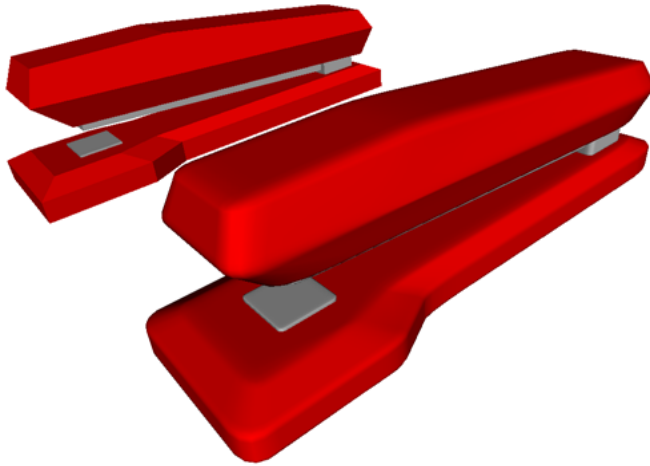


Figure 5.11. The final rounded stapler with the original in the background

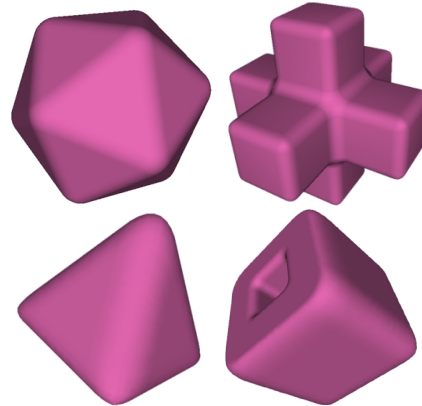


Figure 5.12. Other examples of mesh pre-partitioning polyhedron bevelled objects

## 5.8 Chapter Conclusion and Discussion

Perfect polyhedral models in computer graphics have a harsh appearance that detracts from the realism of the scene. Our first attempt at volume rounding – mesh pre-partitioning polyhedron bevelling – was presented in this chapter. The essence of the method is to introduce partitioning into the initial mesh such that the desired shape is produced after the applying recursive mesh subdivision. Users adjust a single scalar roundness control  $\alpha$ , which

alters the partitioning and hence indirectly controls the degree of roundness of the resultant shape.

The elegance of the approach is that the resultant surface is simply a limit surface of the recursive mesh subdivision process. Doo-Sabin refinement was selected as the mesh subdivision scheme and, as a result, mesh pre-partitioning polyhedron bevelled volumes will have  $GC^1$  continuity as if Doo-Sabin surfaces. The new rounding method produces excellent quality results on a wide range of models.

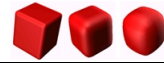
The algorithm also executes very quickly. Typical models can be rounded in less than a second. For example, the stapler model as shown in figure 5.11 takes less than 0.1 second for the entire rounding process. This promising result makes the method appropriate for the “just in time” modification we proposed earlier and numerous other real-time applications.

However, the method is not perfect. It suffers from several drawbacks.

Firstly, the algorithm does not work on concave polygonal faces, which must be cut into convex polygons in advance.

Secondly, the stability of the partitioning scheme is of concern. There is a fundamental limit on the roundness control  $\alpha_{\max}$ , which is not achievable with the partitioning scheme presented. The fundamental limit  $\alpha_{\max}$  of a face varies from one to another depending on the shape of the input face. For example, it is not possible to produce rounded cubes with  $\alpha > 0.5$  with the method. More seriously is the problem with the fundamental limit on the roundedness,  $\rho_{\max}$ . For an  $n$ -sided input polygonal face, the effect on the resultant roundness of  $\alpha$  diminishes as  $n$  increases. Therefore the method works well only for input faces with a small number of vertices.

Last but not least, the method produces far more fragmentation than optimal. General-purpose optimisation methods can be employed for better rendering speed. This, however, adds to the cost of rounding.



## Chapter 6 Optimisation Methods for Mesh Pre-partitioning Polyhedron Beveling

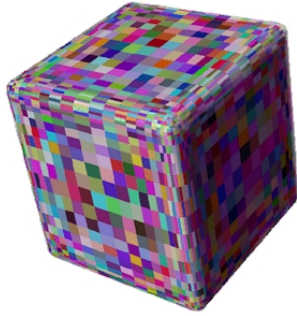


Figure 6.1. The high fragmentation problem illustrated by a rounded cube with randomly coloured faces

One obvious problem with mesh pre-partitioning polyhedron beveling is the high fragmentation of the resultant surface. The problem is highlighted in figure 6.1, which shows a cube rounded by the method with each face shaded with a random colour.

It is apparent that there are many redundant subdivisions. For example, subdividing two coplanar neighbouring faces will always result in a set of smaller coplanar faces, which can be more appropriately combined into a larger element.

This idea leads to the development of pre-subdivision optimisation, which avoids redundant subdivisions by removing appropriate faces right before the subdivision phase. Pre-subdivision optimisation not only reduces fragmentation but also speeds up the rounding algorithm since a huge amount of unnecessary subdivision is avoided.

### 6.1 Optimisation by Extracting Planar Regions

There is no doubt that the most distinguishing characteristic of a polyhedron-bevelled object is the large area of preserved planar regions. Since we know in advance that the regions are planar, any subdivision over those regions will be obviously redundant. An improvement can be achieved by extracting those regions before mesh subdivision and fitting a large polygon over each of the planar faces.

Wünsche [Wüns96] was the first to employ this idea in his Triage polygonisation. Flat regions are removed right at the beginning of the algorithm and thus give rise to a fast polygonisation scheme with low fragmentation.

Our refined algorithm is best summarised in the architectural diagram shown in figure 6.2.

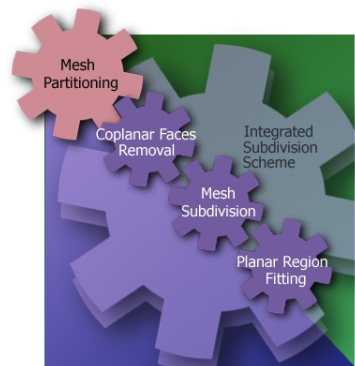


Figure 6.2. Architecture of optimisation with planar regions extracted

In the optimised method, the original mesh subdivision phase has been separated into three parts, namely: *coplanar neighbour removal*, *mesh subdivision* and *planar region fitting*.

The mesh subdivision phase is identical to the original. In the subsequent sections, we will concern ourselves with the coplanar neighbour removal and the planar region fitting phase.

### 6.1.1 Coplanar Neighbour Removal

As the name implies, the coplanar neighbour removal phase removes any faces that are coplanar with all their neighbours. Any subdivision on such a face will simply result in a large set of smaller coplanar faces.

By considering our mesh pre-partitioning scheme given in the previous chapter, one should notice that only the centroid faces after partitioning will be removed at this stage. Although there is an exception that some edge faces should be removed if two adjacent faces are coplanar even before mesh pre-partitioning, this situation is uncommon. Since the removal of the coplanar non-corner boundary faces is not essential to the “correctness” of the algorithm, we claim that it is more “economical” to simply ignore them and remove only the centroid faces.

Figure 6.3(a) shows a cube after mesh partitioning and coplanar neighbour removal. Three steps of Doo-Sabin subdivision are then applied and the result is shown in figure 6.3(b).

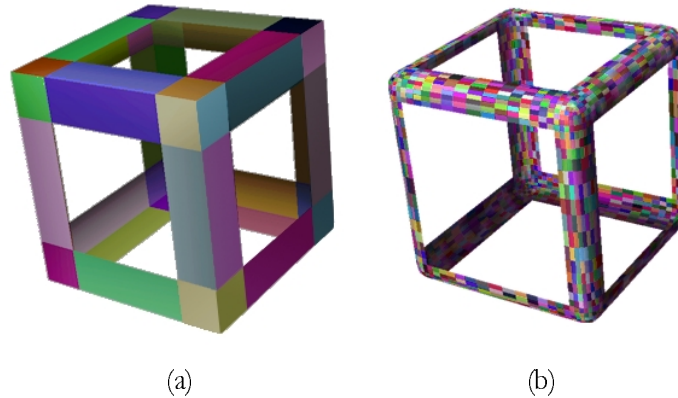


Figure 6.3. A cube: (a) after mesh pre-partitioning with all the centroid faces removed (b) after mesh subdivision; both with randomly coloured faces to illustrate the partitioning

### 6.1.2 Planar Region Fitting

It is clear that the remaining job is to fit the missing planar regions back into the mesh. The planar region fitting phase is nothing more than identifying all the missing holes from a polygon mesh and then replacing them with polygonal faces.

An example algorithm is:

```

Initialise  $E$  to be the set of all boundary edges of the mesh
    // i.e. all edges that belong to only a single face
while  $E$  is not empty
    Initialise  $e_0$  be any element in  $E$ 
     $F \leftarrow \{\}$ 
     $e \leftarrow e_0$ 
     $E \leftarrow E \setminus \{e_0\}$ 
    repeat
        Find  $e' = (e.p_2, \_) \in E$ 
            // where  $e.p_1$  and  $e.p_2$  is the first and second point of  $e$  respectively
            // and  $\_$  denotes any point
         $E \leftarrow E \setminus \{e'\}$ 
         $F \leftarrow F + \{e'.p_2\}$ 
         $e \leftarrow e'$ 
    until  $e = (\_, e_0.p_1)$ 
    Add  $F$  to mesh
    
```

Three points are worth emphasising. First, the sentence `Find  $e' = (e.p_2, \_) \in E$`  is essentially searching for a successor edge segment from the set of all boundary edges. This can be done in constant time, using any of a wide range of well known techniques. A dynamically-sized hashtable is used in our implementation. Secondly, the algorithm requires access to the set of boundary edges of the mesh. These are determined at almost no cost during the final step of mesh subdivision. Finally, one should also notice the algorithm presented is based on the assumption that the input volume is closed.

### 6.1.3 Result of the Optimisation by Extracting Planar Regions

Figure 6.4 shows the resultant cube rounded by the optimised method by first extracting planar region. The result is promising. Huge amount of fragmentation over those planar regions is completely eliminated.

In the case of a cube, original 3458 polygons are reduced to 2114 polygons with three steps of Doo-Sabin subdivision. The exact saving depends upon the input mesh and the required number of steps of mesh subdivision. Roughly, a 30-40% reduction in the number of polygons is attainable.

The optimised method also cuts the required execution time approximately in half by avoiding large amount of unnecessary mesh subdivisions.

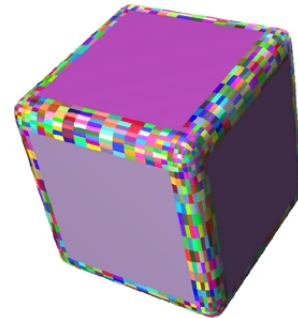


Figure 6.4. Resultant cube rounded with the optimised method by first extracting the planar regions



## 6.2 Optimisation by Corners Subdivision and Reconstruction

Even with the optimisation method of extracting planar regions, it can be seen in figure 6.4 that the curved “fillet” along the straight edges of the polyhedron is still unnecessarily fragmented: single polygon strips from one corner to the other would be more efficient.

This section presents a further improvement over the last method, which eliminates also the unnecessary fragmentation over those curved edges. Figure 6.5 shows the resultant cube rounded by such improved method.

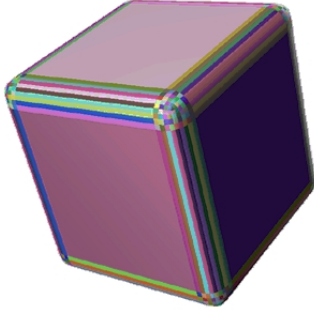


Figure 6.5. Resultant cube rounded by the optimised method that perform mesh subdivision only on corner faces

This further improvement is achieved by performing the mesh subdivision only on the corners of the original shape and fitting back the appropriate edges and planar faces afterwards.

One may notice that this method is only an approximation of the original mesh pre-partitioning method since, as explained before, the edge of the preserved planar region may not be linear on faces that are not 3 or 4-sided.

Our argument is that the straight edges of the preserved planar region are exactly what we want. It is, however, important to notice that the result (and hence the continuity) may not be the same as the original method.

### 6.2.1 Architecture of Corners Subdivision and Reconstruction

Figure 6.6 shows the immediate results for rounding a cube with the optimised method. Each of them corresponds to a single phase of the method. They are:

1. *Corner Extraction* – is an analogue of the mesh partitioning phase of the original method except that only the corner faces are retained in the process. Extra topological (or connectedness) information of the input shape is required for the reconstruction of the volume in the later stage. A set of disjoint corner meshes is created after this phase.
2. *Mesh Subdivision* – is the same as the original. Since only the corners are retained from the original shape, the rounding applies only to them. Unfortunately, not only does the mesh need to be refined, the topological information needs to be modified at each step of the refinement in order to reconstruct the volume after this phase.
3. *Edges Fitting* – is the first part of a *volume reconstruction* process. In this phase, the rounded edges are reconstructed based on the refined topological information.
4. *Planar Regions Fitting* – is the second part of a *volume reconstruction* process. It is the same as the one used in the first optimisation method except the number of vertices of each planar face is dramatically reduced.

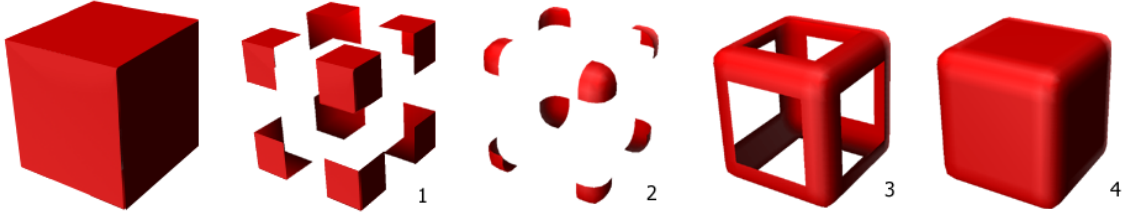


Figure 6.6. Immediate result of each step of the corners subdivision & reconstruction method

### 6.2.2 Volume Reconstruction of Corners Subdivision and Reconstruction

The reconstruction of the volume requires the topological information of the original mesh, which unfortunately will normally be lost after the mesh subdivision phase. In order to reconstruct the volume, this information must be stored elsewhere and updated during the process of mesh subdivision.

We will present in this section our implementation of the volume reconstruction process.

Our method starts with labelling each of the disjoint corner meshes during the corner extraction phase. The intention here is to set up virtual linkages between each of the disjoint corner meshes and its neighbour meshes.

First, we define any edge that belongs only to a single face as a boundary edge and any vertex that is connected to the boundary edges as a boundary vertex.

Artificial “*labels*” containing topological information are attached to each of the boundary vertices of each corner mesh. Each label of vertex  $p$  contains two pieces of information. The first one indicates the opposite corner meshes to which  $p$  should connect. The second is the relative indexing along all pairs of vertices between the two corner meshes. Since there are, by construction, three pairs of vertices on each edge of the original mesh, the initial indexing will either be 0, 1 or 2. Figure 6.7(a) shows an example configuration after the initialisation. The labelling of the boundary vertices sets up implicit linkage along different corner meshes, which makes volume reconstruction possible.

The label-initialising algorithm is given as:

```

 $M_A \leftarrow$  initial input mesh
 $M_B \leftarrow$  mesh after extracting corners
for each disjoint corner mesh  $M_i$  in  $M_B$ 
    Label  $M_i.p$  with  $p_i$  where  $p_i$  is the common vertices of  $M_i$  and  $M_A$ 
for each disjoint corner mesh  $M_i$  in  $M_B$ 
    for each boundary vertex  $p$  of  $M_i$ 
        if  $p$  is on some edge  $e = (M_i.p, q)$  or  $(q, M_i.p)$  of  $M_A$ 
            Label  $p$  with  $(q, 1)$ 
        else ( $p$  is on some face  $F = \{ \dots, p_{i-1}, p_i, p_{i+1}, \dots \}$  of  $M_A$ )
            Label  $p$  with  $(p_{i-1}, 0)$  and  $(p_{i+1}, 2)$ 
    
```

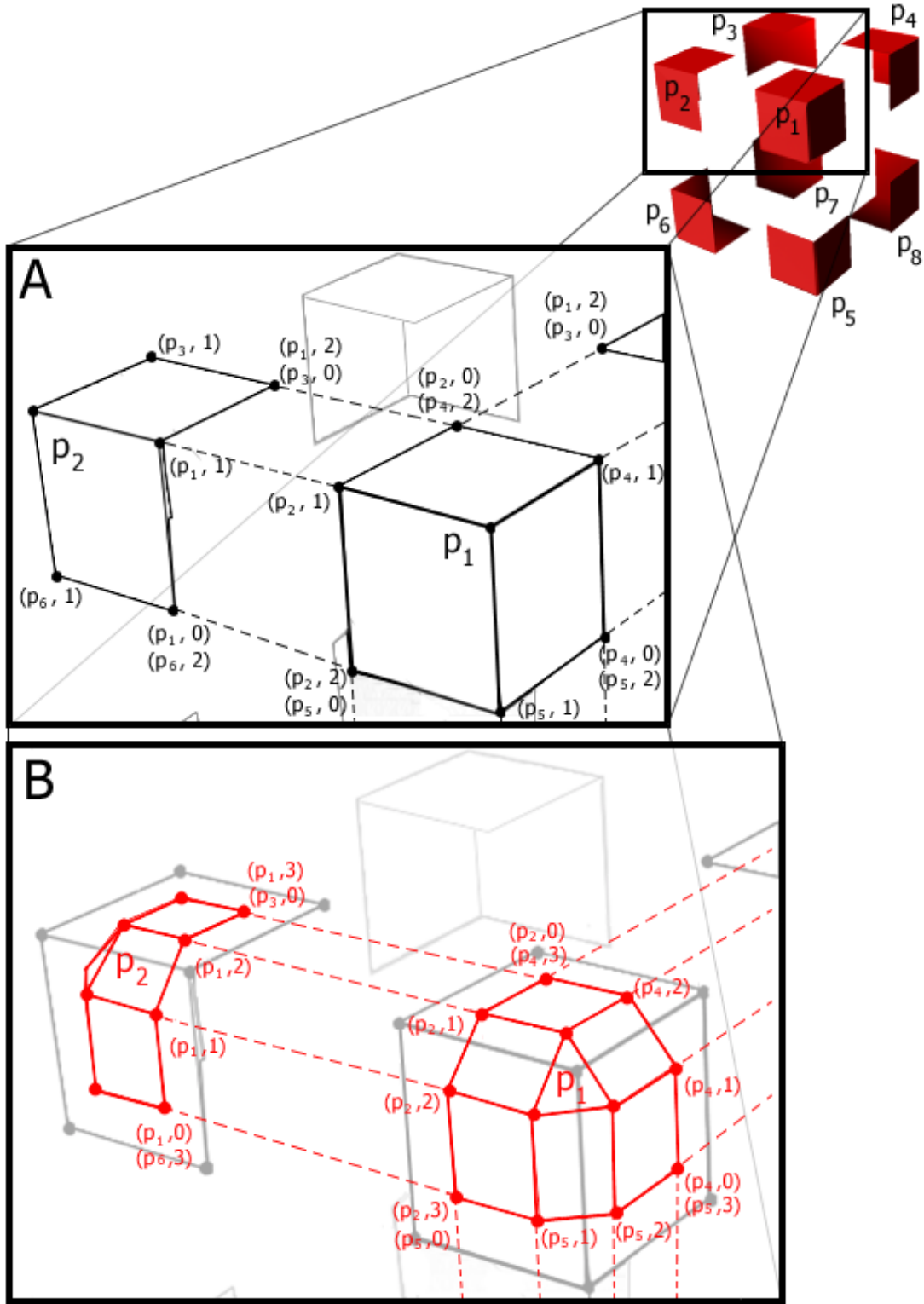


Figure 6.7. Topological information used in the corner subdivision and reconstruction method

The remaining issue is to update those labels at each step of mesh subdivision, so that the implicit linkages (relative indexing) are maintained during the process.

We now define parent-and-child relationships between *vertices* on the  $i^{\text{th}}$  and  $i+1^{\text{th}}$  refined mesh of mesh subdivision. For mesh subdivision, any vertex on the  $i+1^{\text{th}}$  refined mesh is given as some linear combinations of the vertices on the  $i^{\text{th}}$  refined mesh. We can therefore define a vertex  $p$  on the  $i^{\text{th}}$  refined mesh to be the parent of vertex  $q$  on the  $i+1^{\text{th}}$  refined mesh if  $p$  has the highest coefficient in the linear combination that defined  $q$ .

We can now define the updating process as:

```

for each face  $F$  on the mesh
  for each refined vertex  $q_i$  on  $F$ 
    for each label  $(x, n)$  attach on the parent  $p_i$  of  $q_i$ 
      if a label  $(x, n-1)$  is attached on the predecessor  $p_{i-1}$  of  $p_i$  in  $F$ 
        Attach label  $(x, 2n-1)$  to  $q_i$ 
      else
        Attach label  $(x, 2n)$  to  $q_i$ 
    
```

Figure 6.7(b) shows that the labelling of the boundary vertices is propagated on to the next step of the refined mesh. The topological information is passed through this parent-and-child hierarchy during the mesh subdivision phase. After the mesh subdivision phase, each edge of the original mesh will end up with some configuration as shown in figure 6.8. Appropriate long strips can be added according to the indexing to fit over each edge. Finally the process of fitting planar regions with large polygons is identical to the one described before.

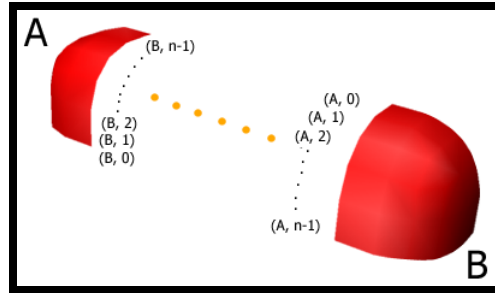


Figure 6.8. The topological labelling on each curved edge

### 6.2.3 Result of Corners Subdivision and Reconstruction

With this method, we are able to bring down the number of polygons from the original 3458 to only 602 in case of rounding a cube with three steps of Doo-Sabin refinement. In general, this optimisation method outputs only about 20% of the polygons of the original method. Even with the overhead required to manage the topological information for the volume reconstruction, the execution speed is approximately twice that of the first optimisation method presented in section 6.1 because of the huge reduction in unnecessary subdivisions.

For the stapler model shown in figure 5.11, an execution time of 0.02s is recorded on a 166MHz Pentium with this method. That is, approximately 4 – 5 times faster than the original method presented.

Furthermore, the corners subdivision and reconstruction method offers a better algorithmic stability than the original method. As stated before, there is a fundamental limit on the roundness control  $\alpha_{\max}$  which is not achievable in the original mesh pre-partitioning method. The problem arises as the centroid and edge faces will flip if the  $\alpha$  value is too large.

Since the new method performs rounding only on corner faces, it does not matter even if the two other types of faces are flipped. Figure 6.9 shows five cubes rounded by the new method with  $\alpha$  values range from 0.5 to 0.9, which are normally unachievable with the original method. In theory,  $\alpha_{\max}$  of 1 is achievable with the method if the mesh subdivision phase is recursive ad infinitum. In practice, a sufficient number of subdivision must be applied to pull the corner meshes apart so that they do not overlap each other. For example, rounded cubes with  $\alpha$  equals 0.87 and 0.93 are attainable with three and four steps of subdivision respectively.

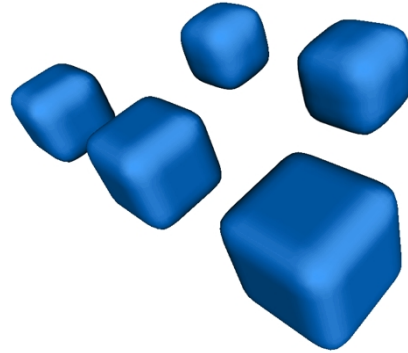


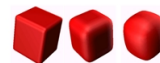
Figure 6.9. Five cubes rounded using the corners subdivision and reconstruction optimisation method with roundedness that normally are unachievable using the normal method

The benefits of the method, however, do not come without a price: The algorithm is considerably harder to implement especially maintaining the topological information in each step of mesh subdivision. The following section will be devoted to describing how the volume reconstruction works in the implementation level.

### 6.3 Chapter Summary and Conclusion

Mesh pre-partitioning polyhedron bevelling suffers from the problem of high fragmentation, some of which is unnecessary and avoidable. Two optimisation methods, namely *planar regions extraction* and *corner subdivision and reconstruction*, were developed and presented in this chapter. Not only have we achieved an approximately 30% and 80% cut in fragmentation from the two methods respectively, but also the execution speed roughly doubles and increases four times due to the fact that most unnecessary subdivisions are avoided.

An extra benefit of the corner subdivision and reconstruction method is that it offers better algorithmic stability by pushing the fundamental limit of the roundness  $\alpha_{\max}$  close to the optimal value of 1.



## Chapter 7 Polyhedron Beveling by Target-driven Subdivision

The preceding chapters focus on the mesh pre-partitioning polyhedron beveling method, which rounds volume with user specified roundness by introducing partitioning that indirectly controls the final shape. In this chapter, we will shift our focus to polyhedron beveling by target-driven subdivision or, simply, target-driven polyhedron beveling, which varies the actual mesh subdivision process to preserve planar regions in a more direct manner. Our discussion starts with why the former method is not appropriate in some rare situations, which led to the parallel development of both methods. At the end of the chapter, we will compare and contrast the two.

### 7.1 Rationale

The development of the target-driven approach is largely because of the unsatisfactory result of the mesh pre-partitioning approach in some circumstances. The major limitation of the mesh pre-partitioning approach is perhaps the assumption of convex input faces.

The concavity problem is faced by many polygon-based algorithms. Interestingly enough, a common approach taken by these algorithms is to simply ignore the concavity problem by assuming convex input or to rely on pre-processing methods such as triangulation.

Recursive mesh subdivision, unfortunately, takes the same approach to resolve the concavity problem. Although the author is unaware of any reference from literature, the problem can be realised from figure 7.1. The diagram shows a shape in wireframe processed by the Doo-Sabin subdivision method. An undesired lump can be clearly noticed on the concave corner in the enlargement. The formation of the lump is due to the fact that each vertex is moving toward the face centroids in each step of recursion, while the face centroids of a concave face may lie outside of the face itself.

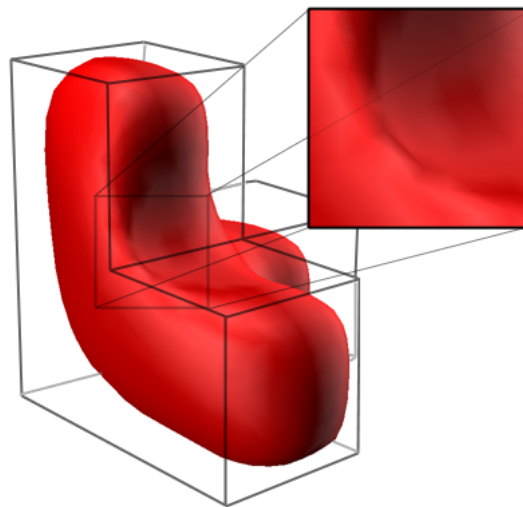


Figure 7.1. An example of artefacts produced on a input mesh with concave faces by the Doo-Sabin refinement scheme

The concavity problem encountered by recursive mesh subdivision, to some extent, justifies why we do not consider concave faces in the mesh pre-partitioning approach. Given that

recursive mesh subdivision is, by definition, one of the separated phases of mesh pre-partitioning polyhedron bevelling, it seems unlikely that any mesh pre-partitioning method can get around with the problem and, at the same time, keep the mesh subdivision phase unaltered.

Apart from that, the partitioning scheme presented in Chapter five fails on concave faces. Several alternative partitioning schemes were explored and examined, which involve special handling of concave corners. However, not only do those methods compromise the simplicity of the algorithm, but also none of the attempts is robust enough to handle arbitrary concave inputs.

All the above give rise to the development of target-driven polyhedron bevelling, which avoids the partitioning phase altogether.

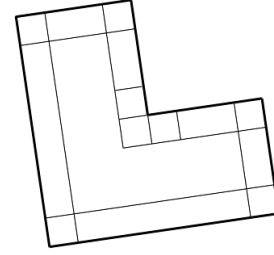


Figure 7.2. An example of partitioning of a concave face with one of the special partitioning scheme explored

## 7.2 Two-dimensional Analogy – A New Concept of “Centroids”

As mentioned in the previous section, recursive mesh subdivision fails on the concavity problem mainly because the centroid of a concave polygon may lie outside the polygon itself. The target-driven polyhedron bevelling method resolves the problem by introducing the concept of “*destination point*”. However, before going into the details, we will in this section reorganise the presentation of the two-dimension mesh subdivision given in Chapter five, but from a perspective of moving vertices towards the centroids at each step of recursion.

### 7.2.1 Mesh Subdivision as Moving Vertices Towards Centroids

Recall from Chapter five that the Chaikin refinement scheme computes the two new vertices on each successive edge by a linear combination of the two endpoints of the edge.

Let  $p_j^i$  be the  $j^{\text{th}}$  vertices on the  $i^{\text{th}}$  iteration mesh. Then the refinement rule given earlier can be written as:  $p_{2j}^{i+1} = \frac{3}{4} p_j^i + \frac{1}{4} p_{j+1}^i$  for all even vertices and  $p_{2j+1}^{i+1} = \frac{3}{4} p_{j+1}^i + \frac{1}{4} p_j^i$  for all odd vertices.

Now we define also  $c_j^i$  as the centroid of the  $j^{\text{th}}$  segment of the  $i^{\text{th}}$  iteration mesh, where the  $j^{\text{th}}$  segment is the one that has endpoints of  $p_j^i$  and  $p_{j+1}^i$ . See figure 7.3.

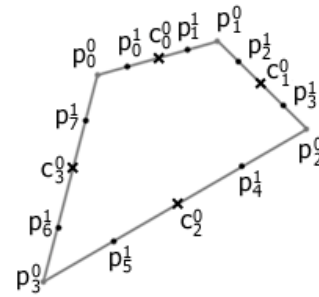


Figure 7.3. The nomenclature for a step of Chaikin refinement

The refinement rules can now be rewritten in terms of  $c_j^i$  as:



$$p_{2j}^{i+1} = \frac{1}{2} p_j^i + \frac{1}{2} c_j^i \quad \text{and} \quad p_{2j+1}^{i+1} = \frac{1}{2} p_{j+1}^i + \frac{1}{2} c_j^i$$

Note also that the centroids remains unchanged in each step of recursion. That is, any even centroid in the  $i+1^{\text{th}}$  iteration mesh will be the same as the corresponding centroid in the  $i^{\text{th}}$  iteration mesh:  $c_{2j}^{i+1} = c_j^i$ . The Chaikin refinement can hence be described as a process of moving vertices towards the centroids.

At each step of recursion, each vertex is moving half way towards the centroid of the corresponding segment. Examining the arithmetic sequence,  $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots = 1$ , confirms the vertex coincides with the centroid at the limit. We therefore call the centroid of each segment the destination point of the recursion.

### 7.2.2 Rounding by Moving Vertices Towards Target Points

We will now extend the concept of destination points of recursion to achieve polygon rounding with a user specification of roundedness.

In two dimensions, there are two endpoints on each segment of the mesh. We would like the two endpoints to move to two different final positions, so that a final linear segment with a user-defined length is preserved in between. We therefore define the target, a generalisation of centroid, of the  $j^{\text{th}}$  segment on the  $i^{\text{th}}$  iteration mesh as a tuple of two destination points:

$$t_j^i = (d_{0j}^i, d_{1j}^i)$$

where  $d_{0j}^0$  and  $d_{1j}^0$  are the two endpoints of the  $j^{\text{th}}$  desired final linear segment.

The target is a generalised concept of a normal “centroid”. Hence the even target is defined like the centroid as:

$$t_{2j}^{i+1} = t_j^i$$

while the odd target  $t_{2j+1}^i = (d_{02j+1}^i, d_{12j+1}^i)$  will have two coincident destination points that are the same as the normal centroid:  $d_{02j+1}^i = d_{12j+1}^i = \frac{1}{2} p_{2j+1}^i + \frac{1}{2} p_{2j+2}^i$  for all  $i > 0$ .

The refinement rules for both even and odd vertices also need to slightly alter to reflect the tuple nature of the target:

$$p_{2j}^{i+1} = \frac{1}{2} p_j^i + \frac{1}{2} d_{0j}^i \quad \text{and} \quad p_{2j+1}^{i+1} = \frac{1}{2} p_{j+1}^i + \frac{1}{2} d_{1j}^i$$

In this modified refinement scheme, each vertex is moving towards a corresponding destination point, which may or may not be the same as the centroid.



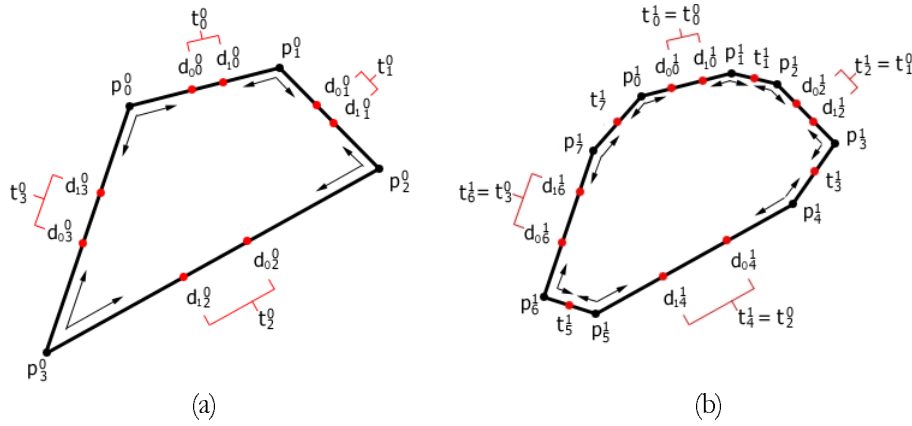


Figure 7.4. Target-driven polyhedron beveling in 2-dimensions: (a) initial polygon and nomenclature (b) after one step of subdivision

Figure 7.4 shows an example of how the modified refinement scheme performs rounding in two dimensions. The endpoints of the desired final linear segments are marked as red points in figure 7.4(a). Each vertex is moving towards the corresponding destination points in the directions shown by the arrows. Figure 7.4(b) shows the resultant polygon after the first step of subdivision.

We can see in the figure that all the even segments are portions of the initial segments and the even target (“centroids”) remain unchanged. Odd segments are inserted and odd targets are computed by the normal definition of centroid. The process then carries on recursively ad infinitum.

The same limit curve is obtained as the mesh pre-partitioning method with the pre-partitioning phase totally eliminated.

We will finish the discussion by summarising the three parts of the functional definition of the target, which is the key aspect of the extended method:

1. *Initial targets*,  $t_j^0$ , also known as the zero iteration targets, defines the preserved linear regions. In the case of  $t_j^0 = (c_j^0, c_j^0)$ , the resultant curve will be identical to that generated with the Chaikin method. This shows the Chaikin refinement is a proper subclass of the extended method.
2. *Even targets*,  $t_{2j}^i$ , are defined to be the same as the targets of the previous iteration:  $t_{2j}^{i+1} = t_{2j}^i$ . This guarantee the “centroid” interpolation property, which requires the “centroid” to stay unchanged in each step of refinement.
3. *Odd targets*,  $t_{2j+1}^i$ , are defined as a tuple of two coincident points, both of which are the normal centroid of the segment. This ensures the method to be identical to the Chaikin method in case of  $t_j^0 = (c_j^0, c_j^0)$  and thus guarantees the smoothness of the resultant curve.

### 7.3 Treatment in Three-Dimensions

We are now going to look at how the treatment can be mapped into three dimensions. The key part of the method is again the functional definition of the target. The target function of an  $n$ -sided polygonal face is extended to an  $n$ -tuple or  $n$ -vector:

$$t_j^i = (d_{0j}^i, d_{1j}^i, \dots, d_{n-1j}^i)$$

while  $t_j^i$  denotes the target of the  $j^{\text{th}}$  polygonal face of the  $i^{\text{th}}$  iteration mesh.

For an  $n$ -sided polygonal face,  $f_j^i = \{p_{0j}^i, p_{1j}^i, \dots, p_{n-1j}^i\}$ , each vertex  $p_{kj}^i$  is moving half way towards its corresponding destination point  $d_{kj}^i$ . The new control points are connected as if with the normal quadratic Catmull-Clark/Doo-Sabin refinement. Hence the concept of odd and even segments in two dimensions needs to map to the three face-types (F, E and V-faces) in quadratic Catmull-Clark refinement. At each level of recursion, F-faces, which correspond to even segments in the two dimensions, are computed based on the image of each of the original faces,  $f_j^i$ :

$$f_j^{i+1} = \{p_{0j}^{i+1}, p_{1j}^{i+1}, \dots, p_{n-1j}^{i+1}\} \quad \text{for } 0 \leq j \leq N_i$$

where

$$p_{kj}^{i+1} = \frac{1}{2} p_{kj}^i + \frac{1}{2} d_{kj}^i \quad \text{and}$$

$N_i$  is the number of faces in the  $i^{\text{th}}$  iteration mesh

Figure 7.5(a) shows an initial mesh with the destination points,  $t_j^0$ , defined in red, in which we define the boundary of the desired planar regions. Each vertex is moving half way towards the corresponding destination points and these define the control mesh of the next iteration. E-faces and V-faces,  $f_j^{i+1}$  for  $N_i < j < N_{i+1}$ , are constructed as if the Catmull-Clark refinement as shown in figure 7.5(b).

F-targets, like the even targets in two dimensions, remain as is:

$$t_j^{i+1} = t_j^i, \quad \text{for } 0 < j \leq N_i$$

while E-targets and V-targets,  $t_j^{i+1}$  for  $N_i < j < N_{i+1}$ , are computed by a specific functional definition, which we will discuss in the following section. E and V-targets are marked with question marks in figure 7.5(c). The new control points are continuously refined by moving steadily towards the corresponding destination points.

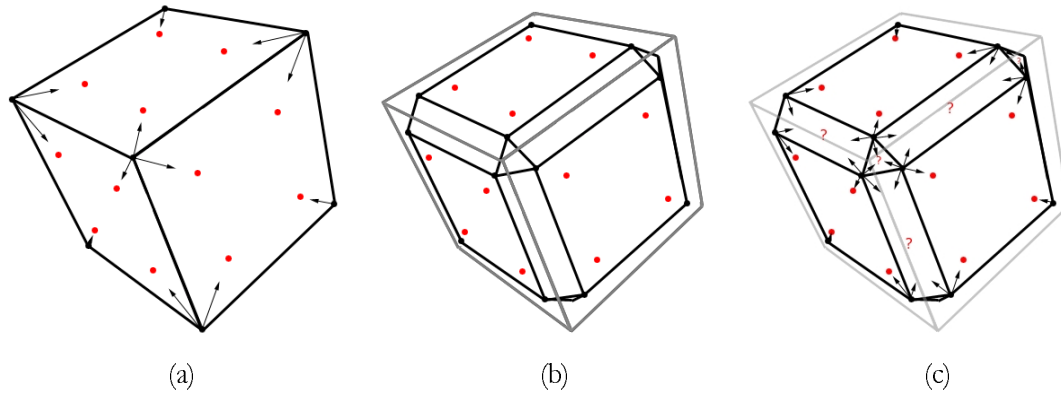


Figure 7.5. Polyhedron beveling by target-driven subdivision in 3D. (a) initial mesh with initial centroids defined. (b) the mesh after the first step of subdivision. (c) the E and V-targets are re-computed and the new control points are moving towards the corresponding destination points in the subsequent refinements

## 7.4 Defining the Target Function - The Skeletal Target

The behaviour of the target-driven polyhedron beveling method depends largely upon how functional definition of the targets within the generic framework. The function affects both the use of the rounding parameter and the continuity of the resultant shape.

We believe that two criteria are essential to the way the target of the polygonal face,  $p$ , is defined. They are:

1. All the destination points must lie within  $p$ .
2. The target must, to some extent, describe the topology of  $p$ .

It is not hard to notice the two criteria are very much reflecting the quality criteria of a rounding operation in Chapter two. For instance, the first criterion is directly related to the criteria of monotonic reduction of planar area.

Our research in this point is driven heavily by the second criterion. This led to the development of the concept of ‘*skeletal*’ target. This functional definition of target makes use of a simple variant of skeleton called *straight skeleton* [AA96]. We begin with a brief background on skeletons and an introduction to the straight skeleton. Most importantly how the straight skeleton is employed in the definition of the target function is presented afterwards. Since our primary focus here in this chapter is to describe and define the volume rounding method, the divide-and-conquer algorithm we employed for computing straight skeleton is included only in the Appendix B.



### 7.4.1 Background on Skeleton

Skeletons or medial axis transforms [BB82] are heavily used in image processing and computer vision. The concept of a skeleton is a useful representation of many topological and size characteristics of the polygon. As its name implies, the skeleton always lies within the boundary of the polygon. All these properties make it ideal for serving as a replacement of the normal definition of polygonal centroid.

The skeleton or medial axis transform of a two-dimensional region is defined as the locus of the centre of an inscribed circle of maximal diameter as it moves inside the object as shown in figure 7.6. Another definition of medial axis transform is that it consists of all interior points with more than one closest point on the polygon boundary.

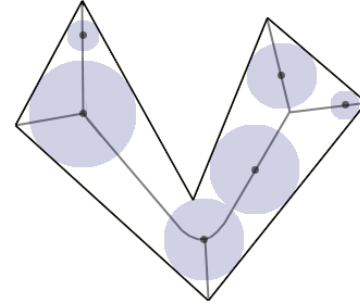


Figure 7.6. Medial axis transform

We call segments of the skeleton that are equidistant from two adjacent edges *limbs* of the skeleton while the rest of it we will call the *backbone*.

Note that the medial axis transform is made up of straight-line segments and parabolic arcs. The parabolic arcs are formed at points that are equidistant from a concave vertex and ‘opposing’ edges. These parabolic arcs are considered as disadvantage in our application both in terms of representation and construction.

### 7.4.2 Introduction to Straight Skeleton

Recently, Aichholzer and Aurenhammer [AA96] introduced a new type of skeleton, the straight skeleton, which has the nice property that it consists of only straight-line segments with no parabolic arc. An example of a straight skeleton is shown in figure 7.7.

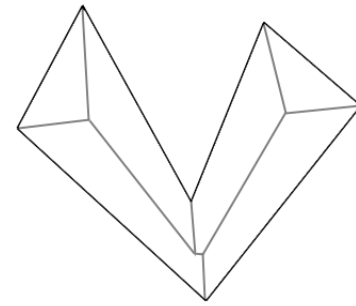


Figure 7.7. Straight skeleton

Unfortunately, while the medial axis transform is defined geometrically, Aichholzer and Aurenhammer, although attempted, could not come up with a nice geometric definition of the straight skeleton.

The straight skeleton is defined<sup>4</sup> procedurally as follows:

For an  $n$ -gon  $G$ ,  $n$  wavefronts arises initially from the  $n$  edges for  $G$ . Each wavefront propagates inwards simultaneously, at the same speed, and in a self-parallel manner. The

<sup>4</sup> The definition given here is slightly simplified from that given in the original paper which works on an arbitrary straight line graph rather than just a closed polygon.

endpoints of each wavefront move along the angular bisectors with the two adjacent wavefronts.

Each point on the wavefront continues to propagate until it hits some other wavefront. The straight skeleton of  $G$  is then defined as all points where wavefronts hit.

While the straight skeleton represents many of the same topological and size characteristics of the polygon as the medial axis transform does, it is different from the medial axis transform in two major aspects:

1. Straight skeleton contains only straight-line segments while medial axis transform will have parabolic arcs if concave vertices exist. This makes the straight skeleton computationally easier to compute. In fact, medial axis transforms are mostly employed in image-based application since the vector-based construction of medial axis is extremely hard and computational expensive, while the pixel-based approximation is simple to compute.
2. More importantly, every single vertex of a polygon will have a corresponding limb in the straight skeleton while only convex vertices will have a limb in the medial axis transform. Hence for an  $n$ -sided polygon, the straight skeleton will have exactly  $n$  limbs.

Both of these make the straight skeleton particularly suitable for our application.

#### 7.4.3 Functional Definition of Targets Using the Straight Skeleton

Recall from earlier in the chapter that a complete definition of a target function in three dimensions can roughly be divided into three parts namely: the initial targets, the F-targets and the E or V-targets.

Since all F-targets are by definition given as:  $t_j^{i+1} = t_j^i$  for  $0 < j \leq N_i$ , only the initial targets and the E or V-targets need to be specified for the skeletal target function. Again,  $N_i$  is the number of faces of the  $i^{\text{th}}$  iteration mesh.

Let  $\phi(f)$  be a function which maps an  $n$ -sided polygon face  $f = \{p_0, p_1, \dots, p_{n-1}\}$  to an  $n$ -tuple  $(j_0, j_1, \dots, j_{n-1})$ , while  $j_k$  is the joint point between the limb of  $p_k$  and the backbone of the straight skeleton of  $f$ .

Figure 7.8 shows the joint points of a straight skeleton. The correspondence of each joint point with the vertex is stressed by the indexing in the figure

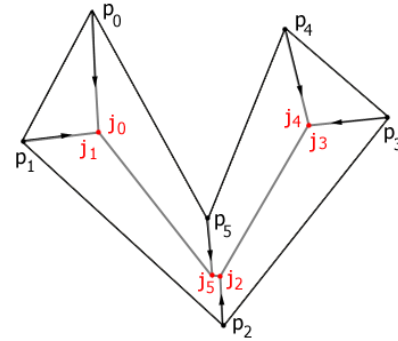


Figure 7.8. Joint points of straight skeleton

E or V-targets for any non-zero iteration mesh face is simply given as:



$$t_j^i = \varphi(f_j^i) \quad \text{for } i > 0 \text{ and } N_i < j < N_{i+1}$$

The zero iteration targets implicitly define the preserved planar regions of the resultant shape and hence the roundedness of the resultant shape, and so it is sensible to define them either with automatic or interactive methods. We will focus here only on the automatic definition with a single scalar control of roundness.

We can define the zero iteration targets with the roundness parameter,  $\alpha$ , as:

$$t_j^0 = (1 - \alpha) V(f_j^0) + \alpha \varphi(f_j^0)$$

where  $V(f)$  is the vertex function for an input face  $f$ , i.e. a function which maps an  $n$ -sided polygonal face,  $f$ , to an  $n$ -vector  $(p_0, p_1, \dots, p_{n-1})$  where  $p_i$  is the  $i^{\text{th}}$  vertex of  $f$ .

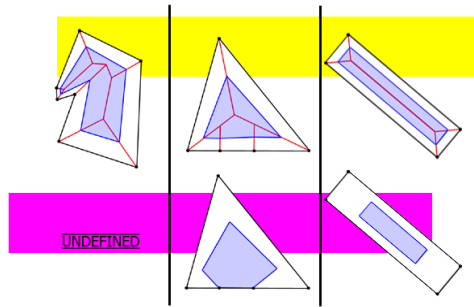


Figure 7.9. Preserved planar regions defined using the straight skeleton (upper row) and mesh pre-partitioning (lower row)

Zero iteration targets defined using the straight skeleton offer many advantages over those defined using the partitioning scheme in the mesh pre-partitioning approach. Not only does the former support concave input faces, but also it produces more “reasonable” result in situation like input face having collinear boundary segments or large relative length between different edges.

Figure 7.9 shows in the upper row the preserved regions defined using a straight skeleton on three different shapes. The lower row shows the corresponding ones defined using the mesh pre-partitioning approach. The first pair in the left illustrates the behaviour on concave faces. The pair in the middle shows a triangular face with two extra vertices on its base edge. It can be seen that the preserved region defined using a straight skeleton is more robust to the collinear segments. The last pair shows that rounding is applied more evenly with a preserved region defined using a straight skeleton on a face with a large aspect ratio.

The solution, however, is not perfect. Problems can arise in the definition of the initial skeletal target with some very concave faces when the required rounding is large. Figure 7.10 shows a case where the defined region to be preserved is not a valid polygon. The suggested approach to dealing with the situation is *partition on detection*. Since it is easy to detect the validity of the preserve regions once created, we can triangulate the polygonal face if needed. This approach minimises the triangulation required. Triangulation, to some extent, destroys the essential polygonal nature of the mesh, which we would like to preserve during the rounding process.

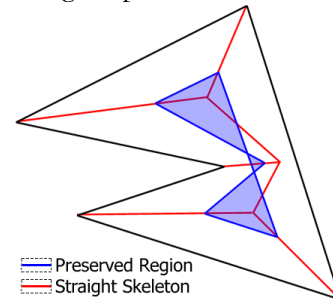


Figure 7.10. Problem of the initial skeletal target

Figure 7.11 shows examples of target-driven polyhedron bevelling using the skeletal targets on both convex and concave input faces. The straight skeletons are shown in blue and the targets are in red.

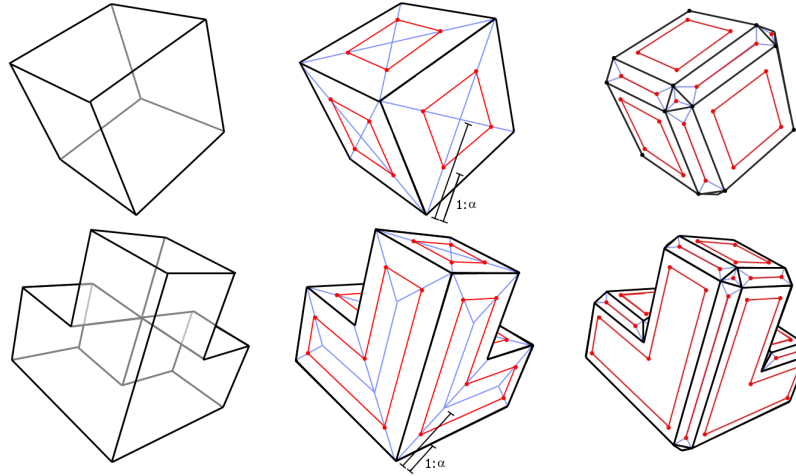


Figure 7.11. The initial and the first iteration mesh of target-driven polyhedron bevelling using skeletal targets

## 7.5 Results and Discussion

The major contribution of target-driven polyhedron bevelling is the stability offered by eliminating the partitioning process. It does not suffer from the limitation of fundamental limit of the roundness  $\alpha_{\max}$  as in the mesh pre-partitioning polyhedron bevelling. The method is stable for all input meshes with convex faces. It is also applicable to concave input faces in most circumstances.

Figure 7.11 shows some example objects rounded by the method. They are normally unachievable by the mesh pre-partitioning method without having their concave faces cut into convex polygons in advance.



Figure 7.11. Examples of target-driven polyhedron bevelling using skeletal targets

The other area of concerns is the continuity of the resultant surface. Although one can see from figure 7.11 that the results are visually pleasing, their continuity is unknown to the author. The resultant surface is not the same as the standard Catmull-Clark/Doo-Sabin surface both because of the replacement of centroids with the skeletal targets and the idea of stepping linearity half way towards the destination per iteration.

Figure 7.12 depicts the fragmentation of a target-driven polyhedron bevelled object. The result is similar to that of the corners extraction and reconstruction method described in the previous chapter. The fragmentation occurs only where necessary, over the regions of rounded corners and edges, and large area of planar region are left untouched.

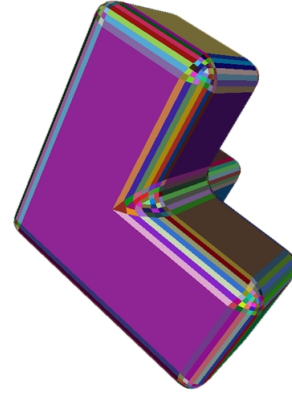


Figure 7.12. Fragmentation of a target-driven polyhedron bevelled object

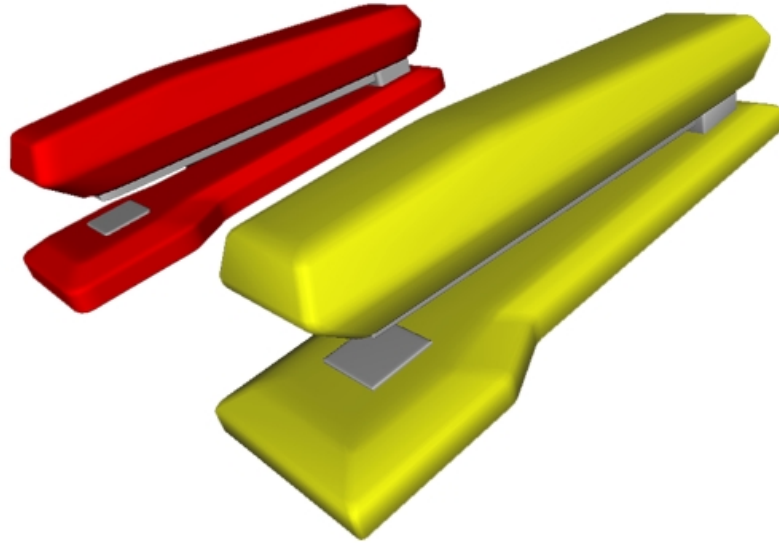


Figure 7.13. Staplers rounded with the two different polyhedron beveling methods.

Owing to the algorithmic simplicity, the method also executes very quickly. Its execution speed is even faster than the mesh pre-partitioning approach and is approaching the speed of the Catmull-Clark/Doo-Sabin subdivision methods. The only extra computation, apart from the mesh subdivision, is to calculate the skeletal targets. The computation of the straight skeleton can be rather expensive; however, the target of each face needs to be computed only once and stays unchanged throughout the recursive process. Also, all E and V-faces generated after the second iteration are only 4-sided and hence their straight skeleton can be computed very fast. Its execution speed is sufficient for real-time usage. For instance, our Java implementation achieves an execute-time approximately of 0.02s for the yellow stapler shown in figure 7.13.



## 7.6 Comparison with the Mesh Pre-partitioning Approach

Most of the differences between target-driven polyhedron bevelling and mesh pre-partitioning polyhedron bevelling have already been discussed. Key points are tabulated below.

Features	Polyhedron Bevelling by Mesh Pre-partitioning	Polyhedron Bevelling by Target-driven Subdivision
<b>Algorithmic Stability</b>	Fair. The method does not support concave faces. Even with convex input faces, there is a fundamental limit on the roundness parameter, $\alpha_{\max}$ , which cannot be achieved. The optimisation method ‘corners subdivision and reconstruction’ resolves this limitation by pushing the fundamental limit close to the optimal value of 1; however, the method still suffer from the low fundamental limit on the roundedness, $\rho_{\max}$ on an input face with large number of vertices	The stability is very much enhanced. It suffers from neither the $\alpha_{\max}$ nor the $\rho_{\max}$ problems, as the pre-partitioning phase is completely eliminated. Any arbitrary convex faces are supported. Concave polygonal faces are as well supported, but the definition of initial target using straight skeleton may fail on such input if the desired roundness is large. Fortunately the problem is easily detectable once the target is constructed. Such concave face can be then cut into convex polygons.
<b>Continuity</b>	Analytical smooth, $GC^1$ , but occasionally produces unpleasant results in situations like several consecutive collinear edges in a polygonal face.	Visually smooth. Analytical smoothness is unknown. One may notice the result, in some cases, is not as smooth as that of the mesh pre-partitioning approach by very careful examination; however, it is generally more robust to situation like consecutive collinear edges and so forth.
<b>Execution Speed</b>	Fast in comparison with most existing volume rounding methods. Models with typical complexity can be rounded within a second.	Even faster than the mesh pre-partitioning method. Our prototype indicated an approximately the same execute speed as the optimised mesh pre-partitioning method ‘corners subdivision and reconstruction’. We, however, believe that the method is potentially a lot faster due to its simplicity.
<b>Fragmentation</b>	The resultant surface exhibits much higher fragmentation than necessary. The problem is however resolved by the more complicated optimisation methods.	The fragmentation is similar to that resulted from the corners-subdivision and reconstruction method.



## 7.7 Chapter Summary and Conclusion

This chapter presented the polyhedron beveling by target-driven subdivision. The idea of the method is to modify the actual mesh subdivision phase to achieve the task of preserving planar region in a more direct and simple manner.

The target-driven polyhedron beveling method arises directly from the idea of two-dimensional Chaikin refinement method where, at each step of refinement, each vertex is moving halfway toward the edge centroid. The generic framework of target-driven polyhedron beveling replaces the concept of centroid by the so-called *targets* or a set of destination points towards which the vertices are moving gradually at each step of recursion.

The choice of target function affects both the use of the rounding parameter and the continuity of the resultant shape. The use of a target that makes use of the straight skeleton was investigated and proved very effective except for polyhedra with extremely concave faces.

The target-driven approach differentiates itself from the mesh pre-partitioning approach largely by the stability advantages offered. It is robust to all arbitrary convex input polygonal faces as well as to most concave ones.

Although the result is visually pleasing, the continuity is unknown to the author due to the lack of geometrical definition of straight skeleton. This opens up future research directions in studying the exact mathematical properties and, if necessary, in developing a new target function that leads to an analytically smooth resultant surface.



## Chapter 8 The Just-In-Time Implementation

In this chapter, we shift our focus to the practical implementation of the VRML just-in-time volume rounder. Since the preceding chapters have already covered the rounding algorithm – polyhedron bevelling – the focus here will be primarily on the architecture and various other details of the implementation.

The discussion in this chapter assumes familiarity with the VRML97 language definition [VRML97] and the notation used in that definition. The chapter is, however, designed so that readers unfamiliar with the definition can understand without too much concerning about the exact VRML syntax.

### 8.1 Scope and Objective

The aim of the implementation is to extend VRML to allow scene object with user specified roundness. We will limit our scope to deal with only polyhedra (or, in VRML terminology, *IndexedFaceSet*); other types of VRML built-in scene object like “cylinder” or “box” will be not considered here. The first version of the prototype system will also assume closed input mesh.

As we have briefly mentioned right at the beginning in Chapter one, the target system is a just-in-time implementation: the VRML file describes the original “unrounded” volume and the rounding operation is performed just before rendering. The benefits of the approach have already been discussed in section 1.4. The JIT modification is best described by its conceptual model shown in figure 8.1. The VRML file may be located either in a remote host or locally on a client machine. It is transferred over a communication link upon request. The communication link can be the Internet or a local area network using the hypertext transfer protocol (HTTP) or merely a local data bus if it is a local file. The JIT plug-in (or browser extension) is launched and the rounding algorithm is executed. The rounded mesh is, finally, rendering by the VRML browser.

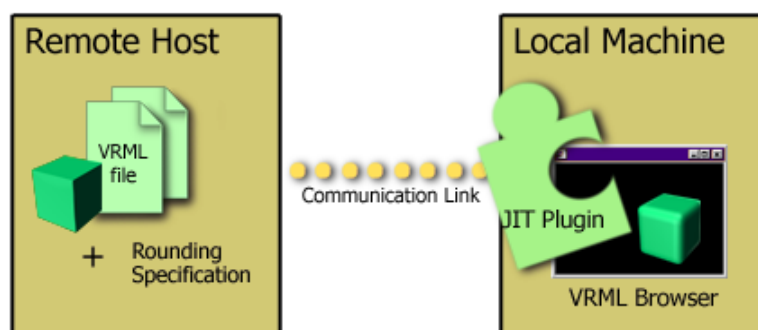


Figure 8.1. Conceptual model of the JIT modification

The rounding specification of the prototype system is limited to a global rounding operation with a single scalar control. The system should, however, be easily extendable to allow more complex roundness control.

Three other requirements / design goals are as listed below:

1. The effort required to convert existing VRML models so as to benefit from the rounding extension should be minimised.
2. The integration between the JIT plug-in and the VRML browser should be as seamlessly as possible.
3. The client-side installation should be avoided / minimised.

## 8.2 Extensibility of VRML

Before we move on to our implementation details, we will first briefly describe how VRML offers extensibility to the base language. Being one of the six design criteria stated in the international standard [VRML97], extensibility is mainly provided via two mechanisms in VRML. They are prototyping and scripting, both of which will be used in our implementation.

### 8.2.1 Prototypes

Prototyping is a mechanism for defining new node types in terms of already-defined node types. The VRML standard defines a set of built-in node type. Node types are defined in the standard as:

*“A characteristic of each node that describes, in general, its particular semantics. For example, Box, Group, Sound, and SpotLight are node types.”*  
[VRML97 3.67]

A node is the fundamental component of the scene described in a VRML file. The relationships between nodes, node types, built-in node types and prototypes are best illustrated in a *metamodel*<sup>5</sup> shown in figure 8.2. The diagram reads as follow: each node has a node type, which can be either a built-in type or a prototype. A prototype is in turn defined using a set of nodes of already-defined node types.

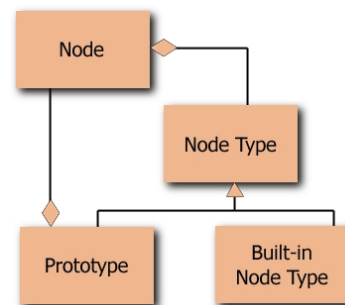


Figure 8.2. A portion of the metamodel of VRML using the notation of the Unified Modelling Language [FS97]

<sup>5</sup> The term “metamodel” means a model that describe a model.



The prototyping mechanism allows extension to the base language. One can, for instance, define a geometry node type of a “Cube” using the “Box” built-in node type with syntax:

```
PROTO Cube [] { Box [] }
```

The ‘Cube’ node can then be instantiated as if standard node type:

```
Shape { geometry Cube { } }
```

More complex node types can be defined by both aggregation and parameterisation. For example, more than one node type can be used to define a single new type, and also parameters or “*fields*” can be added to differentiate the behaviours of each instantiation of the same node type.

### 8.2.2 Scripts

Scripting is yet another mechanism allowing extensibility in VRML. With scripting, a VRML world can change dynamically in response to user inputs, external events and the current state of the world. There are two scripting languages proposed by the standard<sup>6</sup>. They are Java and ECMAScript. The extension that allows one of an object-oriented, general-purpose programming language like Java provides virtually unlimited ability for dynamically manipulating the virtual world. Our discussion will focus only on scripting with Java.

Java source code is first compiled into “bytecode” or “class file”, which can be executed within the virtual machine of the browser. The binding between the bytecode and VRML file is established via a “Script” node with syntax:

```
Script { url "...../foo.class" }
```

where the *url* field of the Script node contains the uniform resource locator (URL) of a file containing the Java bytecode.

Scripting in VRML is event-driven. Events to the Script node will trigger the corresponding methods in the script. On the other hand, the script manipulates the world again by sending events. The output events generated from the script are routed to the appropriate node in the scene graph in order to vary its behaviour.

## 8.3 The Language Extension

The implementation of the JIT VRML volume-rounding extension is accomplished by definition of a new geometry node type using the prototyping mechanism in VRML. We call this new node type “*BevelledPolyhedron*”. VRML authors should be able to use the new node type as if it was a the standard geometry node type:

---

<sup>6</sup> Note, however, that no particular scripting language is compulsory to the standard. The only available VRML browser that currently supports Java implementation is Cosmoplayer 2.x by Silicon Graphics.

```
Shape { geometry BevelledPolyhedron { ... } }
```

where the ellipsis stands for various input fields, which will be discussed in this section.

As mentioned before, prototyping in VRML allows parameters or fields to be included so as to differentiate the behaviour of each instantiation. Each field has a type, a name and an optional default value.

For our application, two parameters are required. They are the initial mesh configuration and the rounding specification. Since we are interested in a global rounding operation with a single scalar control, the rounding specification is therefore simply a floating-point value from zero to one. We use the roundness control,  $\alpha$ , in the algorithm directly for the sake of simplicity and name it the *roundnessParameter*. A default value of 0.5 is given. The declaration is therefore:

```
field SFFloat roundnessParameter 0.5 #[0, 1]
```

As for the initial mesh configuration, there are numerous different representations. However due to the fact that one of our design goals is to minimise the effort required to convert existing VRML models in order to benefit from the rounding extension, the exact same representation employed by the VRML IndexedFaceSet node type is chosen. The corresponding declarations are:

```
field SFNode coord NULL  
field MFInt32 coordIndex []
```

The *coord* field contains a Coordinate node that defines the three-dimensional vertices referenced by the *coordIndex* field. BevelledPolyhedron uses the indices in its *coordIndex* field to specify the polygonal faces of the initial mesh by indexing into the coordinates in the Coordinate node. An index of "-1" indicates that the current face has ended and the next one begins. The last face is optionally followed by a "-1" index.

Each face of the node will have:

1. at least three non-coincident vertices,
2. vertices that define a planar polygon,
3. vertices that define a non-self-intersecting polygon.

We assume also that the input mesh is closed. Therefore:

1. Each edge defined by two connected vertices must be shared by two polygonal faces

Figure 8.3 shows the syntax of a rounded scene object and its corresponding non-rounded IndexedFaceSet. The minor differences are shown in bold. Also because of the default value in the roundness parameter, that field is optional. Hence the only mandatory conversion is to simply change the name of the node type.



<pre> Shape {   geometry BevelledPolyhedron {     coord Coordinate {       point [ 0 0 0, 1 0 0,               1 1 0, 0 1 0,               0 0 1, 1 0 1,               1 1 1, 0 1 1 ] }     coordIndex [ 0, 3, 2, 1, -1,                  6, 7, 4, 5, -1,                  7, 3, 0, 4, -1,                  2, 6, 5, 1, -1,                  4, 0, 1, 5, -1,                  2, 3, 7, 6, -1 ]      roundingParameter 0.5   }   appearance Appearance { } } </pre>	<pre> Shape {   geometry IndexedFaceSet {     coord Coordinate {       point [ 0 0 0, 1 0 0,               1 1 0, 0 1 0,               0 0 1, 1 0 1,               1 1 1, 0 1 1 ] }     coordIndex [ 0, 3, 2, 1, -1,                  6, 7, 4, 5, -1,                  7, 3, 0, 4, -1,                  2, 6, 5, 1, -1,                  4, 0, 1, 5, -1,                  2, 3, 7, 6, -1 ]    }   appearance Appearance { } } </pre>
--	--

Figure 8.3. VRML code for a rounded scene object and its corresponding non-rounded version

Finally, an optional field *convex* can be included. This is a hint that may be used for future optimisation purposes. This field specifies whether the input mesh contains only convex polygonal faces. It defaults to *true*.

```
field SFBool convex TRUE
```

## 8.4 Implementation and Architecture of the Just-In-Time Modification

The essence of JIT modification is that the actual rounding operation is performed in real-time using the scripting mechanism. The rounding algorithm is implemented in the Java scripting language compiled into bytecode.

The architecture of the system is shown in figure 8.4. The intention here is to introduce a JIT VRML plug-in into the VRML browser, so that the VRML files with the extended syntax can be processed just in time by the plug-in before they are rendered in the browser. There are two components in our JIT VRML plug-in. The first is the prototype definition of the BevelledPolyhedron node type, which defines the parameterisation, sets up the proper event routing and indicates the location of the Java bytecode. The second is the actual Java bytecode, which implements the polyhedron bevelling method.

The two components integrate seamlessly into a VRML browser. The left-hand side of the figure 8.4 shows an internal structure of a typical VRML browser. The VRML file containing the extended syntax is received by the browser. The file is first interpreted by the parser. The

parser “understands” the extended syntax by reading in the prototype definition from the plug-in. The scene graph is then built and the execution engine fires the initialising event, which triggers the Java bytecode of polyhedron beveling to run. The rounding method is performed and the scene graph is modified accordingly. The rounded volume is finally sent to the presentation module for rendering.

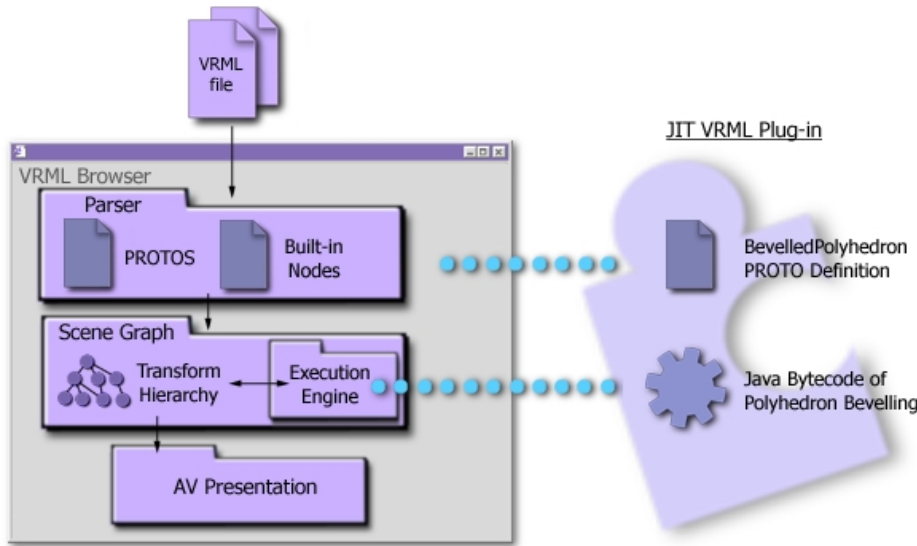


Figure 8.4. The integration of JIT VRML plug-in and VRML browser

The architecture shown in figure 8.4 requires that the JIT VRML plug-in be installed on the local machine prior to the interpretation of the VRML file. The requirement of the client side pre-installation of the plug-in may detract from its commercial value, since casual users may not be bothered to install or serious users may refuse to do so for security reasons.

Figure 8.5 shows a more general structure in which the plug-in may be sitting on a network remote to the local machine.

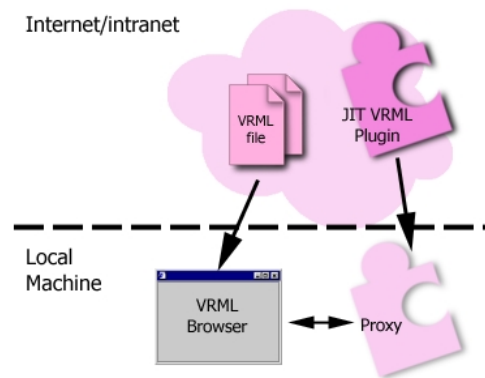


Figure 8.5. Network transparency of the JIT modification

The VRML browser can access the plug-in as if they are on the same machine. This, however, requires the VRML file to include the declaration of the external prototype definition:

```
EXTERNPROTO BevelledPolyhedron [...] "http://.../bevelledpolyhedron.wrl"
```

where the parameter list is included in between the square brackets and the URL of the VRML file containing the prototype definition is indicated at the end of the declaration. The client-side installation is, thereby, completely eliminated.





The complete declaration of the BevelledPolyhedron prototype is included in figure 8.6. Various other implementation details such as event routing and field binding will not be discussed here. Readers are referred to the standard [VRML97] for those concepts.

```

PROTO BevelledPolyhedron [
  field SFNode coord NULL
  field MFInt32 coordIndex []
  field SFFloat roundingParameter 0.5 #[0,1]
  field SFBool convex TRUE
] {
  DEF shape IndexedFaceSet {
    coord DEF coordinate Coordinate { }
    creaseAngle 3.14
    convex IS convex
  }
  DEF script Script {
    field SFNode coord IS coord
    field MFInt32 coordIndex IS coordIndex
    field SFFloat roundingParameter IS roundingParameter
    eventOut MFVec3f vertices_changed
    eventOut MFInt32 coordIndex_changed
    url "PolyhedronBevelling.class"
  }
  ROUTE script.vertices_changed TO coordinate.set_point
  ROUTE script.coordIndex_changed TO shape.set_coordIndex
}

```

Figure 8.6. The prototype definition of the BevelledPolyhedron node type

## 8.5 Result

Figure 8.7 shows a stapler rounded by target-driven polyhedron bevelling using the JIT approach in a VRML browser. The file containing the initial mesh with the rounding specification is only four kilobytes in size. In contrast, the fully expanded file is of 215 kilobytes.

Assuming a typical home-use network connection with 56 kbps (kilobits per second) at full capacity, the full file takes half a minute or more to transfer. While it takes only approximately half a second to download a four kilobytes file plus one tenth of a second to perform the rounding operation with the JIT approach.

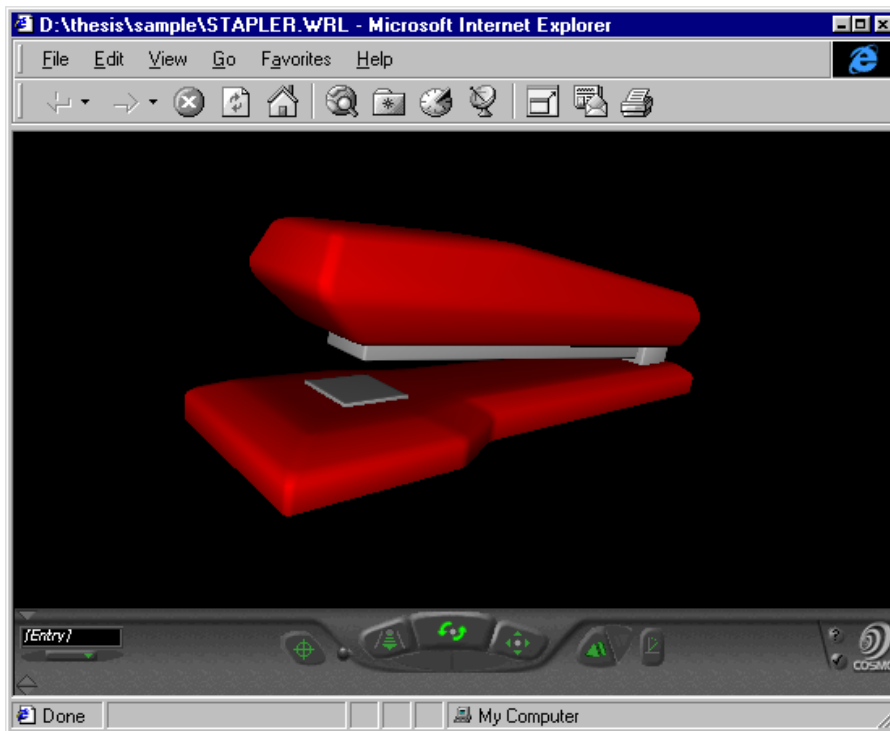


Figure 8.7. A rounded stapler in VRML browser

The reduction of the downloading time together with the enhanced readability and reusability of the file format has proven the tremendous potential of the JIT approach.

## 8.6 Chapter Summary and Conclusion

The chapter has discussed the implementation and architectural details of just-in-time volume rounding using VRML as the base language. We have proposed a language extension to VRML that allows rounded object to be specified in terms of the initial control mesh and a rounding specification. The implementation integrates seamlessly into the VRML environment and shows encouraging results.



---

## Chapter 9 Conclusions

We have now presented our new volume rounding method – polyhedron bevelling – and its practical implementation in the just-in-time VRML volume rounding extension. The results are very promising. Chapter nine concludes with a summary of the achievements of the thesis and suggestions for future work.

### 9.1 Research Overview

Our research was initiated from the goal of developing a fast volume rounding algorithm and, ultimately, a practical VRML volume rounding tool. Throughout the year, we have gone through several different stages of development.

Beginning with the problem analysis and requirement gathering, we explored the conceptual view of the problem. The meaning of “rounding” of a three-dimensional object was investigated. The subject is neither scientifically defined nor of an objective nature. A survey on how humans perceive roundness was therefore conducted and, as a result, a set of quality criteria of a rounding operation was developed. Most importantly, we have identified the surface area ratio between the planar regions and the rounded regions as the most influential factor that affects an object’s perceived roundness. This finding led to a significant effort being put into preserving planar regions with our rounding algorithm.

Two polyhedron-bevelling methods have been developed. Both are based on recursive mesh subdivision. The mesh pre-partitioning approach leaves the recursive mesh subdivision phase unaltered while adding a pre-partitioning phase to indirectly control the resultant roundness. Mesh pre-partitioning polyhedron bevelling inherits the analytical smoothness from a recursive subdivision surface; however, the pre-partitioning phase suffers from several construction limitations and stability problems if the desired roundness is large. The target-driven approach, on the other hand, modifies the actual recursive mesh subdivision to accomplish more directly the task of preserving planar regions. The method is more stable. It works on any arbitrary convex polygonal faces and most concave ones. The two methods produce excellent results on a wide range on input meshes both in terms of execution speed and resultant quality.

The final stage involved implementing a JIT VRML plug in, which allows rounded VRML scene objects to be specified by the initial mesh and the rounding specification. Mostly, VRML files are located on a remote machine and get transferred upon request. Hence, because of the huge difference in size between a VRML file containing rounded scene objects and one with the corresponding initial meshes, the JIT approach saves enormous cost in network traffic and downloading time. The rounding algorithm can be executed within a fraction of a second just before the scene is rendered. The development of the JIT implementation has proven the practicality of the polyhedron bevelling methods.

## 9.2 Limitations and Directions of Future Research

The development of polyhedron bevelling is still in its infancy. Although the resultant surface of the target-driven polyhedron bevelling is visually smooth, its continuity is yet to be investigated. We wish to understand the exact mathematical behaviour of the target-driven polyhedron bevelled surface with skeletal targets. Better still, alternative functional definitions of targets that guarantee analytical smoothness are sought.

The practical implementation of polyhedron bevelling works beautifully in VRML. The major limitation of the current implementation is the lack of texture mapping [FV96] supports. With texture mapping, one can position a texture on a shape hence adding fine details to the scene without imposing the cost of explicit modelling of the geometric details. Because of its benefit, texture mapping is widely employed in most computer graphics scene modelling, and thus the lack of texture mapping support severely constraints the practical use of polyhedron bevelling. We realise the real solution here is the three-dimensional texture mapping; however, the support for three-dimensional textures is expensive and therefore uncommon. For example, VRML currently support only two-dimensional texture mapping. Since our ultimate goal is to round existing VRML scene object with polyhedron bevelling and most of the existing scene are two-dimensional textured, we believe that the two-dimensional texture mapping support for polyhedron bevelled object will be, by itself, a challenging and viable topic.

A final research direction we wish to suggest is related to the recent interest in view dependent modification and levels of detail. While it is adequate to perform polyhedron bevelling to a typical scene with moderate complexity in real time, the situation will not be the same for highly complicated scene objects. The idea of view dependent modification and levels of detail is to give the best accuracy to the parts of the scene that are clearly viewable in the current view while ignoring the exact computation of the other parts. The scene is adjusted accordingly in real time when the viewpoint is changed. The idea is applicable to polyhedron bevelled objects, since the quality of the resultant shape is dependent upon the number of refinement steps taken. The initial mesh gives a rough approximation to the final shape and a better approximation is obtained from each extra step of refinement with an increasing computation cost. The number of refinement steps taken is hard-coded in the current JIT plug-in. View dependent and levels of detail support which adjust the required quality dynamically is desired in the future.



## Appendix A Mathematical Proofs

### A.1 Proof of Linear Boundary Property

This section provides the proof of the linear boundary property of preserved regions of a mesh pre-partitioning polyhedron bevelled volume for arbitrary 3 or 4-sided input faces.

The proof will be given in two parts. We know that the resultant limit surface will interpolate the centroid of the boundary face in between the two corner faces. The first part of the proof shows the linearity of the three consecutive centroids of an original edge. See figure A.1.

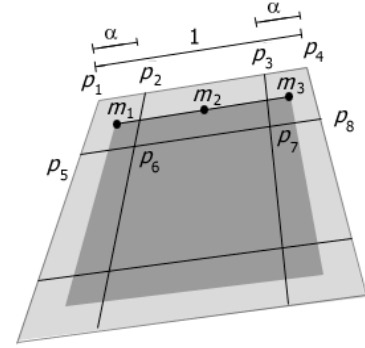


Figure A.1. Linearity of centroids of the three consecutive boundary faces

The second part of the proof shows the boundaries of the preserved region are straight lines joining consecutive centroids. Because of the symmetry of the configuration, we need to show only that one of the segments is actually straight.

By construction, all *boundary* faces after pre-partitioning are 4-sided. Hence we can write the centroids  $m_1$ ,  $m_2$  and  $m_3$  in figure A.1 as:

$$m_1 = \frac{1}{4} (p_1 + p_2 + p_5 + p_6), \quad m_2 = \frac{1}{4} (p_2 + p_3 + p_6 + p_7), \quad m_3 = \frac{1}{4} (p_3 + p_4 + p_7 + p_8)$$

Also due to the linearity of  $p_1, p_2, p_3$  and  $p_4$ , we can write  $p_2$  and  $p_3$  in terms of  $p_1, p_4$  and the roundness control,  $\alpha$ , as

$$p_2 = \alpha p_4 + (1 - \alpha) p_1 \quad \text{and} \quad p_3 = \alpha p_1 + (1 - \alpha) p_4$$

Similarity,  $p_6$  and  $p_7$  can be given as:

$$p_6 = \beta p_8 + (1 - \beta) p_5 \quad \text{and} \quad p_7 = \chi p_5 + (1 - \chi) p_8$$

where  $\beta$  and  $\chi$  are the *internal edge length ratios* in which,

$$\beta = \frac{|p_5 p_6|}{|p_5 p_8|} \quad \text{and} \quad \chi = \frac{|p_7 p_8|}{|p_5 p_8|}$$

It is easy to verify that  $m_2$  is the mid-point of  $m_1$  and  $m_3$  i.e.  $m_1, m_2$  and  $m_3$  are collinear, if  $\beta = \chi$ . Hence in order to show the linearity of  $m_1, m_2$  and  $m_3$ , we need to show the length of  $p_5p_6$  is the same as that of  $p_7p_8$ .

First, for the sake of simplicity, we will consider an arbitrary triangular face as shown in figure A.2. For the constraint  $\beta = \chi$  to be satisfied, we need to show that the length of  $p_5p_6$  is the same as that of  $p_7p_8$ .

Since,

$$\frac{|p_5p_{12}|}{|p_1p_{12}|} = \frac{|p_8p_{12}|}{|p_4p_{12}|} = \frac{1-\alpha}{1}$$

$\therefore p_5p_{12}p_8$  and  $p_1p_{12}p_4$  are similar triangles.

$\therefore p_1p_4 \parallel p_5p_8$

Similarly, it can be shown that  $p_2p_{11} \parallel p_1p_{12}$ . Therefore all the corner faces are parallelograms.

Thus  $|p_5p_6| = |p_1p_2| = |p_7p_8| = |p_3p_4| = \alpha$ .

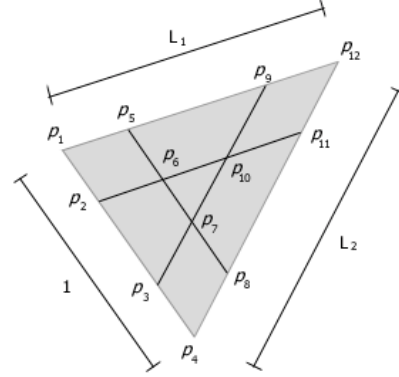


Figure A.2. Partitioning of a triangular face

We have thereby completed the first part of the proof of the linear boundary property for any arbitrary triangular face.

For a quadrilateral, we define four edge vectors  $v_0$  to  $v_4$  as shown in figure A.3.

$$v_0 = p_4 - p_1, \quad v_1 = p_{16} - p_4,$$

$$v_2 = p_{13} - p_{16}, \quad v_3 = p_{13} - p_1,$$

We can also express the fourth vector in terms of the other three as:

$$v_3 = v_0 + v_1 + v_2$$

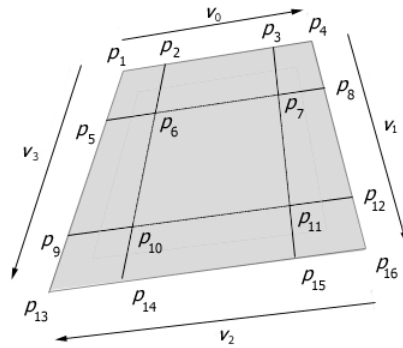


Figure A.3. Partitioning of a quadrilateral face

By assuming  $p_1$  is located at the origin for simplicity,  $p_2, p_8, p_{14}$  and  $p_5$  can be written as:

$$p_2 = \alpha v_0$$

$$p_8 = v_0 + \alpha v_1$$

$$p_{14} = v_0 + v_1 + (1 - \alpha) v_2$$



$$p_5 = \alpha v_3 = \alpha(v_0 + v_1 + v_2)$$

Now the intersection point,  $p_6$ , of  $p_2p_{14}$  and  $p_5p_8$  can be expressed as either:

$$p_6 = p_2 + \delta(p_{14} - p_2)$$

or

$$p_6 = p_5 + \epsilon(p_8 - p_5)$$

where  $\delta = \frac{|p_2p_6|}{|p_2p_{14}|}$  and  $\epsilon = \frac{|p_5p_6|}{|p_5p_8|}$

By expanding both the equations of  $p_6$  and comparing the coefficients of  $v_0$ ,  $v_1$  and  $v_2$ , one can verify that  $\alpha = \delta = \epsilon$ . Also, because of the symmetry of the configuration, it is sufficient to show that  $|p_5p_6| = |p_7p_8|$ ,  $|p_9p_{10}| = |p_{11}p_{12}|$ ,  $|p_2p_6| = |p_{10}p_{14}|$  and  $|p_3p_7| = |p_{11}p_{15}|$ . Hence the constraint  $\beta = \chi$  is satisfied for an arbitrary quadrilateral.

Note that not only are all the internal edge length ratios the same for any 3 or 4-sided polygonal face, but also all of them are equal to the roundness control  $\alpha$ . This finding is useful in later part of the proof.

We will now move onto the second part of the proof, in which we need to show that the boundaries of the preserved region are actually straight lines joining consecutive boundary centroids.

As previously noted, because of the symmetry of the partitioning, we need to show only one of the segments is actually straight. We can show that the boundary segment between  $m_1$  and  $m_2$  in figure A.4 is linear.

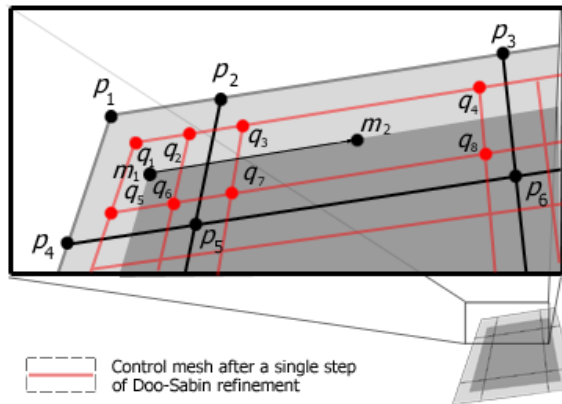


Figure A.4. Linearity of the boundary segment between two consecutive boundary centroids

In order to prove that, we will show the boundary control points ( $q_1$ ,  $q_2$ ,  $q_3$  and  $q_4$  as well as  $q_5$ ,  $q_6$ ,  $q_7$  and  $q_8$ ) remain collinear after one step of mesh subdivision. The proof can hence be

applied recursively to show that all points on the limit boundary between  $m_1$  and  $m_2$  are collinear. That is, the segment is a straight line.

Because of the partitioning scheme,  $p_1, p_2$  and  $p_3$  are collinear and so are  $p_4, p_5$  and  $p_6$ . We want to show that  $q_1, q_2, q_3$  and  $q_4$  as well as  $q_5, q_6, q_7$  and  $q_8$  in the next step of refinement are also collinear. Again due to the symmetry of the setting, we need only to show  $q_1, q_2$  and  $q_3$  are collinear.

First because of the linearity of segment  $p_1p_2p_3$  and segment  $p_4p_5p_6$ , we can rewrite  $p_3$  and  $p_6$  as:

$$p_3 = p_2 + \phi(p_2 - p_1)$$

$$p_6 = p_5 + \gamma(p_5 - p_4)$$

where  $\phi$  and  $\gamma$  are some scalar constant are the length ratios  $p_2p_3 : p_1p_2$  and  $p_5p_6 : p_4p_5$ .

Due to the fact that all the boundary faces after partitioning are 4-sided, the control points after a single step of Doo-Sabin refinement are given as:

$$q_1 = \frac{9}{16}p_1 + \frac{3}{16}p_2 + \frac{3}{16}p_4 + \frac{1}{16}p_5$$

$$q_2 = \frac{9}{16}p_2 + \frac{3}{16}p_1 + \frac{3}{16}p_5 + \frac{1}{16}p_4$$

$$q_3 = \frac{9}{16}p_2 + \frac{3}{16}p_3 + \frac{3}{16}p_5 + \frac{1}{16}p_6$$

Again, the linearity of  $q_1, q_2$  and  $q_3$  is satisfied if and only if we can rewrite  $q_3$  in the form:

$$q_3 = q_2 + \eta(q_2 - q_1)$$

By substitution and comparison, we can again verify the linearity property is satisfied if  $\phi = \gamma$ . That is, the length ratio between  $p_1p_2$  and  $p_2p_3$  is the same as that between  $p_4p_5$  and  $p_5p_6$  in figure A.4. This condition holds for any arbitrary 3 or 4-sided polygonal face after the mesh pre-partitioning, as we have already proven that all the internal corner edge ratios are  $\alpha$ . Hence both  $\phi$  and  $\gamma$  must be equal to  $(1-2\alpha) : \alpha$ .

Since we have shown that  $\eta$  is also a constant, the consecutive edge length ratio for the next refinement step ( $q_1q_2 : q_2q_3$  and  $q_4q_5 : q_5q_6$ ) will again be the same because of the symmetry of the configuration. This completes the proof that the boundary of the preserved region is linear between two consecutive boundary face's centroids.

This together with the first part of the proof show that the resultant preserved region for a 3 or 4-sided polygonal face is actually a 3 or 4-sided polygon respectively. It can be shown numerically that the property does not hold for any arbitrary  $n$ -sided polygonal faces. This, however, does not necessarily disprove the property for all non-“3 or 4-sided” input faces. For example, it can be shown that the linear boundary property holds also for all input of regular  $n$ -gons.





## A.2 Mapping $\rho$ to $\alpha$ for Regular N-gons

This section shows how the mapping from roundedness,  $\rho$ , to the roundness control,  $\alpha$ , on a regular  $n$ -gonal face with mesh pre-partitioning polyhedron bevelling is derived.

Figure A.5 shows the nomenclature used in this derivation.

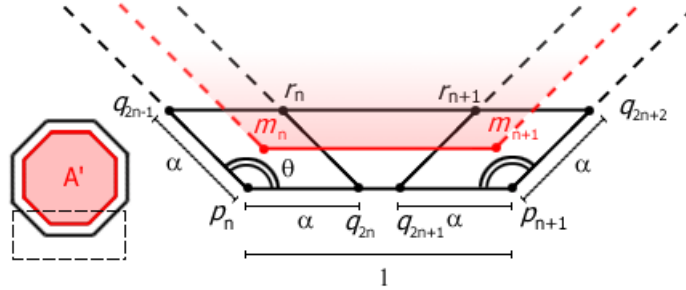


Figure A.5. The nomenclature used in deriving the roundness control from the roundedness

For simplicity, we assume that the length of each edge of the regular  $n$ -gon be 1.

Hence the length of each edge of the corner faces will be  $\alpha$  by construction. Also since the lengths of the edges  $q_{2n-1}p_n$  and  $q_{2n+2}p_{n+1}$  are both  $\alpha$  and the internal angle  $\theta$  of the  $n$ -gon is the same at any vertex, any internal edge  $q_{2n-1}q_{2n+2}$  will be parallel to the corresponding initial edge  $p_n p_{n+1}$ .

Given that:

$$\rho = 1 - \frac{A'_p}{A_p}$$

where  $A'_p$  is the area of the preserved region and  $A_p$  is the area of the original polygonal face.

$$\rho = 1 - \left( \frac{|m_n m_{n+1}|}{|p_n p_{n+1}|} \right)^2 = 1 - |m_n m_{n+1}|^2$$

$$\therefore |m_n m_{n+1}| = \sqrt{1-\rho}$$

Since  $m_i$  is the centroid of the  $i^{\text{th}}$  4-sided corner face,

$$\left| \frac{p_n + q_{2n-1} + q_{2n} + r_n}{4} - \frac{p_{n+1} + q_{2n+1} + q_{2n+2} + r_{n+1}}{4} \right| = \sqrt{1-\rho}$$

with some algebra, it can be shown that:

$$1 - \alpha - \alpha \cos \theta = \sqrt{1-\rho}$$



$$\therefore \quad \alpha = \frac{1 - \sqrt{1 - \rho}}{1 + \cos \theta}$$

where  $\theta = \frac{(n-2)\pi}{n}$  for the internal angle of a regular  $n$ -gon



## Appendix B Implementation of Straight Skeleton

Although an algorithm for constructing straight skeletons can be found in the original paper [AA96], we have followed a slightly different approach. This section presents our divide-and-conquer method for computing straight skeletons.

At each step of the recursion, we compute the minimum collapsing time,  $t_{min}$ , defined as the time taken for the wavefronts to propagate before a collision occurs between two non-adjacent wavefronts. Each wavefront then propagates inwards with time  $t_{min}$ . Zero or more polygons, separated by the collision point, will be formed by the current edges of the wavefronts. The process is then recursively applied to each of the polygons formed.

We illustrate with an example in figure B.1. Wavefronts are initiated from the edges of a polygon and propagate simultaneously until two non-adjacent wavefronts hit at the collision point as shown in figure 7.7(a). Two polygons, a triangle and a quadrilateral, are formed. The algorithm is recursively applied on each of them. Figure 7.7(b) shows further propagation in the quadrilateral, in which only one polygon is formed after the process. Figure 7.7(c) shows that the algorithm terminates when no further polygons are formed by the process. Figure 7.7(d) shows the computed straight skeleton.

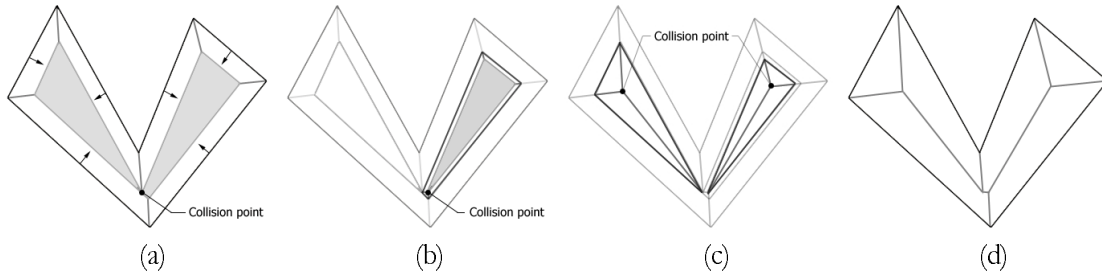


Figure B.1. A divide-and-conquer method for computing straight skeleton

The remaining question is how to determine the minimum collapsing time,  $t_{min}$ , at each step of the recursion. This involves the calculation of the time taken for each angle-bisecting vector,  $v_i$ , hits each of the non-adjacent wavefront of  $v_i$ . We define the angle-bisecting vector at each vertex as the vector pointing inwards with the direction of the angle bisector of the vertex while the length of the vector indicates how fast the adjacent wavefront propagates. See figure B.2. We assume each wavefront propagates inwards at a constant speed of 1. The length or the speed of angle-bisecting vector is hence given by  $\sin \frac{\theta}{2}$ , where  $\theta$  is the internal angle at the vertex.

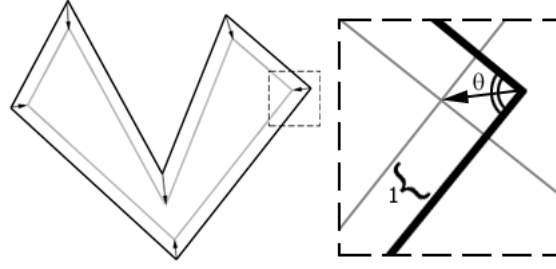


Figure B.2. Angle-bisecting vectors

The time taken for an angle-bisecting vector to hit a wavefront  $w_i$  can be approximated by calculating the required hit time between the vector and a halfspace  $b_i$ , which moves inwards perpendicular at unit speed. The halfspace  $b_i$  is defined by a partitioning plane that coincide with the edge of the corresponding wavefront  $w_i$ . This approximation gives incorrect answer occasionally. Figure B.3(a) shows an angle-bisecting vector hits a halfspace in a short time; however, the actual location of collision is outside of the wavefront. Therefore it is required to check whether the collision point is actually located inside the corresponding wavefront. The problem however is that we do not know the exact shape of a wavefront until we have completed the calculation of the straight skeleton.

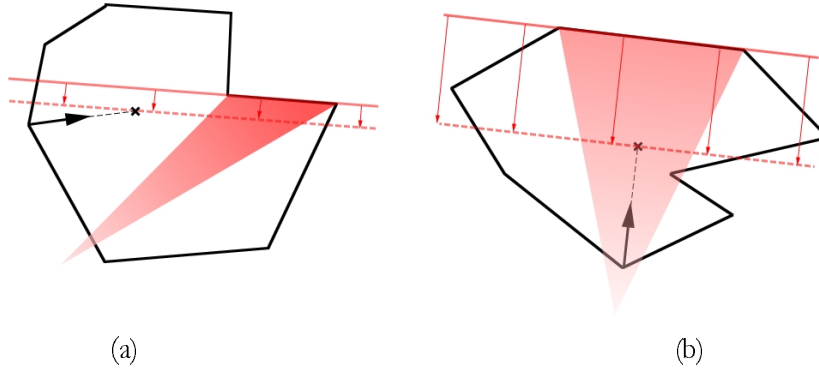


Figure B.3. Calculating hit time between an angle-bisecting vector and a wavefront

Fortunately, we can approximate the shape of a wavefront  $w_i$  by a pseudo-wavefront  $w'_i$ , which is the space clipped by the edge of  $w_i$  and the two adjacent angle-bisectors. Pseudo-wavefronts are shown as the red regions in the figure B.3. Figure B.3(a) shows the case in which the collision point is not inside the pseudo-wavefront. Since the pseudo-wavefront is guaranteed to be a super-set of the corresponding wavefront, we know immediately that the angle-bisecting vector will not hit the wavefront. Figure B.3(b) shows the case in which the collision point is inside the pseudo-wavefront but not the actual wavefront. This potential fault, however, will not affect the correctness of the solution, since we are interested only in the minimum collision time. If an angle-bisecting vector  $v_i$  hits a halfspace  $b_j$  within the pseudo-wavefront  $w'_j$  but not the corresponding wavefront  $w_j$  at time  $t_i$ , there must exist another collision between  $w_j$  and some other angle-bisecting vector  $v_k$  at time  $t_0$ , which cause the shape of  $w_j$  to change, while  $t_0 < t_i$ . The minimum collision time obtained with this heuristic must therefore be correct.

## Bibliography

---

- [AA96] Aichholzer, O. and Aurenhammer, F., Straight Skeletons for General Polygonal Figures in the Plane, *Proceedings of Second Annual International Conference Computing and Combinatorics (COCOON '96)*, Lecture Notes in Computer Science 1090, Springer, 117-126, 1996.
- [BB82] Ballard, D. and Brown, C., *Computer Vision*, Prentice-Hall, 1982.
- [CC78] Catmull, E. and Clark, J., Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshed, *Computer Aided Design*, Vol. 10, No. 6, September 350-355, 1978.
- [CECIL] <http://www.cecil.edu/>
- [Chai74] Chaikin, G., An Algorithm for High Speed Curve Generation, *Computer Graphics and Image Processing*, Vol. 3, 346-349, 1974.
- [Colb90] Colburn, S., Solid Modeling with Global Blending for Machining Dies and Patterns, *SAE technical paper series*, SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001 U.S.A., April 1990. 41<sup>st</sup> Annual Earthmoving Industry Conference.
- [Dans96] Dansted, Convolutional Smoothing of Polyhedra, *Master's Thesis*, Department of Computer Science, University of Auckland, 1996.
- [DS78] Doo, D. and Sabin, M., Behaviour of Recursive Division Surfaces near Extraordinary Points. *Computer Aided Design*, Vol. 10, No. 6, September 356-360, 1978.
- [DLG90] Dyn, N., Levin, D. and Gregory, J. A. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM Transactions on Graphics*, Vol. 9, No. 2, April 160-169, 1990.
- [FS97] Fowler, M. and Scott, K., *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley, 1997.
- [FV96] Foley, J. D, van Dam A. et al., *Computer Graphics: Principles and Practice*, Addison-Wesley, 1996.
- [HDD93] Hoppe H., DeRose, T. and Duchamp, T., Mesh Optimization, In *Computer Graphics Proceedings*, Annual Conference Series, 35-44, 1993.
- [HKD93] Halstead, M., Kass, M. and DeRose, T., Efficient, Fair Interpolation using Catmull-Clark Surfaces. In *Computer Graphics Proceedings*, Annual Conference Series, 35-44, 1993.
- [HL97] Hoschek, J. and Lasser, D., *Fundamentals of Computer Aided Geometric Design*, A K Peters Wellesley, Massachusetts, 1997.
- [JM96] Joy, K. I. and MacCracken R., The Refinement Rules for Catmull-Clark Solids, *Technical Report CSE-96-1*, Department of Computer Science, University of California, Davis, 1996.

- [KT96] Kalvin, A. D. and Taylor, R. H., Superfaces: Polygonal Mesh Simplification with Bounded Error, *IEEE Computer Graphics and Applications*, Vol. 16, No. 3, May 64-77, 1996.
- [LC87] Lorenson, W., and Cline, H., Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, Vol. 21, No.4, July 163-169, 1987.
- [Lobb96] Lobb, R. J., Quasiconvolutional Smoothing of Polyhedra, *The Visual Computer*, 12, 373-389, 1996.
- [Loop87] Loop, C., Smooth Subdivision Surfaces Based on Triangles. *Master's thesis*, Department of Mathematics, University of Utah, 1987.
- [Loop94] Loop, C., Smooth Spline Surfaces over Irregular Meshes. In *Computer Graphics Proceedings*, Annual Conference Series, 303-310, 1994.
- [Nasr87] Nasri, A. H., Polyhedral Subdivision Methods for Free-Form Surfaces, *ACM Transactions on Graphics*, Vol. 6, No. 1, January 29-73, 1987.
- [Nasr91] Nasri, A. H., Surface Interpolations on Irregular Networks with Normal Conditions. *Computer Aided Geometric Design*, Vol. 8, 89-96, 1991.
- [PR97] Peters, J., and Reif, U., The Simplest Subdivision Scheme for Smoothing Polyhedra, *ACM Transactions on Graphics*, Vol. 16, No. 4, October 420-431, 1997.
- [Reif95] Reif, U., A Unified Approach to Subdivision Algorithms near Extraordinary Points. *Computer Aided Geometric Design*, Vol. 12, 153-174, 1995.
- [Ries75] Riesenfeld, R., On Chaikin's algorithm. *IEEE Computer Graphics and Applications*, Vol. 4, No. 3, 304-310, 1975.
- [Sier92] Sieradski, A. J. *An Introduction to Topology and Homology*, Boston, Mass, 1992.
- [VRML97] The Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1:1997.
- [Wuen96] Wüensche, B. C., A Fast Polygonization Method for Quasi Convolutional Smoothed Polyhedra, *Master's thesis*, Department of Computer Science, University of Auckland, 1996.
- [ZSS96] Zorin, D., Schröder, P. and Sweldens, W., Interpolating Subdivision for Meshes with Arbitrary Topology, In *Computer Graphics Proceedings*, Annual Conference Series, 189-192, 1996.