

# Interactive Design of Triangular Meshes

**PhD student:** Anand Murugappan (anandm@cc.gatech.edu)

**Project advisor:** Prof. Jarek Rossignac (jarek@cc.gatech.edu)

**Abstract** - Triangle meshes are very important entities in computer graphics and form the building blocks for variety of entities like terrains & surfaces and solid objects as shown in the example below. Triangle meshes are very suitable simply because of the fact that triangles can approximate almost any surface and at the same time remain computationally cheap. It is essential to understand what triangle meshes can offer and explore new avenues like how can we compress triangle meshes and achieve better performance when it gets to transferring model details over the network. This project is an attempt towards building simple, easily extensible triangle mesh editing tool that help make this happen in a far more easy & generic way. This report discusses the motivation and design of the tool.

## 1. Introduction

Among the different polygons that can be used to represent objects - triangles emerge out to be the simplest and most elegant. Triangle meshes have been around for a long time. Researchers have proposed several algorithms which are based on manipulating triangle meshes. Some of them include those to improve various factors like look & feel, compression for better frame rates and transfer of images over a network. There are good tools like "Art of illusion" [1] and several CAD tools which have been built for the purpose of constructing objects with triangle meshes but none of the tools that I am aware of are simple and small enough in terms of code to be easily modified to experiment new ideas. This is exactly what this project is intended to solve - design and build a simple and small triangle mesh editor which would let people interactively design triangular meshes and if needed also let people use the code and extend it to fit additional needs.

This report is divided into 3 main sections. In section 3 related work and concepts that are fundamental to the design of this tool are discussed. Section 4 contains a description of the data structures used. Section 5 contains the results of the experiment and a few snapshots of objects built with the tool. The report ends with what future work is possible.

## 2. Related work

The idea of triangular meshes is not new and many researchers in the past have put in a lot of effort on this front. A very simple and efficient approach to representing and manipulating triangular meshes can be found in the Edgebreaker paper [2].

In the EdgeBreaker paper [2] the authors use 3 tables to represent a triangle mesh. What makes this efficient is the fact modifications to these tables to reflect manipulation of the triangle mesh can be done almost in constant time.

A quick overview of these 3 tables:

- **G table** - Geometry table (G[i] represents the coordinates of the point in 3D space)



- **V table** – Incidence table ( $V[i]$  represents a vertex. The value stored is nothing but an index into the G table)
- **O table** – Opposite table ( $O[i]$  represents the vertex opposite to  $i$ . This is an index into the V table)

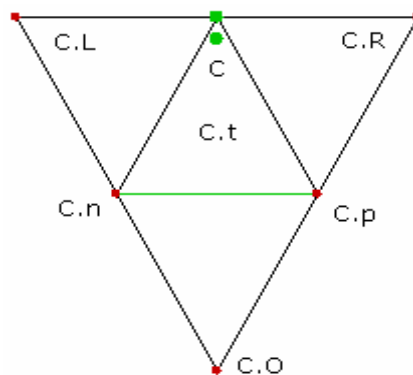
This project implements triangular meshes totally based on this idea.

### 3. System Model

#### Definitions:

Before the system model is discussed one needs to be familiar with a few definitions: These definitions are as found in the EdgeBreaker paper [2].

1. *Triangle* - A node of the connectivity graph representing a closed point-set that is the convex hull of three noncollinear vertices.
2. *Corner* - A corner  $c$  associates a vertex  $c.v$  with one of its incident triangles  $c.t$ .
3. *Edge* - A link in the connectivity graph representing a relatively open line segment joining two vertices of a mesh, but not containing them.
4. *Incidence table* - An array  $V$  of  $3t$  entries, where  $V[c]$  is denoted  $c.v$  and contains a vertex Id Entries for  $c.p$ ,  $c$ , and  $c.n$  are consecutive in  $V$ . Therefore,  $c.t$  is the integer division  $c.t \div 3$ ;  $c.n$  is  $c-2$ , when  $c \bmod 3$  is 2, and  $c+1$  otherwise and  $c.p$  is  $c.n.n$ .
5. *Corner operators* – These are operators required to navigate through the triangle mesh:
  - *Next operator ( $c.n$ )* – The next anticlockwise corner.
  - *Previous operator ( $c.p$ )* – The next clockwise corner.
  - *Opposite operator ( $c.o$ )* – Two corners  $c$  and  $b$  are opposite when  $b.n.v=c.p.v$  and  $b.p.v=c.n.v$ . The opposite corner of  $c$ , denoted  $c.o$ , is a corner Id stored as the entry  $O[c]$  in the O table.
  - *Left operator ( $c.l$ )* -  $c.n.o$
  - *Right operator ( $c.r$ )* -  $c.p.o$
6. *Cascading corner operators* – Corner operators defined above can be cascaded. For instance  $c.n.v$  is the same as  $(c.n).v$
7. *Corner table* – The arrays  $V$  and  $O$  together are referred to as the corner table.

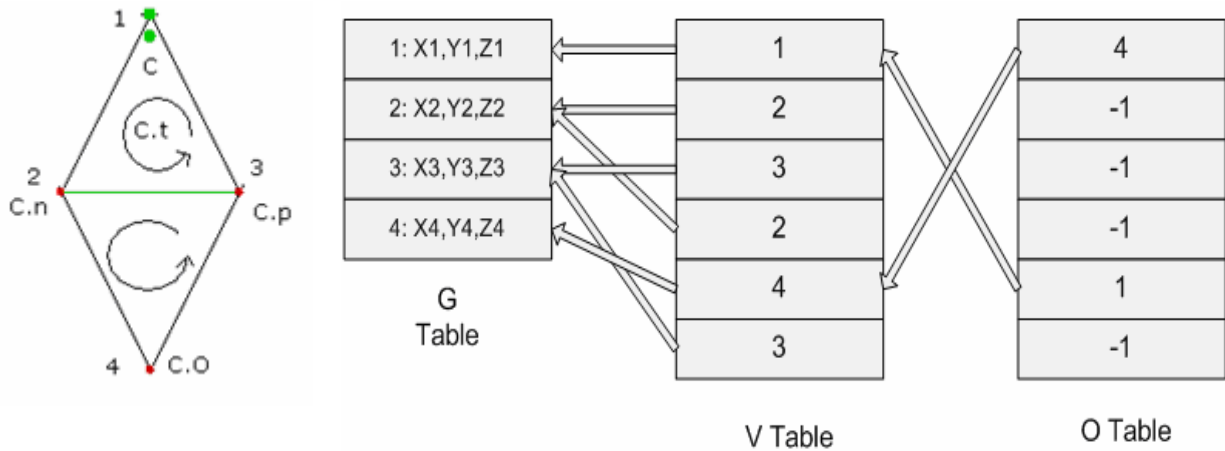


**Fig 1. Corner table representation of triangle meshes**



## 4.1 Data Structures:

This section covers in detail what the 3 tables – O, V and G represent.



**Fig 2.**

An illustration of how the 3 tables come together to represent any triangular mesh!

- **G Table:** This contains only the coordinates of the various vertices in 3D space.
- **V Table:** This table is organized as triplets. Each triplet comprised of 3 rows represents one triangle in the anticlockwise direction. The values in this table point to the G table.
- **O Table:** This table has a one to one correspondence with the V table.  $O[i]$  has the value of the index of the opposite corner in the V table. -1 represents the state where there is no opposite corner.

## 4.2 Design:

The design of the tool revolves around manipulating the data structures above to grow, modify and navigate about the triangular mesh.

### *Creation of new triangles:*

To create a triangular mesh the parallelogram prediction is used.

Parallelogram prediction is as simple as  $c.n.v.g + c.p.v.g - c.v.g$ . Based on this the opposite vertex is determined and created. Once the vertex is created in G table the corresponding edges are added to the V table to form the new triangle. The O table is updated with the new vertex 'o' being the opposite of the current vertex 'c' and the current vertex 'c' has the 'o' for its opposite. As expected if the current vertex already has an opposite i.e. anything other than -1 in this representation, no new vertex is created.

Steps involved in creation of a new triangle:

1. Identify the new vertex by using the parallelogram prediction  
 $C'.g = c.n.v.g + c.p.v.g - c.v.g$
2. Add a new triangle to the incidence table by doing the following:
  - v.last++;
  - v.last = v.n
  - v.last++
  - v.last = g.last



```
v.last++  
v.last = v.p
```

Above v.last and g.last represent the last indices into the V and G table that are filled. The new triangle will share the next and previous vertices of the current triangle and the 3<sup>rd</sup> vertex will be predicted by the parallelogram method.

3. Modify the opposite table by doing the following:

```
o.last = -1  
o.last = c  
o.last = -1  
c.o = v.last-2
```

The new triangle is new so only one of the vertices will have an opposite. Also the new vertex created will be the opposite for the current vertex.

#### *Navigation:*

Next, Previous, Left, Right and Opposite operators are implemented as defined in the definitions section.

#### *Modification:*

Additionally one should be able to modify any vertex along the x, y and z axis. Apart from this Merge operator is implemented. Merge simply identifies the vertex closest to the current vertex and merges the 2. The corresponding changes to the 3 tables are implemented. [See fig 5.1 and 5.2]

#### *Visualization:*

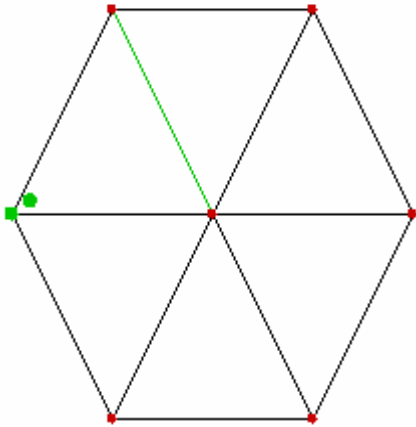
Apart from this for suitable visualization one should be able to toggle off and on the coordinate axis and finally also be able to fill and shade the object created. Also keystrokes are to be supported for viewers to rotate the object to view the in angle desired.

## **5. Results:**

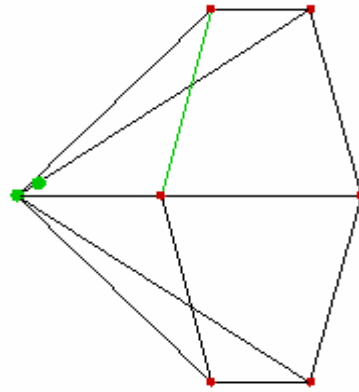
The design was implemented in processing [3] which is a simple and elegant tool to develop java based graphics tools. The resultant code is small and the tool can also generate an applet that can be posted out on the web for interested people to try out without having to install any additional software.

The underlying code is very small and additional material can be added to it easily to facilitate experimentation with new ideas involving triangular meshes.

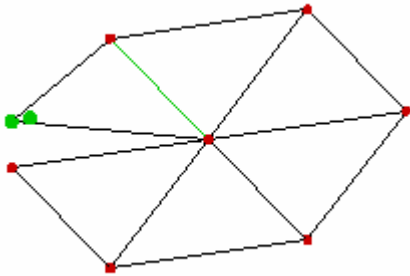
The following are some of the objects created using this tool:



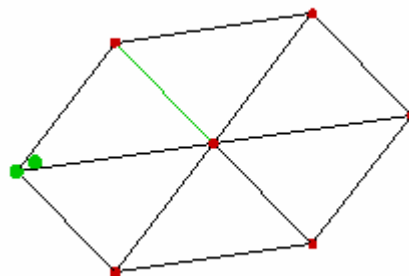
**Fig 3.** Simple Cone as viewed from top



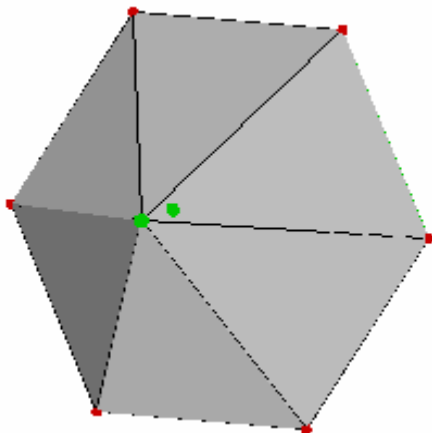
**Fig 4.** Same cone with the camera rotated about Y axis



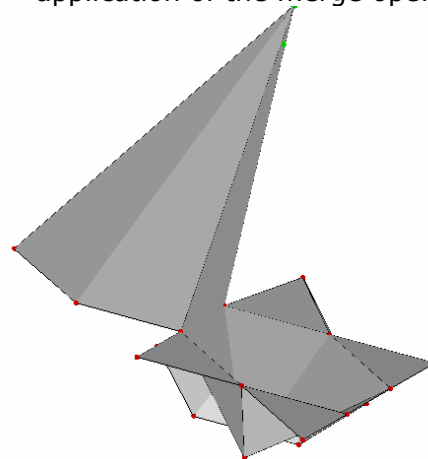
**Fig 5.1** 2 different vertices before merge operator is applied



**Fig 5.2** The current vertex has been merged with the nearest vertex after application of the merge operator



**Fig 6.** Cone with lighting



**Fig 7.** Slightly more complex objects built out of this tool

The red dots above indicate vertices and the green dot the current vertex. The green edge and the green vertex help in uniquely identifying the current corner. To make

this simpler an additional green dot is created close to the vertex on the side of the triangle it belongs to.

Creation of the objects as shown above hardly takes a few minutes and is very easy. Objects like cones, terrains etc can easily be created with this tool. However, creation of objects that require accurate folding of faces like those to construct a rectangular box or a torus is still difficult and the tool needs improvement.

### Keystrokes:

The table below gives an overview of the various key strokes currently supported by the tool.

<b>Creation</b>		
C	Create	The opposite vertex is determined based on parallelogram prediction
<b>Navigation</b>		
N	Next	Goto next corner (anticlockwise) – c.n
P	Previous	Goto previous corner (anticlockwise) – c.p
L	Left	Goto left corner. Same as c.n.o
R	Right	Goto right corner. Same as c.p.o
O	Opposite	Goto the opposite corner – c.o
<b>Modification</b>		
A	Increase X	Moves the current vertex along X axis
D	Decrease X	Moves the current vertex along X axis
W	Increase Y	Moves the current vertex along Y axis
S	Decrease Y	Moves the current vertex along Y axis
E	Increase Z	Moves the current vertex along Z axis
Z	Decrease Z	Moves the current vertex along Z axis
M	Merge	Identifies nearest vertex and merges it
<b>Visualization</b>		
T	Rotate X +ve	Rotates camera about X axis
G	Rotate X -ve	Rotates camera about X axis
Y	Rotate Y +ve	Rotates camera about Y axis
H	Rotate Y -ve	Rotates camera about Y axis
U	Rotate Z +ve	Rotates camera about Z axis
J	Rotate Z -ve	Rotates camera about Z axis
<b>Miscellaneous</b>		
X	Fill & Light	Fills up the mesh created with grey color and lights it up.
V	Toggle Axis	Toggles on and off the coordinate axis

### Future work:

The following are some of the areas that are suitable candidates for future work:

- Improving usability by allowing transformations and modification of the triangles with local coordinates.
- Improving usability by automatic focusing of camera such that the view is normal to the current triangle and the current edge is aligned horizontally.
- Smoothing of edges to be able to build more realistic objects.

**Conclusion:**

This report covers the design of a simple and short tool built to interactively design triangular meshes. The main philosophy, data structures used, design and implementation results have been discussed.

**References:**

- [1] "Art of Illusion"- <http://www.artofillusion.org/artgallery>
- [2] Edgebreaker - Connectivity compression for triangle meshes - Jarek Rossignac
- [3] Processing - <http://www.processing.org/> - Ben Fry and Casey Reas

