Sterling Systems Pvt. Ltd.
Excellence I Innovation I Quality

Registered Office:
Bhagyashree Bangalow, Sadgurunagar, Patas Road, Baramati-413102. Dist. Pune.
Development Center:
Sandeep Complex, Shop No.19, Near HP Petrol Pump, Bhigwan Road,
M.I.D.C. Baramati-413133. Dist. Pune. Maharashtra India ☎ +91 2112 244109.
info@sterlingsys.com · www.sterlingsys.com

17 June 2013

To Whom It May Concern

This is to certify that Mr. Sagar Gandhi is undergoing an Industrial Traineeship in our organization and has successfully completed IT project on June 30, 2013; under the guidance of Dr. Vijay D. Gokhale.

The details of which are provided below:

Start Date	: 02 January, 2013
Completion Date	: 30 June, 2013
Title of Project	: Polygon Reduction Tool
Technologies Used	: OpenGL 1.1, C++, Visual Studio 2008

During this period his performance is found Good.

Yours Sincerely,

For Sterling Systems Pvt. Ltd.

For Sterling Systems Pvt. Ltd.
Authorised Signatory

Interdisciplinary School of Scientific Computing

University of Pune

CERTIFICATE

This is to certify that,

Mr. Sagar Gandhi

Has completed his/her Industrial project report entitled **"Polygon Reduction Tool"** required for Semester-IV of M.Sc. in Scientific Computing course conducted by Interdisciplinary School of Scientific Computing, University of Pune during the academic year 2011-2012.

Dr. Anjali Kshirsagar
Director,
Interdisciplinary School of Scientific Computing,
University of Pune,
Pune-411007

Date:

Acceptance

This manuscript has been read and accepted in satisfaction of the Industrial Training Project requirement of M.Sc (Scientific Computing) 2011− 2012, Interdisciplinary School of Scientific Computing, University of Pune.

Examination Committee:

Name	Signature
1._____	_____
2._____	_____
3._____	_____

Place:

Date:

Index

1. Company Profile

1.1 About Sterling Systems

We are today living in an era where we cannot imagine a world without technology. The breakthrough technological developments have simplified our life in more than one way. Technological advancements have shown substantial growth in each and every field of humanity be it communication systems, astronomy, infrastructure, medical fields, automobiles, entertainment or media.

Over time technology has also become easily accessible. And with this, the only way businesses can make a mark is by partnering with technology solution providers who can provide optimum solutions that successfully help to achieve business objectives.

At **Sterling Systems**, our constant endeavor is to develop cutting edge solutions for businesses and consumers with an aim to make life better, easier and more interesting.

Sterling Systems is a leading software product development company based out of India. Our experts are specialized in developing products based on **Computer Graphics** which includes **Interactive 3D Modeling** and **Augmented Reality.**

A bunch of technology enthusiasts, we strongly believe in the "Client First" work ethic. With a vision to be one of the best software product development companies across the globe, we constantly strive to create and sustain strong business relationship with our genuine, value based and innovative solutions and at par customer service.

1.2 Guide's Profile

Dr.Vijay Gokhale

Technical Director

Dr. Vijay D. Gokhale is currently working as a technical director, in all of the following companies:

- Pi Tech Solutions [2003-now] : http://www.piinfotech.com/
- Sunbeam Institute of Information Technologies (Training company) [2007-now] : http://www.sunbeaminfo.com/
- Sterling Systems private limited [2011-now] : http://sterlingsys.com/

Domain-wise description of software skill set of Dr. Gokhale Sir

- Win 32 SDK and COM: He has done extensive project work using these two technologies. He has used almost all services that are available through these technologies at one point or the other during his project work. (e.g. Winsock, Win Comm, Win MM, Band object, IE plug in, office plug-ins, activeX, TAPI, MAPI, SAPI etc.)
- Window devices drivers (VxD's to WDF).
- "IX" World :
    - He has studied the internals of all main ix systems viz. Unix, BSD versions, Solaris, Linux and Mac OS X. He has written device drivers ranging from serial port to GPU on various ix based systems.
    - He is writing an ix based operating system of his own called "Viramix" for the educational purposes.
    - Special mention must be made of his operating system zoo. He is very fond of installing multiple operating systems on single machine and he has installed around 20 operating systems on his server machine including various versions of Windows and Linux distributions, Hacintosh, FreeBSD and OpenSolaris.
- Embedded platform :
    - Hardware Architectures: Intel, ARM.
    - Operating System/software Platforms: Windows CE, iPhone and Android, PocketPC SDK and uCLinux.

- Multimedia programming: He has handled various media formats programmatically and has a sound knowledge of underlying theory of digital signal processing. He is Apple Final Cut Studio Master. Path he had chosen for the certification is as follows:
    - Final Cut Pro: Real time editing of DV, SD HD video formats.
    - Color: Color grading application
    - Motion: Real-time motion graphics design
    - Soundtrack Pro: Advanced audio editing and sound design
    - DVD Studio Pro: Encoding , authoring and burning
- OpenGL and DirectX: He has programmed extensively in openGL using fixed function as well as programmable pipeline on all three major platforms viz. Windows, Mac and Linux. He uses native windowing on all platforms, He has equal amount of work on DirectX9, 10, 11. He has written device driver for nVidia's GeForce GTX 295 GPU for Hacintosh operating system.
- Parallel Programming and GPGPU: He has worked on CUDA and openCL technologies. He can use GPU technologies like CUDA for general purpose programming like handling huge data sets.

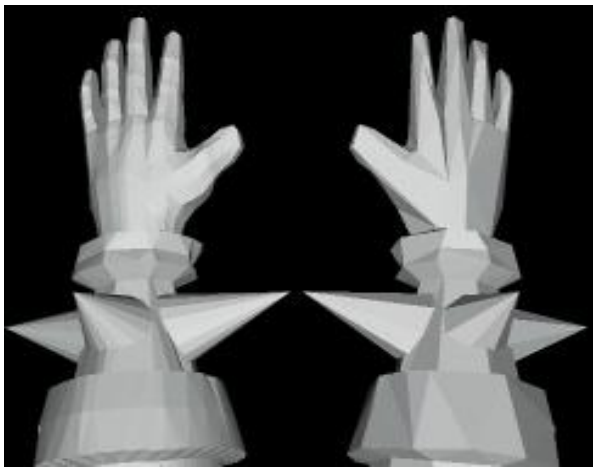
Apart from role as a technical director in above listed companies, he runs private tuition class, where he teaches "Unix System Design" in one semester and "Win 32 SDK, COM and .NET internal architecture" in another. [2003-now] He runs this private class strongly adhering to "Gurukul"* ethos. One of the salient features of his teaching is that he imparts not only the technical content but also "Sanskars"**. He is very respected and adored amongst his students. These sentiments are captured in the web-site, http://astromedicomp.org/ which is run by the students and for the students.

[*Gurukul refers to an ancient Indian system of schooling, where pupils stay at Guru's residence for a decade or so and learn a particular subject with a single minded devotion. Also, Guru is more than just a teacher who takes money and imparts technical contents. Guru imparts Sanskaras.

****Notion of Sanskar, is hard to describe accurately in English, but roughly means 'imparting wisdom, not actively but passively through one's own actions and deeds.' Or in the manner described by Kahlil Gibran, "Wise teacher does not bid you to enter the house of his wisdom, but rather leads you to the threshold of your own mind."]**

To Know More About Dr. Vijay Gokhale, as a person, teacher, Technical Expert, please visit http://astromedicomp.org/

2. Project Specification



2.1 Introduction

For a Graphics Programmer, 3D polygonal models are part of his/her daily life. Rendering massive 3D models in real-time has long been recognized as a very challenging problem because of the limited computational power and memory space available. Most existing rendering techniques, especially level of detail (LOD) processing, have suffered. But if we actually dig a little deeper, we do not really need to render every model in the scene with very high detailing. So we need something which will convert these high detailed 3D models to low detailed one, but with model looking reasonably similar.

2.2 Introduction to OpenGL

OpenGL (**Open G**raphics **L**ibrary) is a cross-language, multi-platform API for rendering 2D and 3D computer graphics. The API is typically used to interact with a GPU, to achieve hardware-accelerated rendering. OpenGL was developed by Silicon Graphics Inc. (SGI) from 1991 and released in January 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games. OpenGL is managed by the non-profit technology consortium Khronos Group.

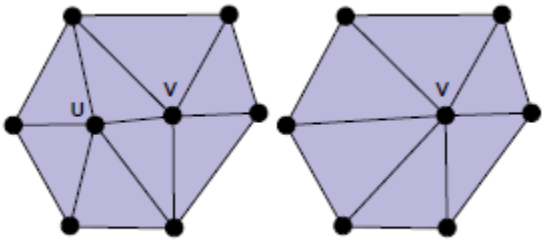
Features of OpenGL

Basic Features:	**Advanced Features:**
--Drawing Primitives	--Texture Mapping
--Transformations	--Vertex Arrays and Vertex Buffer Objects
--Color	--Blending Effects
--Lighting	--Frame Buffer Manipulation
--Display Lists	

Above listed are some of the features of OpenGL 1.1, which is now deprecated, but still used heavily in the industry.

2.3 General Implementation

Rather than attacking this problem all by ourselves, we studied polygon reduction which was already done by some other researchers and whose algorithms were freely available. A lot of research has gone into this subject recently, and most of the better techniques are variations of the progressive meshes algorithm by H. Hoppe.

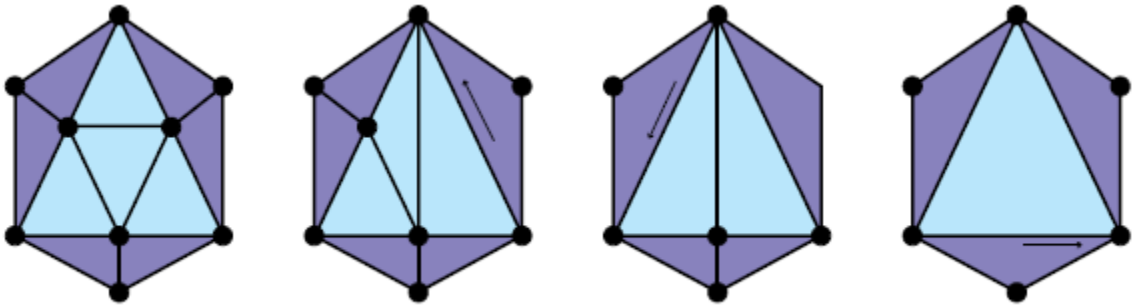


BEFORE AFTER

These techniques reduce a model's complexity by repeated use of the simple edge collapse operation, shown in figure above. In this operation, two vertices *u* and *v* (the edge *uv*) are selected and one of them (*u*) is "moved" or "collapsed" onto the other (in this case, *v*). The following steps implement this operation:

1. Remove any triangles that have both *u* and *v* as vertices (that is, remove triangles on the edge *uv*).
2. Update the remaining triangles that use *u* as a vertex to use *v* instead.
3. Remove vertex *u*.

This process is repeated until the desired polygon count is reached. At each step, one vertex, two faces, and three edges are usually removed. Figure below shows a simple example.



ORIGINAL FINAL

3. Analysis

3.1 Motivation

- The results of presently polygon-reduction tool may not meet your specific needs, and there is a need to build own tool.
- Current polygon-reduction tool may not produce the morph information that you require for smooth transitions between different levels of detail.
- To automate your production pipeline so that the artist has to create only one reasonably detailed model, and the game/rendering engine does the rest.
- In VRML browser, if user would want to provide a menu option for reducing those huge VRML files placed on the Web by supercomputer users who didn't realize the frame rate would be slower on a home PC.
- Special effects in game modify the geometry of objects, bumping up your polygon count and requiring a method by which engine can quickly reduce polygon counts at run time.

3.2 Work Done (Approach and Implementation)

- The trick to producing good low-polygon models is to select the edge that, when collapsed, will cause the smallest visual change to the model. Researchers have proposed various methods of determining the "minimal cost" edge to collapse at each step. Unfortunately, the best methods are very elaborate (as in, difficult to implement) and take too long to compute and by and large this is the basic problem we were trying to solve; Performance!

- So to start with, it made more sence to get rid of small details first, and it was noticable that fewer polygons are needed to represent nearly coplanar surfaces while areas of high curvature need more polygons. Based on these heuristics, we defined the cost of collapsing an edge as the length of the edge multiplied by a curvature term. The curvature term for collapsing an edge *uv* is determined by comparing dot products of face normals in order to find the triangle adjacent to *u* that faces furthest away from the other triangles that are along *uv*. The edge cost formula in more formal notation is

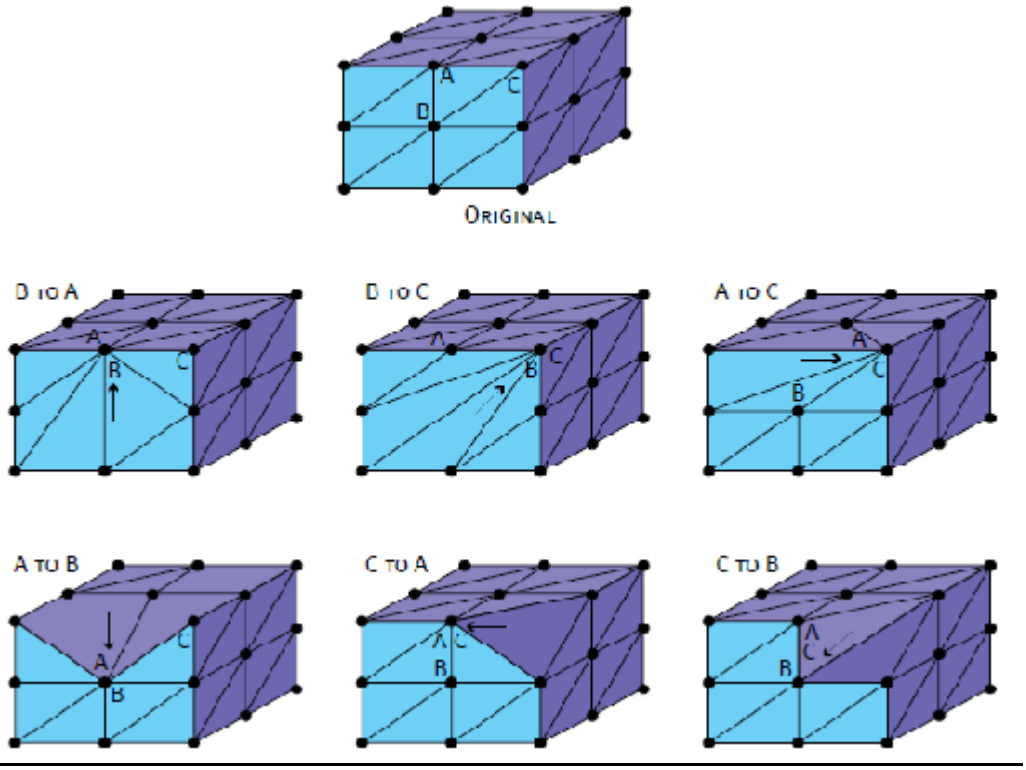
$$\text{cost}(u,v) = \|u - v\| \times \max_{f \in Tu} \left\{ \min_{n \in Tuv} \left\{ \left(1 - f.normal \bullet n.normal\right) \div 2 \right\} \right\}$$

where *Tu* is the set of triangles that contain *u* and *Tuv* is the set of triangles that contain both *u* and *v*.

Algorithm's Features:

- This algorithm balances curvature and size when determining which edge to collapse
- The cost of collapsing vertex *u* to *v* may be different than the cost of collapsing *v* to *u*.
- The algorithm is effective for collapsing edges along a ridge. Although the ridge may be a sharp angle, it won't matter if it's running orthogonal to the edge.

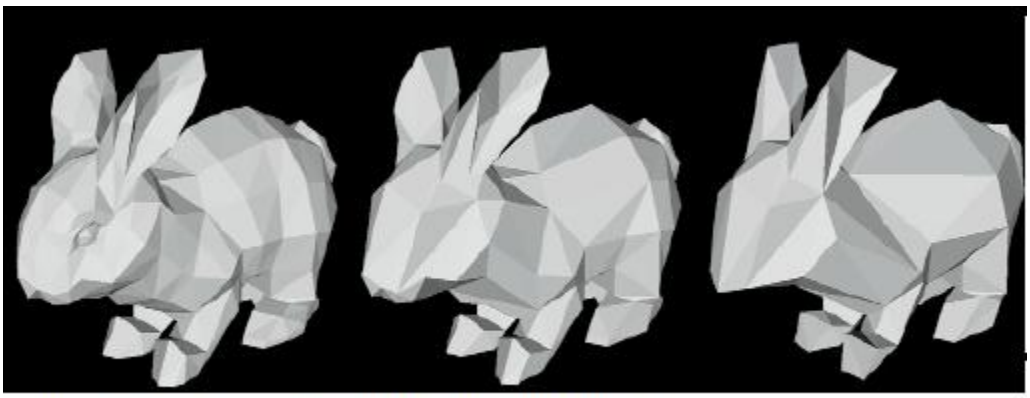
Example:



Clearly, vertex B, sitting in the middle of a flat region, can be collapsed to A or C.
Corner vertex C should be left alone.
It would be bad to move vertex A, sitting along the top ridge, onto interior vertex B.
However, A could be moved (along the ridge) onto C without affecting theoverall shape of the model.

Results:

- The effectiveness of a polygon reduction algorithm is best demonstrated by showing a model before and after it has been simplified. For 3D games, an appropriate (and challenging) test of an algorithm is how it demonstrates its prowess by generating models that use only a few hundred polygons.
- Figure below shows a bunny model. The initial version (left) of the model contains 453 vertices and 902 polygons. Reductions to 200 (center) and 100 (right) vertices are shown. Hopefully, you'll agree that the models look reasonably good given the number of polygons used in each image.



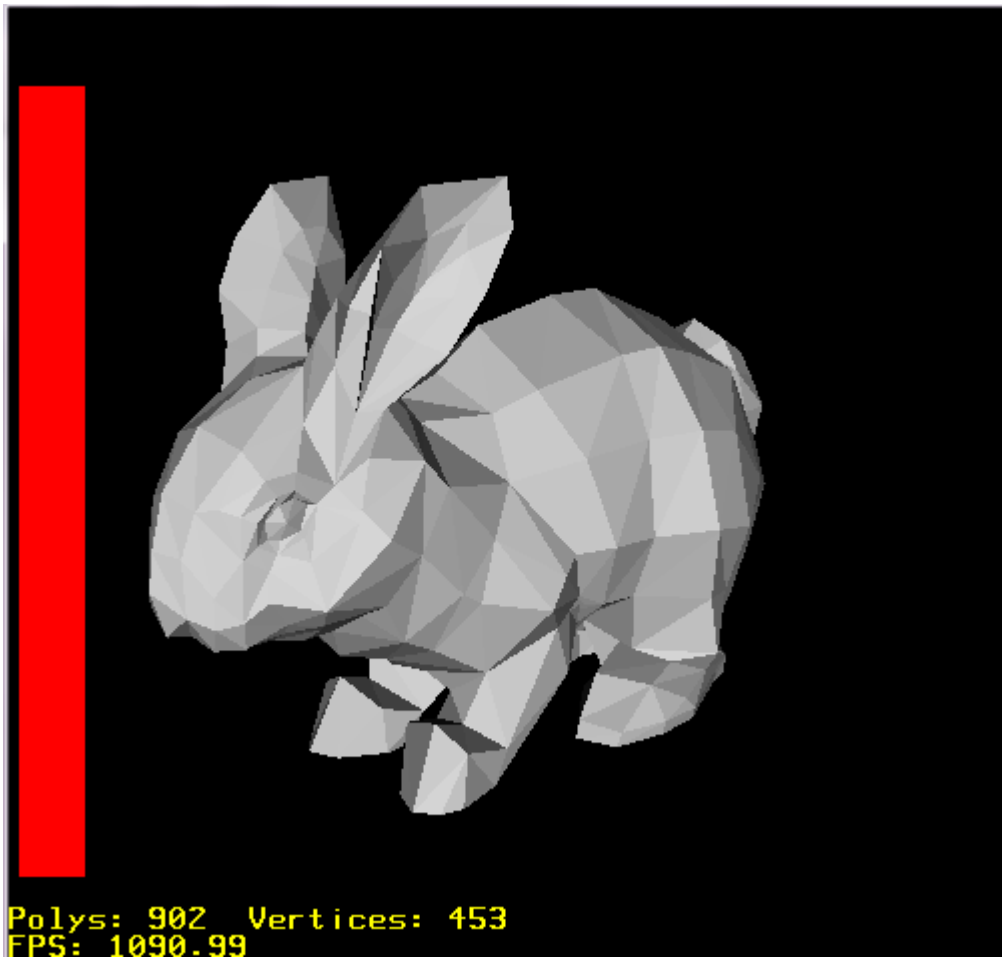
Bunny Model at (Left to Right) 453, 200, 100 vertices

3.3 Limitations and Alternatives

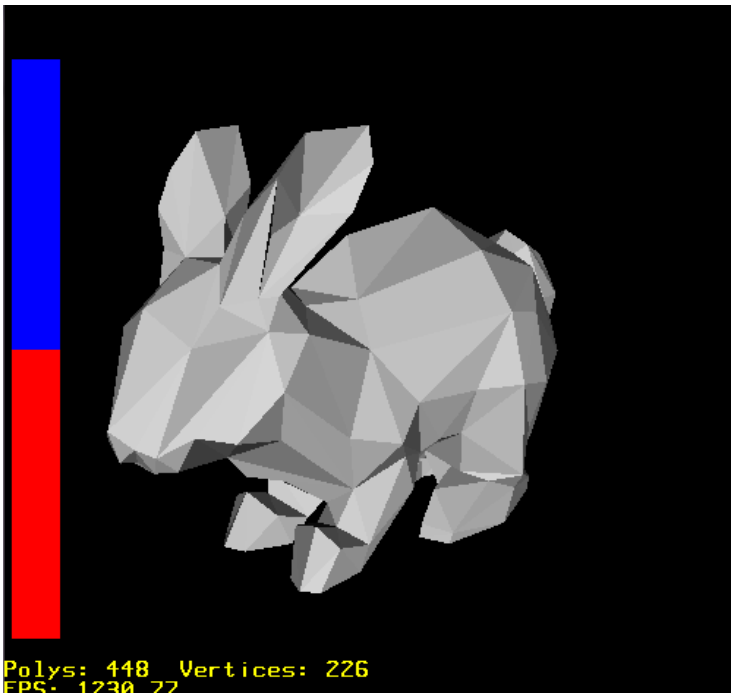
- Polygon reduction algorithms aren't the only way to create a model with fewer faces. Artists will always be able to do a better job of representing a model using fewer polygons than any reduction algorithm.
- One reason is that algorithms have little or no higher-level understanding of the model. An artist, on the other hand, knows the object that he or she is creating and can make careful aesthetic decisions as he or she manually reduces the face count.
- The human visual system is biased towards certain details, our simple algorithm merely compares a few dot products and edge lengths, and obviously doesn't have the intelligence to place automatically varying amounts of importance on different pieces to optimize for human perception.
- Another technique for doing LODs in a game is to represent an object's geometry using parametric surface patches, which are tessellated on the fly to the desired detail.
- Certainly, these surface-based methods are preferable (and probably optimal too) but unfortunately, using curved parametric surfaces isn't always appropriate. Furthermore, jagged objects aren't good candidates for use with curved surface patches because the number of surfaces would be no less than the number of polygons required.

4. Screenshots

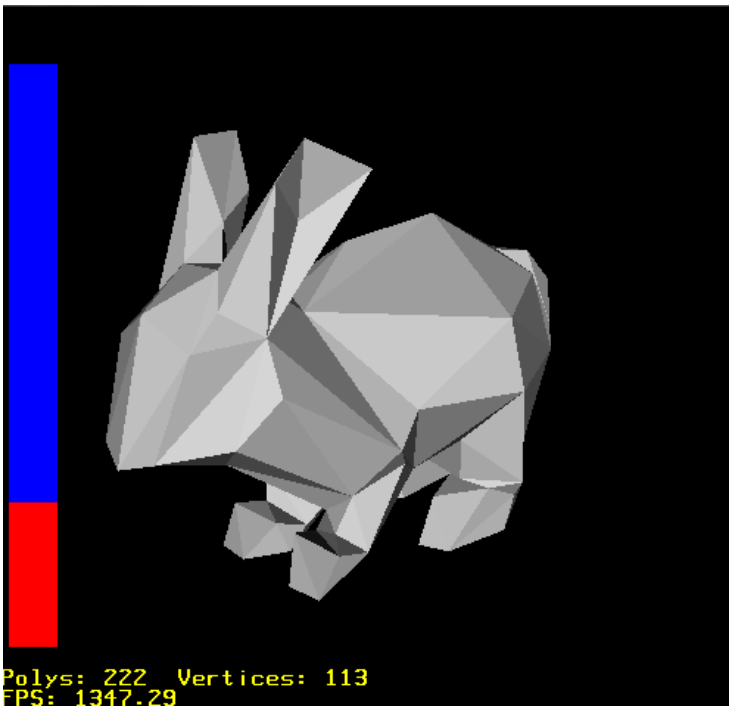
4.1 Bunny model -- Original



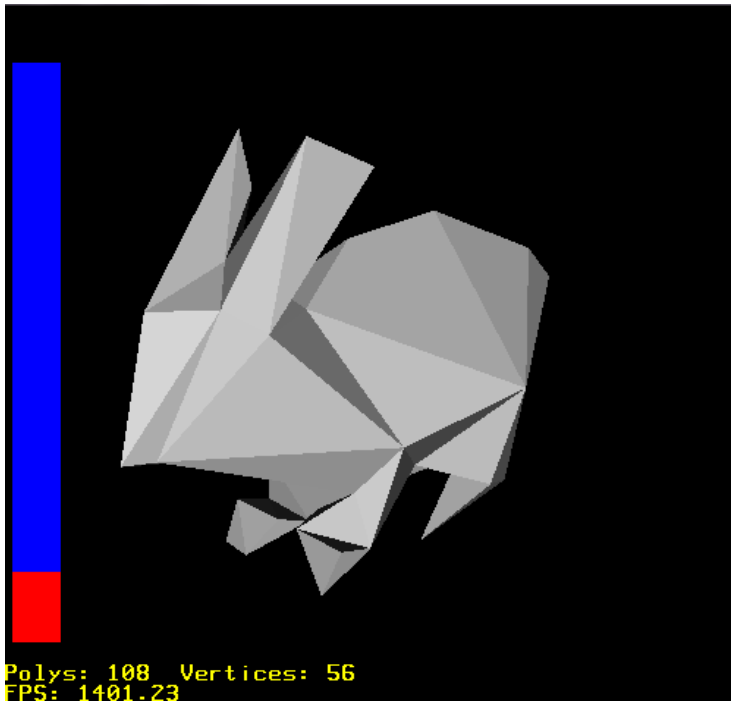
4.2 Bunny Model – Half the vertices



4.3 Bunny Model – One fourth of vertices



4.4 Bunny Model – One eighth of vertices



5. Other Projects worked on in Same Duration

5.1 Volume Rendering – Ray Tracing

5.1.1 – Introduction

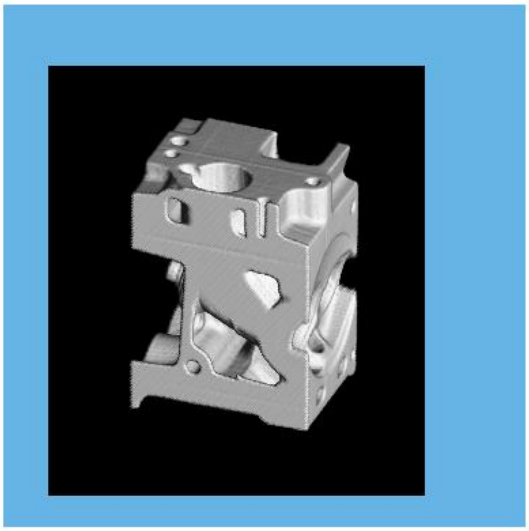
In scientific visualization and computer graphics, **volume rendering** is a set of techniques used to display a 2D projection of a 3D discretely sampled data set.

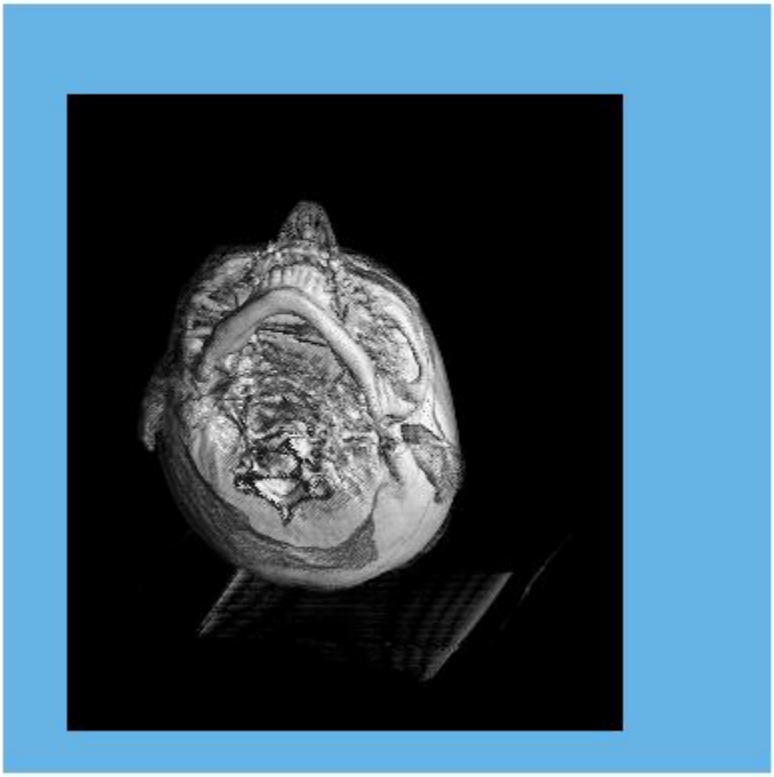
A typical 3D data set is a group of 2D slice images acquired by a CT, MRI, or MicroCT scanner. Usually these are acquired in a regular pattern (e.g., one slice every millimeter) and usually have a regular number of image pixels in a regular pattern. This is an example of a regular volumetric grid, with each volume element, or voxel represented by a single value that is obtained by sampling the immediate area surrounding the voxel.

5.1.2 – Actual Work Done

- Implemented a Maximum Intensity Projection ray-caster using the ray-transform method on GPU.
- Rendering could be done both in orthographic and perspective projection mode.
- Also wrote average ray-caster using same ray-transform method.
- First we read volumetric .vol file, it was a normal parser.
- Then we found the shape of the volume and performed 3D scan conversion within this range to cast rays.
- Later on, little image processing was done while rendering.

5.1. 3 – Results





5. Appendices

5.1 Targeted Platforms and Technology details

- Windows 7 (But can be ported to other OSs by using their windowing system)

- Platform - Windows 7
- Language, Technology & Tools used – OpenGL 1.1, C++, Visual Studio 2008

6. Bibliography

6.1 Websites

http://google.com
http://stackoverflow.com/

6.2 Books Used

- OpenGL(R) SuperBible. Comprehensive Tutorial and Reference 4th Edition (2007).pdf
- OpenGL Programming Guide - RedBook.pdf