ARMY RESEARCH LABORATORY

The Verification and Validation of the Ray-tracing of Bag of Triangles (BoTs)

by Charith Ranawake

ARL-CR-0761 **February 2015**

Under contract

W911NF-10-2-0076

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 22105

ARL-CR-0761 **February 2015**

The Verification and Validation of the Ray-tracing of Bag of Triangles (BoTs)

Charith Ranawake
Survivability/Lethality Analysis Directorate, ARL

Under contract

W911NF-10-2-0076

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
February 2015	Contractor Report	06/2014–08/2014

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
The Verification and Validation of the Ray-tracing of Bag of Triangles (BoTs)	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Charith Ranawake	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
US Army Research Laboratory ATTN: RDRL-SLB-S Aberdeen Proving Ground, MD 21005	ARL-CR-0761
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
POC: Geoffrey Sauerborn

14. ABSTRACT
This report will cover the testing of the Ballistic Research Laboratory–Computer-Aided Design's (BRL–CAD's) ray-tracer on the primitive Bag of Triangles (BoTs). Two programs were created to accommodate each component of this project. The first program created all possible triangles (BoTs) given a bounding box and step size that is specified by the user. The second program fired multiple rays toward various locations on each triangle in the database. The results of these tests were classified as a success or failure; the percentages of success were graphed against the different tolerance levels to determine the accuracy of the ray-tracer on BoTs. These data could be useful in understanding how BRL–CAD's ray-tracer performs in comparison to the Air Force's ray-tracer, FASTGEN, on BoTs.

15. SUBJECT TERMS
BRL–CAD, Ray-trace, BoTs

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
A. Report	b. ABSTRACT	c. THIS PAGE			Geoffrey Sauerborn
Unclassified	Unclassified	Unclassified	UU	18	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6178

Contents

List of Figures

List of Tables

Acknowledgments

The author wishes to acknowledge the mentorship of Mr Sean Morrison and Mr Geoffrey Sauerborn.

Student Bio

Charith Ranawake is attending the University of Maryland College Park as a freshman in Computer Science. Before working on Ballistic Research Laboratory–Computer-Aided Design (BRL–CAD) this summer, he worked as a volunteer programmer with a professor at George Mason University where he helped develop a computer-security software package. Charith is a graduate of Marriotts Ridge High School in Ellicott City, Maryland.

1. Introduction/Background

The Ballistic Research Laboratory–Computer-Aided Design (BRL–CAD) is a solid geometry modeling CAD system that assists the users in constructing and analyzing complex structures that would otherwise require advanced computations.[1,2] BRL–CAD uses a tool called the ray-tracer[3] to help analyze solid objects.[4,5] This tool has the ability to quickly render graphics and perform various operations pertinent when analyzing such geometries as BoTs (Bag of Triangles).[6] A BoT is generally used to represent triangle mesh objects in BRL–CAD. BoT objects have several modes, such as surface mode, volume mode, and plate mode. The mode that was primarily used throughout this project is the plate mode, which is used to represent a solid object in the form of a sheet with a small uniform thickness.

The goal of this project was to create all possible triangles of variegated designs, lengths, and angles within a specified finite range and then to perform tests to assess the functionality of the ray-tracer on these objects. These tests were used to gather information on whether the ray-tracer can hit points at various locations relative to each triangle, such as hitting directly on the triangle or just slightly exterior to the triangle. The ability to extensively test the BRL–CAD's ray-tracer on BoTs will be useful in identifying the baseline accuracy of the ray-tracer and then in identifying ways to improve its accuracy. Having a baseline test will eventually allow the BRL–CAD ray-tracer to yield comparative results that could be assessed relative to itself or to other ray-tracers (such as FASTGEN, a ray-tracer popular in many US Air Force applications).

2. Experiment/Calculations

This project was approached by modifying 2 example programs in the BRL–CAD source code. The first program, wdb_example.c, creates a database of BoTs by reading information about the triangles from a file. wdb_example.c was modified to create the database of BoTs for this project. The second program, rtexample.c, implements a sample ray-tracer by loading a triangle from a database and firing a single ray toward a point relative to the triangle and in a direction normal to the triangle. The rtexample.c was modified to fire rays at a multitude of points relative to each triangle in the database and at various tolerance levels.

The wdb_example.c was used as a starting point and modified to create a new program, (wdb_triangle.c) that would generate all possible BoTs. This process required several steps. Initially, wdb_triangle.c was modified to create a database of a single triangle using user specified coordinates of the 3 vertices input from the command line. This program was then expanded to create all possible triangles for a bounding box and a step size in the x, y, and z directions specified by the user. The BRL–CAD library function for creating BoTs, mk_bot, was

continually called during each iteration of the 9 nested for-loops in the main function to create a triangular object in the database for each possible combination of vertices. This resulted in multiple objects in the database that had the appearance shown in Fig. 1 when opened using the graphics interface application, MGED.

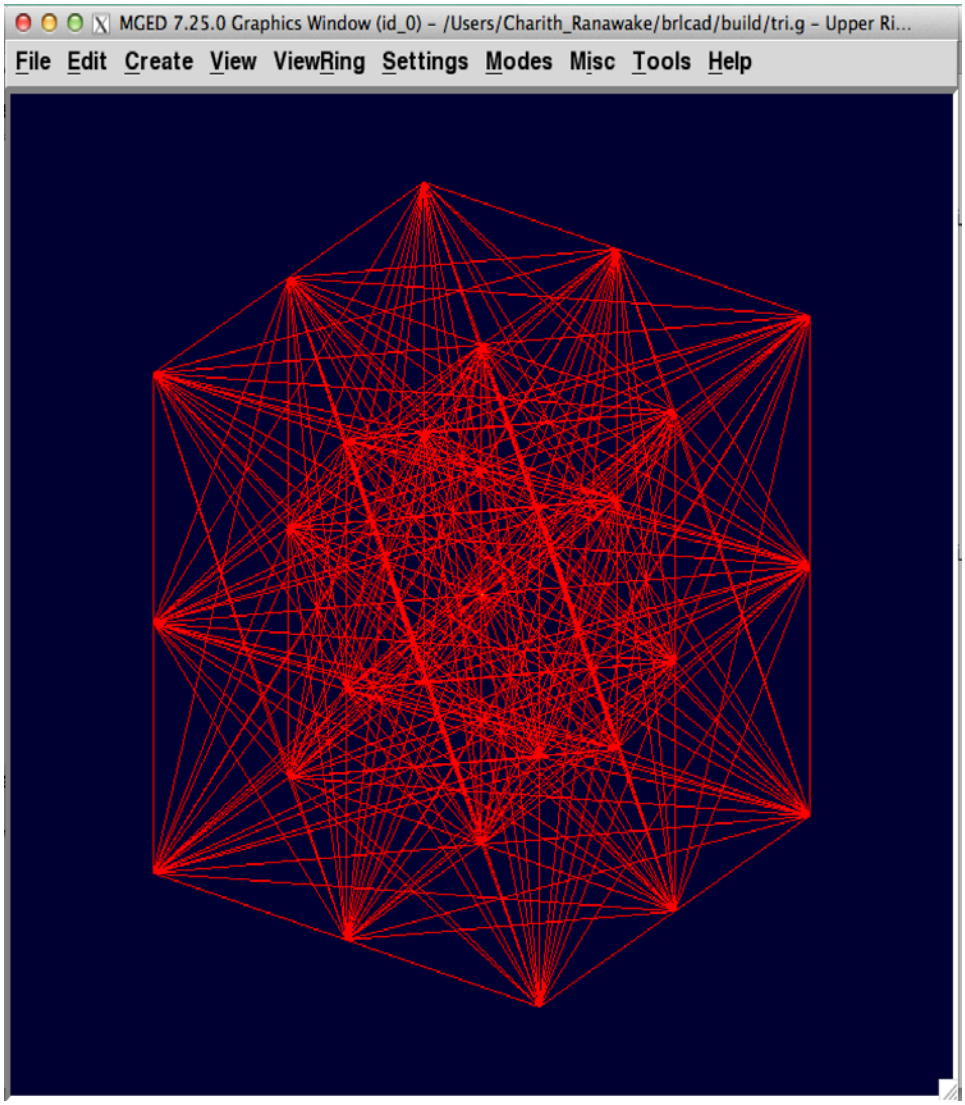

Fig. 1 All possible triangles within bounding box (1, 1, 1) with step size of 0.5

A ray-tracing program, rt_bot.c, was then implemented to fire rays at several locations, both on and off the triangle, for every triangular object in the database that was created by wdb_triangle.c. These locations are shown in Fig. 2.

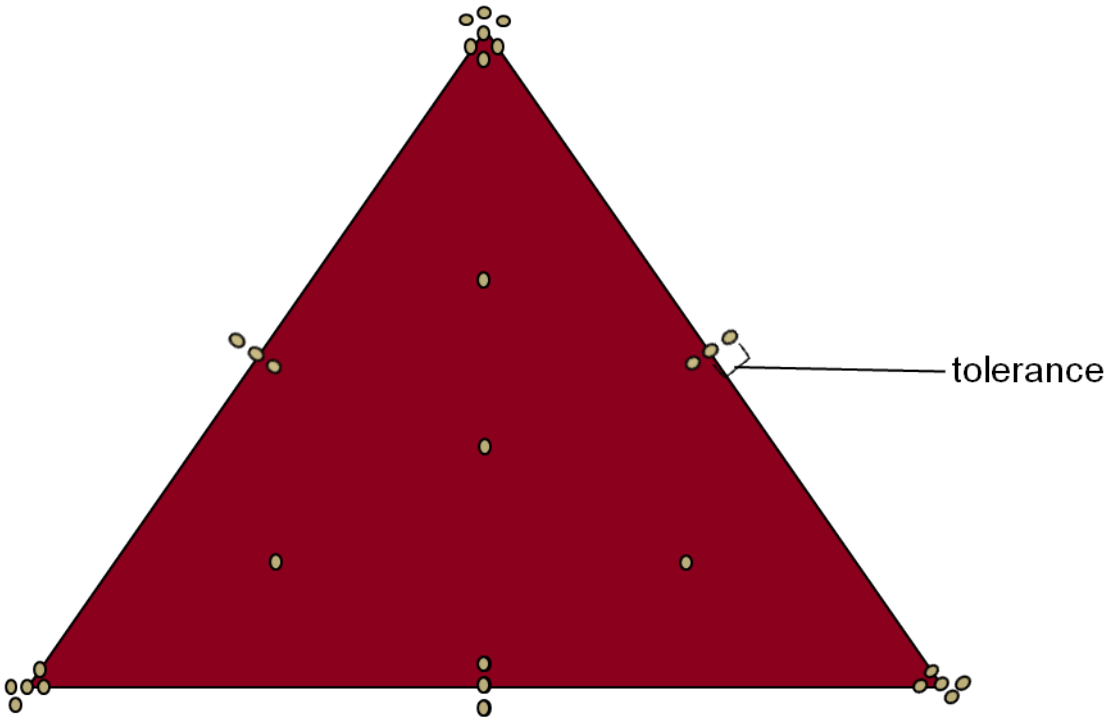

Fig. 2 Rays fired at triangle

Figure 3 shows the basic outline of the ray-tracing program, rt_bot.c. The main program has the same basic structure as the corresponding wdb_triangle.c program that created the BoTs. The user inputs the bounding box and the step size of the program as runtime arguments. The program uses 9 nested for-loops for incrementing vertex coordinates. For each combination of vertices that forms a triangle, wdb_triangle.c loads the corresponding object from the BoT database and calls a function for performing the ray-tracing on the triangle. As mentioned before, ray-tracing involves calculating the coordinates of the target point, the unit normal to the triangle, and the coordinates of the firing point. The ray-tracing function also maintains counters to hold such information as the number of rays fired and the number of successes (hits or misses). For interior target points, vertices, and edges, a success occurs when the ray-tracer records a hit. For an exterior point, a success occurs when the ray-tracer records a miss.

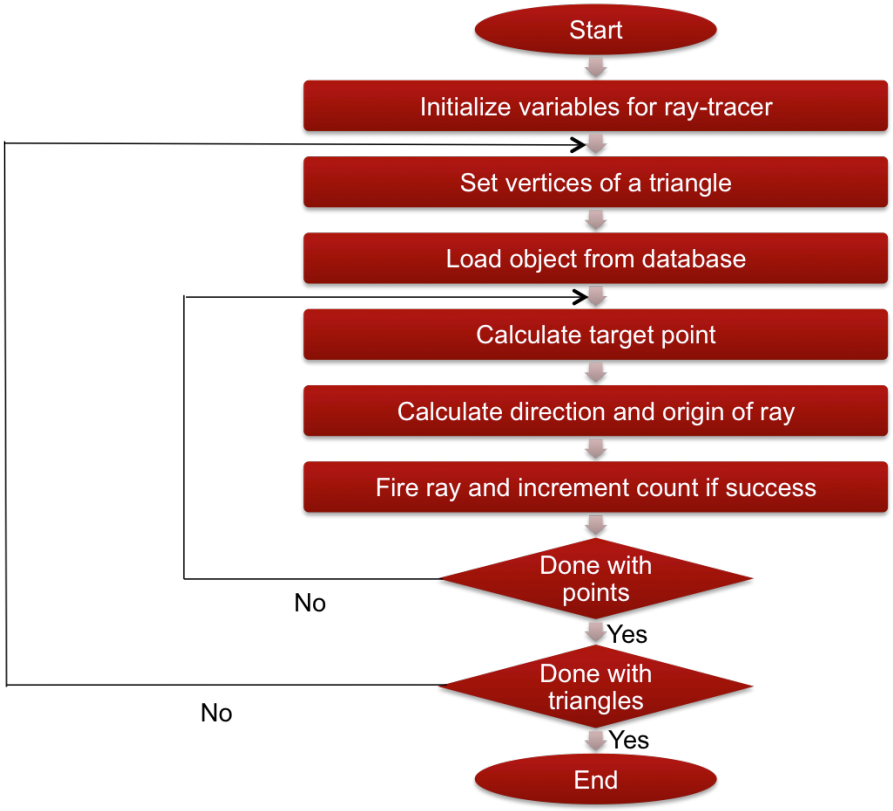

Fig. 3 Ray-tracing program flowchart

Several steps are involved when determining the x, y, and z coordinates for a target point. Figure 4 shows an example in which the target point is interior to the triangle and on the median joining vertex B to mid-point of line AC. The first step is to calculate the midpoint, D of line BC, as the target-point coordinates:

$$x_4 = \frac{(x_1 + x_3)}{2}, y_4 = \frac{(y_1 + y_3)}{2}, z_4 = \frac{(z_1 + z_3)}{2} \quad . \tag{1}$$

Then the distance BD, which we will call "L" the distance to the target point, is calculated from using the formula below:

$$L = \sqrt{(x_2 - x_4)^2 + (y_2 - y_4)^2 + (z_2 - z_4)^2} \quad . \tag{2}$$

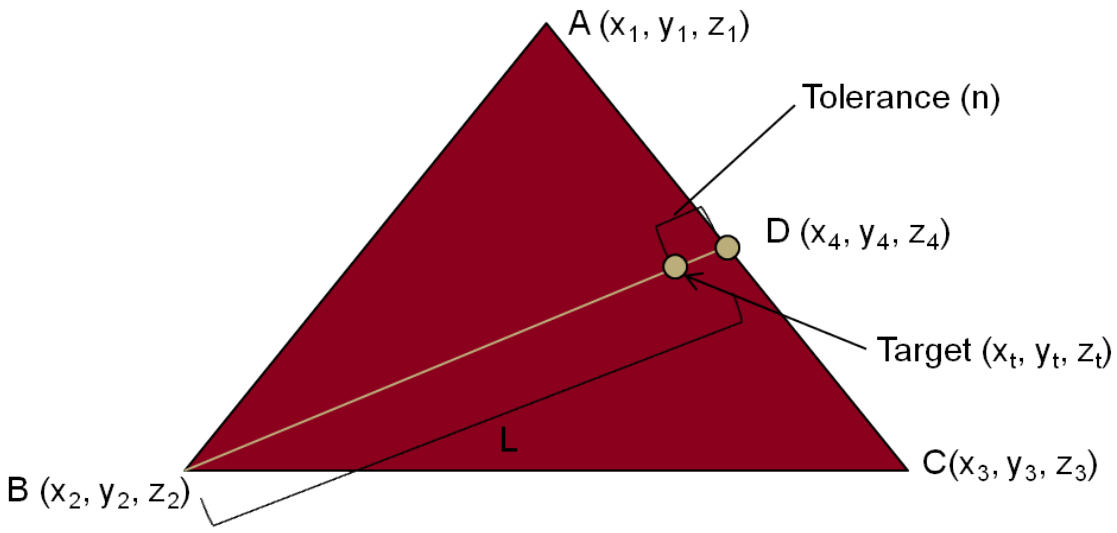

Fig. 4 Target-point diagram

The target point is calculated using Fig. 5 by applying the following proportion formula:

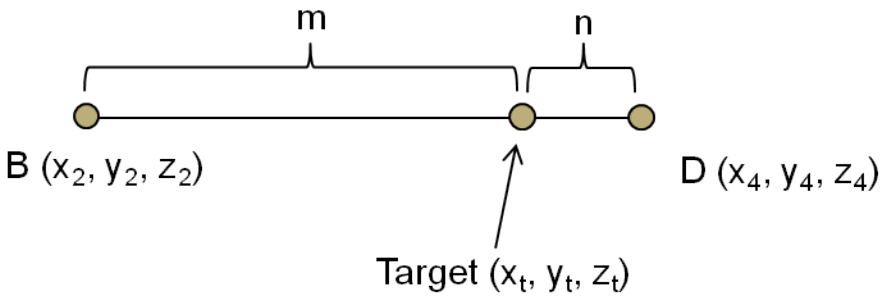

Fig. 5 Coordinates of a point dividing a line segment

Where the target-point coordinates (x_t, y_t, z_t) are calculated by Eq. 3 using the proportional distances *m* and *n* relative to L, where $m = L - n$, and $n = tolerance$

$$x_t = \frac{mx_4 + nx_2}{m+n}$$
$$y_t = \frac{my_4 + ny_2}{m+n} \quad . \tag{3}$$
$$z_t = \frac{mz_4 + nz_2}{m+n}$$

These calculations are specifically for points that are interior to the triangle. The exterior points can be calculated in a similar manner using a slightly different formula.

In addition to the target point, the firing direction and the firing point were calculated. Figure 6 shows the case where a ray is fired from the firing point (x_f, y_f, z_f) in a direction normal to the triangle.

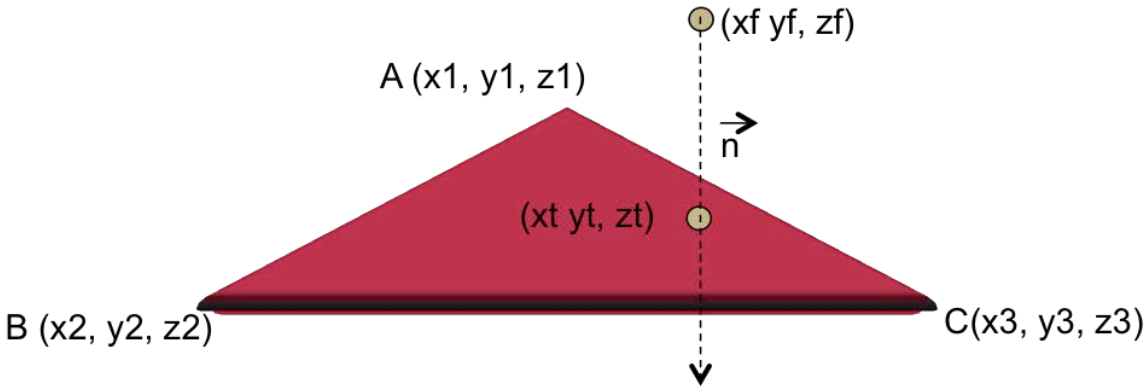

Fig. 6 Triangle with normal vector

Using the coordinates of the target point, the firing point and the direction of the ray were calculated as follows:

- Step 1: Calculate normal to ΔABC using the cross product of 2 vectors (Ex: AB x AC).
- Step 2: Calculate the unit vector $n = < n_x, n_y, n_z >$ of the cross product.
- Step 3: Determine the coordinates of the firing point (x_f, y_f, z_f) as the following:

$$x_f = x_t - hn_x$$
$$y_f = y_t - hn_y$$
$$z_f = z_t - hn_z \quad . \tag{4}$$

3. Results and Discussion

To test the performance of the ray-tracer, a BoT was created for the bounding box (2, 2, 2) with the step size of 1.5. This created a database of 336 triangles. Then the ray-tracer program was executed on the set of triangular objects; the number of successes and the number of attempts were recorded for each type of target point. The data are tabulated in Table 1. Figure 7 is a bar chart showing the variation of percentage of successes with the type of target point. The data show that the ray-tracer recorded 100% success for interior points as expected. Also, as the

tolerance increased from 0.00001 to 0.1, the percentage of successes increased from 91% to 95%. The ray-tracer recorded the smallest percentage of successes when the target point was on a vertex or edge. From these results, it can be concluded that BRL–CAD's ray-tracer is fairly accurate for target points that are interior to the triangle as well as target points that are small tolerance levels away from the edges and vertices. However, the accuracy decreased as the rays were fired at a target point that was on a vertex or edge.

Table Data for the ray-trace of BoTs for the bounding box (2, 2, 2) with step size 1.5

Database: Triangles (336)			
Type	**Successes**	**Attempts**	**Percentage (%)**
Interior	1344	1344	100
Vertex/Edge	1710	2016	84.82142857
Tolerance - 0.1	5802	6048	95.93253968
Tolerance - 0.001	5772	6048	95.43650794
Tolerance - 0.00001	5528	6048	91.4021164

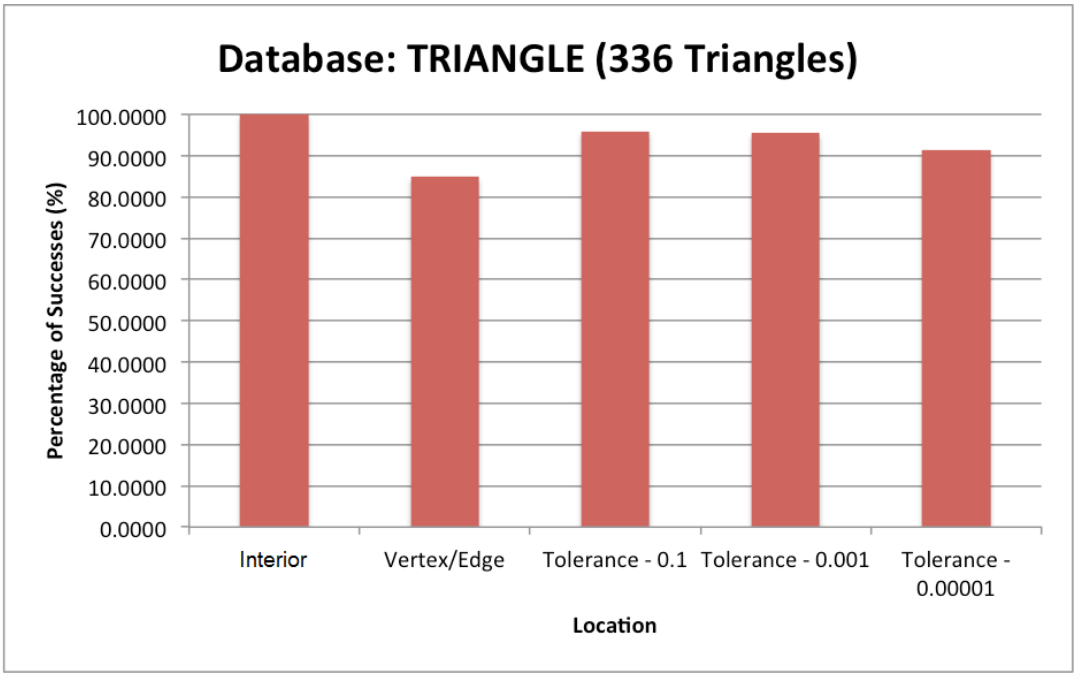

Fig. 7 Data for the percentage of successes for the bounding box (2, 2, 2) with step size 1.5

4. Summary

In this project, BRL–CAD's ray-tracer was tested on the primitive Bag of Triangles (BoTs). Two programs were created to accomplish these tests. The first program created numerous triangles or objects in the database, while the second program was used to fire rays at each object. Based on the results, the ray-tracer was very accurate on points on the interior of the triangles and small tolerances away from the vertices and edges. But the accuracy of the ray-tracer decreased significantly when tested on edge or vertex points.

5. References and Notes

1. About BRL-CAD. BRL-CAD [accessed 2015 Jan 13]. http://brlcad.org/d/about.
2. Feature Overview. Contributors' Guide to BRL-CAD [accessed 2014 Aug 22]. http://en.flossmanuals.net/contributors-guide-to-brl-cad/feature-overview/.
3. There actually are several ray-tracing algorithms available to select within BRL–CAD. This report describes fitness testing that is applicable for any of these ray-tracer algorithms. Therefore, for purposes of this report they are singularly referred to as "the ray-tracer".
4. NMG raytracing performance improvement. BRL-CAD [accessed 2015 Jan 13]. http://brlcad.org/wiki/NMG_Raytracing_Performance_Improvement.
5. Ray tracing (graphics). Wikipedia, the free encyclopedia [accessed 2015 Jan 13]. http://en.wikipedia.org/wiki/Ray_tracing_(graphics).
6. BoT. BRL-CAD [accessed 2015 Jan 13]. http://brlcad.org/wiki/BoT.

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

2 (PDF) DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT

1 (PDF) GOVT PRINTG OFC A MALHOTRA

6 (PDF) DIRECTOR US ARMY RSRCH LAB RDRL SLB S M PERRY W BOWMAN S MORRISON N REED G SAUERBORN C YAPP