

Computational Geometry and Synthetic Environments^{*}

P. O. Bobbie

**Advanced Distributed Simulation Research Consortium
313 Benjamin Banneker Technical Building A
Florida Agricultural and Mechanical University, Tallahassee, Florida
<http://adsrc.famu.edu>**

Abstract:

The primary purpose of the ADSRC effort is to develop a military wargaming environment that uses the latest tools in computerized visualization and network technology. We are primarily concerned with the graphic and behavioral aspects of battlefield simulation process. To achieve the goals of the project, we have contributed in developing various tools and techniques for improving the current state of the art of battlefield simulation. Specifically we have focused on open standards, physical modeling, behavioral modeling, specialty software, and computational geometry. Of these, Computational Geometry forms the basis for the appearance and interaction of components in a three dimensional synthetic environment.

We have developed a learning tool called the Computational Geometry Tutorial, CGT, to aid in explaining several 2D and 3D graphic constructs. The CGT demonstrates some of the basic attributes/functions of 3D geometry. These functions provide an insight into the operations of the 3D graphics engine used in virtual world simulations. The CGT is not only a demonstration of mathematical simulation but also a teaching tool for the next generation of researchers. Based in HTML, VRML, and Java, the CGT is easily viewable by anyone, anywhere using a compliant client browser.

1. Introduction:

Synthetic environments require realistic physical and behavioral modeling. Today's virtual worlds can provide conformance to three of the five senses. Visual, aural, and tactile data can be relayed to the participant. In our studies, we have focused on accurate visual and aural data in physical and behavioral models.

The most striking constructs in synthetic environments are accuracy of scale, shape, and markings when compared to real life environments. For example a model of a jeep can be made to appear just like the real thing through the use of detailed CAD models, photorealistic rendering, and texture mapping. Also, the behavior of the jeep can be modeled in such a manner making it indistinguishable to the real world jeep in the context of the simulation. The jeep must respond to the input of its driver. The driver will experience the conditions associated with driving the jeep. The wheels turn, there is engine and road noise, and the exhaust leaves a heat signature.

The behavior of synthetic objects is not limited to the interaction of vehicle type objects

^{*}This work has been supported by a grant from ARO (ADSRC DAAH04-95-10250)

within the environment. The environment itself must react to its inhabitants to provide a convincing illusion of reality. In the example of the jeep, the vehicle affects the environment in which it resides, thus changing the attributes of the environment. As the operator drives the jeep, the tires should kick dust, the ambient temperature should change, and the jeep should leave a trail [6].

Perhaps the most interesting environmental behaviors are smoke, fire, and dynamic terrain shaping. Propagation of smoke and fire relies upon physics based modeling more heavily than the other physical behaviors previously mentioned. The seemingly random nature of these entities belies their true nature in realm of chaos theory. The computations necessary for accurate representation in a synthetic world are based in mathematical models of real world behaviors. The fire model is based upon a 3D extension of the Ising model; however, physics-based behavioral models are often extremely intensive and consume too much valuable processing time. Given the resolution of many synthetic environments, a realistic model is not always the most accurate one, often trading accuracy for speed and continuity. Many well known video games play upon this factor and exploit it successfully. The technique revolves around reducing polygon count to the bare minimum and taking up the slack with accurate texture maps.

2. The Tutorial Environment

The Computational Geometry Tutorial Project (herein referred to as CGT) began as a need for a web based introductory guide to the basic concepts of computational geometry and computer graphics. This need arose as a part of an ongoing Advanced Distributed Simulation Research Consortium (ADSRC) project. The tutorial is based in HTML, VRML, and Java. Thus, the CGT can be readily reviewed by anyone, anywhere using a compliant client web browser [5]. The current version of the CGT features Java applets demonstrating various geometric constructs and VRML demonstrating the capability of three dimensional (3D) graphics.

In a 3D world, there are three fundamental operations, translation, rotation, and scaling [3,4]. Translation, moving an object from one location to another, is accomplished by the following matrix:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -tx & 1 & 0 & 0 \\ -ty & 0 & 1 & 0 \\ -tz & 0 & 0 & 1 \end{bmatrix}$$

The matrix for scaling a vector by S, is as follows:

$$\mathbf{S} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 1 shows an example of 3D Translation and Scaling.

Rotating an object about an axis is the same as partial rotation about the rotation vector's component axes. For a rotation of j degrees about the x -axis, the required matrix is:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(j) & \sin(j) & 0 \\ 0 & -\sin(j) & \cos(j) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

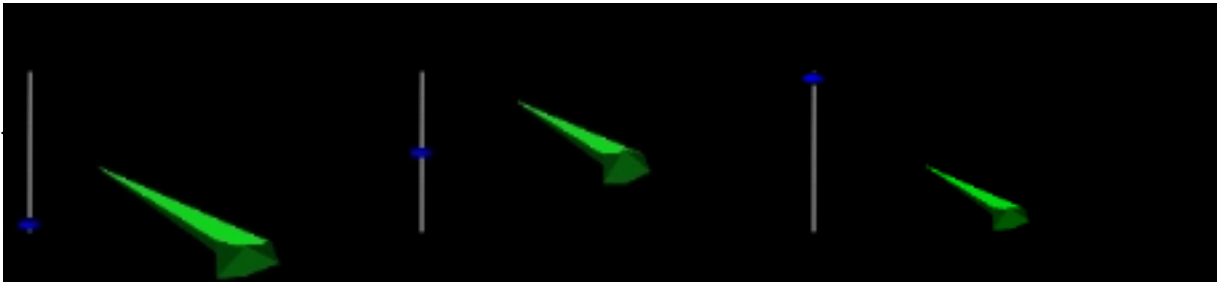


Figure 1. The 3D scaling example from the CGT starts with an object at close(left), away (center), and far (right). In addition, the scaling example shows translation as the object moves in a smooth sine wave pattern.

In addition to scaling, translation and rotation, the CGT looks at several low level graphic constructs and their applications within the simulation environment. Several fundamental problems exist in the simulation environment as well as in other applications. According to Leibling and Prodon [1], some of the problems include:

Intersection Problem: determine the intersection of families of simple objects including objects such as segments, polygons, or rectangles.

Location Problem: given a subdivision of a plane or space into cells, find the ones containing a given point or other objects.

Convex Hulls: find the smallest convex set enclosing a given family of points or other objects in a plane or space.

Proximity Problems: given a cluster of points, find a subdivision or such that each cell or the subdivision is made up of points in space that are closer to one point of the cluster than the others.

3. Implementation:

The current version of the CGT features Java applets demonstrating various geometric constructs, such as the Bezier Curve, B-Spline, Rational Bezier Surface, Voronoi Diagram, and the Convex Hull. The Voronoi Diagram addresses the proximity issue. A set of N Voronoi cells associated with N given points partitions the plane and is called a Voronoi Diagram. If $S = \{P_1, P_2, \dots, P_N\}$, the Voronoi Cell of point P_i is defined as the set of all points of the plane nearer to P_i than any other point of S . This cell is the intersection of the half-planes containing P_i and is delimited by the bisectors between P_i and the other points of S , thus forming a convex polygon. By joining all pairs of points whose Voronoi cells are

neighbors, the Delaunay Triangulation of the set S is obtained. The triangulations work by partitioning the convex hull of S into triangles having the points of S as vertices with none of the points in their interiors.

In the example C++ code below [2], the Edge class is used to realize the Voronoi Region Theorem. The program below is passed as an array, s , of n points, a point, p , and a bounding box, box . This results in the Voronoi region of p relative to point set, s .

```
Polygon *voronoiRegion(Point &p, Point s[ ], int n, Polygon &box)
{
  Edge *edges = new Edge[n];
  for (int i = 0; i < n; i++) {
    edges[i] = Edge(p, s[i]);
    edges[i].rot( );
  }
  Polygon *r = halfplaneintersect (edges, n, box);
  delete edges;
  return r;
}
```

Figure 2 shows a Voronoi Diagram and Delaunay Triangulations for a set of 13 points. The diagram is generated by a Java applet within the CGT. The applet uses a similar approach to obtaining the diagram and triangulation.

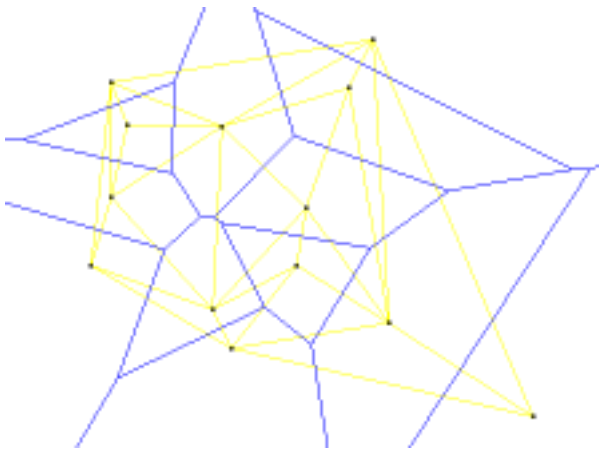


Figure 2. 13 point Voronoi Diagram and Delaunay Triangulation generated by a Java applet.

The VRML portion of the CGT contains user interactive 3D models that demonstrate 2D and 3D transforms. The three transforms covered are Rotation, Translation, and Scaling in 2D-space and 3D-space. The CGT also contains more interactive examples of 3D geometry and modeling in order to give the user a feel for some of the possible applications of computational geometry and computer graphics. The core of the VRML portion revolves around matrix operations. The primary difference between operations in 3D-space versus 2D-space is the extra dimension in the matrix itself, the mathematical basis for both, however is the same. The use of Java and VRML is ideal for our target group. We created the CGT for the purpose of educating students interested in learning about computer graphics in a single location. Prior to the CGT, most of the resources were widely dispersed among several sites with little or no written explanation describing the behavior of the algorithms. Code was obtained using various sources on the World Wide Web. Primary development used a UNIX

based environment on SGI workstations running the Cosmo Software Suite and the Java Development Kit v1.1, it has achieved its purpose as an introductory guide to the basic concepts of computational geometry and computer graphics. The CGT runs on any Java enabled machine with a VRML 2.0 compliant browser. The preferred setup uses Netscape 3.01s and Cosmoplayer 1.0 for IRIX 5.3–6.4.

4. Conclusions

The CGT is in constant evolution. We are currently developing more algorithms and examples of behaviors, as well as writing more thorough explanations of the concepts. Some of the extensions planned for the CGT environment include integration with OpenGL, and interfacing with environments such as Magician. This will allow us to expound upon many more basic concepts, thus enlarging our target audience. In addition to simulating low level graphic functions, we are dissecting many other games and applications that use these features. The knowledge gained from this exercise will also help to educate our target group and at the same time demonstrate the real world capacity of Java and VRML, not just the educational aspects.

Acknowledgements: M. R. Arradondo, C. W. Birmingham

References

- [1] Liebling, T.M., Prodon, A. (1990). Scientific Visualization and Graphic Simulation, Algorithmic Geometry, Wiley.
- [2] Laszlo, M.J., (1996). Computational Geometry and Computer Graphics in C++, Prentice Hall, Inc.
- [3] Platinum Technology, Inc. (1997). VRCreator User Guide, Platinum Technology, Inc.
- [4] VRML Consortium (VRMLC), working groups and other information, (1998), available via <http://www.vrml.org>
- [5] Arradondo, M.R., Birmingham, C.W., Fontenot, A. L., (1997). Computational Geometry Tutorial, available via <http://dondo.ml.org/adsrc/>
- [6] Arradondo, M.R., "Behavioral Modeling and Computational Geometry," Advanced Distributed Simulation Research Consortium Annual Review, University of Houston Downtown, November 20, 1997 presentation slides available via <http://dondo.ml.org:1997/ann97.ps>