EXPLODING TECHNIQUES FOR CIG OBJECTS*

John Booker**, Michael Collery, Charles Csuri and David Zeltzer
Computer Graphics Research Group
The Ohio State University

ABSTRACT

This paper will present programming techniques and mathematical algorithms for producing animated sequences of exploding objects such as buildings and ship targets. The visual effect has obvious extension for application in simulation of combat conditions. Potential applications include simulation of weapons effects on real-time CIG visual systems.

*Supported by Navy contract N61339-8-0C-0008
**Naval Training Equipment Center N-74, Orlando, Florida

INTRODUCTION

Computer animation continues to be a valuable tool for the algorithmic generation of time-dependent phenomena. Techniques for the display of motion that have been developed over the past 15 years, combined with display algorithms for high resolution imagery, enables one to generate more realistic pictures and animation. These displays can add to our understanding of complex processes.

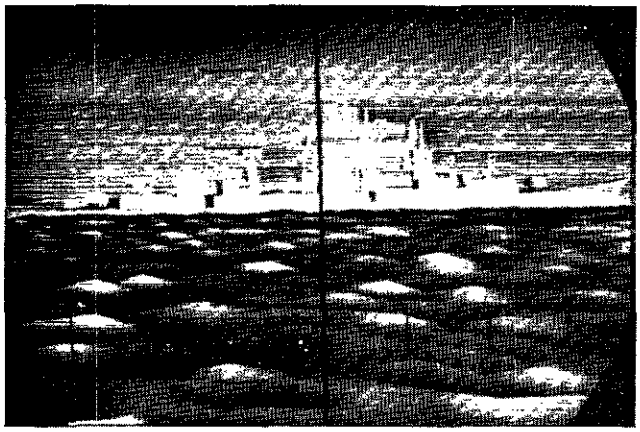
In our research we concentrated on developing tools for highly detailed data base generation. The emphasis has been upon interactive computer graphics techniques to work with a Defense Mapping Agency data base adding cultural features. We also generated at non-real time rates animation which simulated moving along a shoreline of Norfolk, Virginia. During the course of this effort we have considered the requirements for real-time display and the kinds of visual cues one might create to make the simulators more realistic. It occured to us, perhaps serendipidously, that we might be able to simulate explosions because of our data structures and display algorithm.

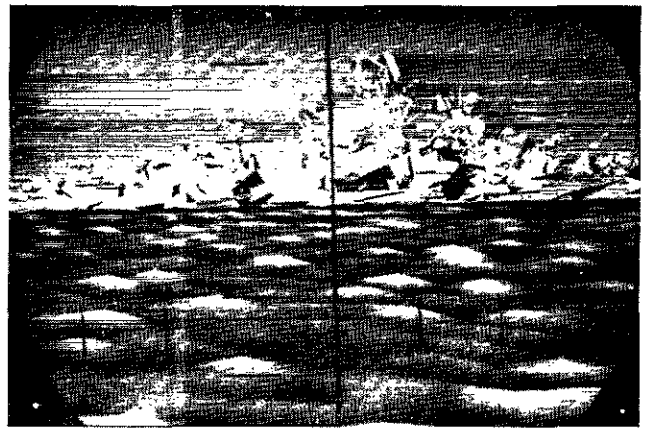
We have developed a program that we believe has applications for simulators and training. The realism of combat situations portrayed in a simulator with graphics capabilities could be enhanced if, along with maneuvers of the craft or vessels engaged, the destruction of targets were simulated as well. The program takes an object described as three-dimensional data and animates the individual polygons that define its shape. This technique has been used successfully to "explode" and "implode" objects and to alter the shape of objects. For example, the first implementation of the program was the explosion of a realistic model of a tank. It should be noted early in this paper that the goal of this program is not to create lifelike explosions. The explosion of a physical object is an extraordinarily complex phenomenon governed by many factors, including the nature of the explosive and the manner of its detonation, and the properties of the exploding object and its surroundings. These factors affect the size of the blast and the amount of smoke, flame, dust and debris produced. For our purposes we are not interested in faithfully reproducing the processes that constitute such an event. Rather, we wish to produce a visually interesting and unambiguous suggestion of a detonation.

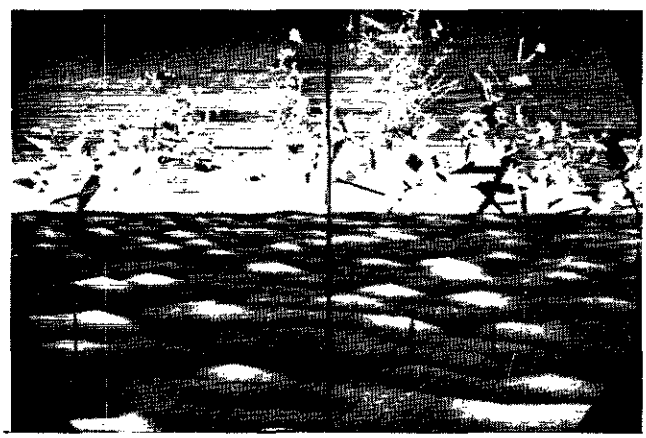
The program is general enough to allow for numerous applications. Buildings, bridges, towers and structures of any kind can be detonated providing simulations for training and education. It can even be used to depict the explosion of a cataract in the simulation of a surgical procedure.

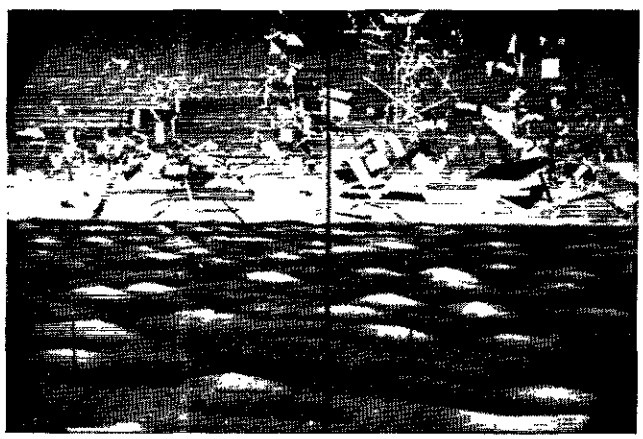
Computer animation deals with manipulation of three-dimensional models. These models are frequently represented as polygonal data. That is, the data is described as a series of coordinate points and a list of polygons that describes the relationships between these points. Traditionally in computer animation, once the object is described there is rarely a need to break it down into sub-parts. During animation, objects are transformed as a single unit to create the illusion of motion. For instance, a three dimensional model of an airplane can be made to fly through space by incrementally translating its position from frame to frame. Objects can be made to grow and shrink by the successive scaling of the coordinate information. These transformations are global, affecting all points and polygons uniformly. The premise of this paper is that since objects are made of component parts (polygons) there exists an inherent ability to manipulate each polygon individually as if they were objects in themselves.

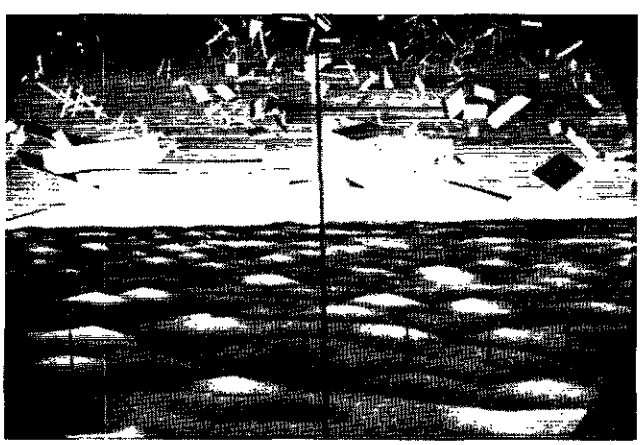
The algorithm to control linear explosions can be explained as follows. The first step is altering the description of the data so that polygons no longer share points with adjoining polygons, as is normally the case. This step increases the number of points in the data description without altering the shape of the object. Next a "movement vector" needs to be calculated for each polygon. This vector determines the direction of travel that a polygon will take during the course of an animation sequence. It is arrived at by subtracting the "center of explosion" from the centroid of each polygon. The "center of explosion" is determined by the animator and is based on the bounding box of the object.

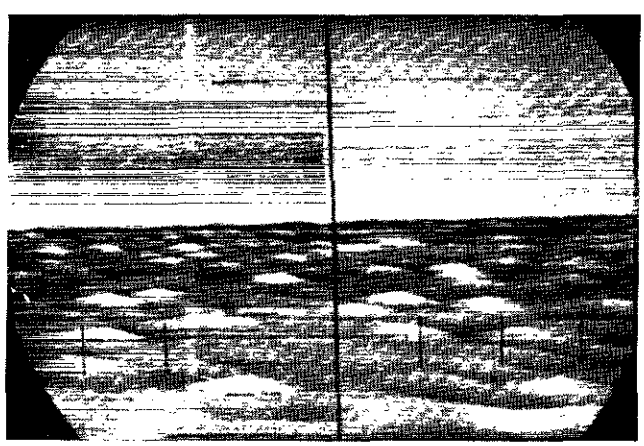












For instance, in the detonation of a tank the "center of explosion" was the minimum height of the bounding box and the center of the object in depth. This "center of explosion" caused the pieces to fly in equal directions in both the horizontal and depth planes, but only upward in the vertical plane, as if the tank was hit dead center by some explosive force.

After the object is redefined into unique polygons and a movement vector is calculated for each polygon the object is ready to be transformed by the main program. The movement that results from this program is based on parameters supplied by the animator. These parameters determine how much and which transformations will occur and their order of occurence. These parameters consist of a range of minimum and maximum values. They are computed internally by the program using a random number generator. It is our experience that this randomness gives the movement a more natural quality. It is important that the random number generator be given the same seed value for each new frame of animation to insure the movement continuity of each polygon.

Within the main program these steps occur. First, each individual polygon is translated to the origin. Then the rotations (if desired) are performed in the order specified by the animator. The polygons are then translated back to their original position. From this position a new location is calculated based on the polygon's "movement vector" and parameters supplied by the animator. It is at this point that functions such as acceleration and deceleration can be applied. These translations can occur equally in all three coordinate planes or a unique value can be calculated for each of the axes. For instance, if the effect of gravity is being simulated the direction of movement in the vertical plane should be positive at first and then gradually shift to a negative translation until the polygon's position is equal to the level of the ground plane.

A more precise mathematical model of an explosion can be developed into the present program if so desired. The first algorithm represented linear explosions. A second algorithm deals with parabolic explosions. Objects launched in a gravitational field at less than escape velocity will follow a parabolic trajectory. In two dimensions, and without regard to friction, the path of an object along such an arc is given by

$$x = t*v0*\cos(e) \qquad (1)$$

$$y = t*v0*\sin(e) - .5*G*t**2, \qquad (2)$$

where e is the angle of elevation above the x axis, G is the acceleration due to gravity (.98 meters per second), is an initial velocity, and t is a parameter. Since equations (1) and (2) give the position without regard to the previous location of the object, we can extend (1) and (2) to three dimensions by simply specifying an additional angle that gives a direction in addition to the elevation. That is, we can compute x and y in the x-y plane, and find the third coordinate by rotating in the x-z plane. This gives

$$x = t*v0*\cos(e)*\cos(a) \qquad (3)$$

$$z = t*v0*\cos(e)*\sin(a), \qquad (4)$$

where a is the azimuth angle, and y is found as in (2) above. This can be expressed conveniently in vector form:

$$x = t*v0x' \qquad (5)$$

$$y = t*v0*y' - .5*G*t**2 \qquad (6)$$

$$z = t*v0*z' \qquad (7)$$

where x', y', and z' are the components of a normalized vector giving an initial direction, v0 is an initial velocity, and t and G are devined as before.

The algorithm for parabolic explosions is analogous to that for the linear case.

```
for each object begin
   compute the center of explosion CE;

   for each polygon Pi begin
   compute the polygon centroid PCi;
   find the vector Vi from CE to PCi;
   compute v0i, the length of Vi;
   normalize Vi;
   end
end

for each frame
   for each polygon Pi begin
      compute xi, yi, and zi using eqns. (5)-(7)
      compute the change in xi, yi, zi since the
      last frame, call this vector Di;
      translate Pi by Di;
   end
```

While the velocity of the exploding polygons is constant using this algorithm, it is possible to scale this velocity in several ways. We initially assumed that the force of the explosion was constant in all directions. We could however, scale PVi based on say, the distance of the polygon from the centroid, which is just the magnitude of Vi. This would have the effect of giving polygons farther away from the center of explosion small initial velocity. Or we could scale PVi based on some measure of the angle between a polygon, the center of explosion, and some reference axis, and thus give a direction to the blast.

The effect of the algorithm is to translate each polygon along a parabolic trajectory radially outward from the center of the explosion. The center of explosion can be the centroid of the object or some other point chosen to achieve the desired effect. As in the linear case, the algorithm specifies only the position of the

polygon, and not its orientation, so rotation transformations must be explicitly applied if desired. We can take advantage of the fact that motion in the x-z plane is constant, only the vertical velocity of the object varies with t. Thus we can compute constant x and z increments once, and we need only compute the altitude each frame.

It is a simple matter to calculate when the height of any polygon reaches 0. If the initial height of the polygon is 0, the final value of t, say T, when the polygon again is 0 is given by

$$T = (2.0 * v0 * y') G. \qquad (8)$$

If the initial height of the polygon is some non-zero value, say h, then we need to increment t beyond the point at which the final height of the polygon reaches zero, since then the object will have returned only to its initial height h above the ground plane. This value of T can be determined from the equation

$$.5*G*t**2 - v0*y'*t + h = 0, \qquad (9)$$

solving for t with the quadratic formula. Both (8) and (9) give a unique value of T for each polygon, as each polygon will follow a different path depending on its initial direction vector and velocity. Since we can determine the time and position at which the polygon should impact, we can portray a splash, a cloud of dust, or a secondary explosion at the point of impact. Alternatively, we can use the direction and velocity of the polygon during preceding frames to compute a subsequent trajectory and thus cause the polygon to appear to bounce after it struck the ground.

An extension to these programs would be the ability to group polygons into units and then animate these units. Another extension which we implemented creates animation quite unlike explosions and implosions. The idea of manipulating polygons is the same. The difference is that the data structure is not altered into unique polygons. Instead the polygons are triangularized, while still sharing points with neighboring polygons. This step is performed to protect against nonplanar polygons which typical scan coverting algorithms do not accept. With this technique objects can be made irregular and new shapes can be formed. This technique was used to create the ocean like data seen in the accompanying photographs. Before modification this "ocean" was a flat plane. The "waves" were created by random rotation of constituent polygons.

SUMMARY

We have described techniques for the simulation of explosions of crafts or vehicles. In the paper we were not concerned about the precise physics involved in such a problem. Our concern was only with the visual effect. We view this effort as a preliminary effort and more work needs to be done for an implementation in a real-time flight simulator. It is important to mention that much of the realism associated with our explosions is due in part to our display algorithm. Further extensions of our techniques to real-time display would require a careful analysis of the relationships, trade offs and computational costs which may be necessary among various algorithms to achieve realistic explosions.

ABOUT THE AUTHORS

JOHN L. (JACK) BOOKER is acquisition director and principal investigator on tasks 8741 low level daytime CIG for trainers and 8743 area of interest CIG for NAVTRAEQUIPCEN computer sytems laboratory code N-74 Orlando, Florida. He served as project engineer for the aviation wide angle visual systems (AWAVS) CIG system procurement. He has been active in computer graphics since 1967 and has served as project engineer on a number of computer graphics systems and procurements including an Idiiom and E and S LDS-1. He received an M.S. Engineering degree from the University of Florida in 1967, BSEE from North Carolina State University in 1961 and an AB degree in Journalism from the University of North Carolina in 1953. Mr. Booker is a member of Tau Beta Pi, Eta Kappa Nu, Sigma Xi, IEEE, IEEE Computer Society, SIGGRAPH and the ACM.

MICHAEL COLLERY, computer animator/ researcher, The Ohio State University. B.F.A. Fine Arts, The Ohio State University. He developed data generation and animation techniques for the Norfolk Data Base project (Navy contract N61339-80-C-0008). He has also created numerous animation sequences of high resolution computer imagery for applications to commercial television, special effects for film and educational projects.

CHARLES A. CSURI, professor, Art Education, Computer and Information Science and Director of the Computer Graphics Research Group, The Ohio State University. Recipient of National Science Foundation support for the past 12 years. Received grants from the Air Force Office for Scientific Research, Bureau for the Education of the Handicapped and the Department of the Navy (NTEC-74). Member, editorial board of the IEEE Journal of Computer Graphics and Applications. He has published widely in the field of computer graphics and animation.

DAVID ZELTZER, B.Sc. Mathematics, Southern Oregon State College, 1978. M.Sc. in Computer Science, The Ohio State University, 1980. Currently Ph.D. candidate in Computer Science, O.S.U. Dissertation research in complex animation involving studies of robotics and artificial intelligence techniques for computer generated figure animation. Published papers in the Canadian - Man Machine Communication Conferences, 1980, Graphics - Interface 1982. IEEE Computer Graphics and Applications, Fall, 1982. Tutorial on 3-D computer animation, SIGGRAPH, 1982.