Animating Exploding Objects

Oleg Mazarak Claude Martins John Amanatides

Department of Computer Science
York University

Abstract

The paper explores the physically-based modeling of a blast wave impact on surrounding objects. We propose a connected voxel representation of objects to model explosions that result in realistic solid debris, rather than flat polygons. The paper also presents improved fracture algorithms capable of accounting for the damage of multiple explosions. The important implementation issues and the results of the simulation are discussed.

Key words: Explosions, blast waves, connected voxels, spring-mass particle model, rigid bodies, solid modeling, physically-based modeling

1 Introduction

Despite the growing interest in explosions, especially in visual effects, recent publications on the topic can hardly be found. Ever since the pioneering work of Reeves [Reeves83], who proposed using particle systems to model explosions, the area of explosion animation seems to have been abandoned by researchers. Particles are very effective at modeling certain after-effects of an explosion—fire and smoke—and creators of visual effects for commercial software have concentrated mainly on perfecting these two phenomena.

An important visual aspect of an explosion is the blast wave it generates. Although the blast wave itself is almost invisible, its influence on surrounding objects is easily discernible and results in multiple deformations, breakage and debris. Realistic simulation of the blast wave impact and debris formation can enhance the visual perception of the explosion by a viewer. Unfortunately, there is currently no tool on the market for modeling realistic debris. Existing commercial solutions either create flat polygonal debris, which is no longer satisfactory for the growing demands of graphic artists, or force users to design debris templates explicitly, based on their best understanding of the explosion phenomenon.

This paper is intended to fill the gap and generate interest inside the graphics community in the physically-based modeling of explosions, by presenting a model for the physically accurate simulation of debris formation and motion. In this work, however, we discuss only the elementary blast wave equations, which are used in our simulation to provide a well-balanced trade-off between physical accuracy and simulation speed. The full scientific simulation of the real physical processes associated with explosions would be unnecessarily complex and too computationally expensive for the field of computer graphics, which is focused mainly on the visualization of the blast wave impact on surrounding objects.

We introduce a connected voxel representation of objects, which allows the creation of realistic volumetric debris and eliminates certain disadvantages of the spring-mass particle model that arise when attempting to model rigid bodies. We also show how to incorporate our blast wave model with a rigid body motion simulator to produce realistic animation of flying debris.

2 Theory of Blast Waves

Any rapid release of energy in air, such as an explosion, generates a *blast wave*. The front of this blast wave exhibits a nearly discontinuous increase in pressure, density, and temperature, and is called a *shock front*.

The transmission of real blast waves in air is an inherently nonlinear process involving nonlinear equations of motion. In this sense, a blast wave differs quite significantly from an acoustic wave, which involves only infinitesimal pressure changes and moves at sonic velocity.

An ideal blast wave, which can be used as a sufficient approximation, is generated under the assumptions that the medium—air—is still and homogenous, and that the source is spherically symmetric. The shock front of an ideal blast wave is perfectly spherical, and therefore the characteristics of the blast wave are functions only of distance R from the center of the source, and the simulation time t [Baker73].

2.1 Blast Wave Pressure Profile

Since the blast wave's impact on surrounding objects is influenced primarily by pressure, we are mainly interested in simulating the pressure changes across the shock front.

The pressure profile generated by an ideal blast wave at some location removed from the center of explosion is shown in Figure 2-1. Before the shock front

reaches the given point, the pressure is equal to the *ambient pressure* p_0. At arrival time t_a, the pressure rises discontinuously to the peak value of $p_0 + P_s^+$. The quantity P_s^+ is called the *peak overpressure*. The pressure then decays to ambient in total time $t_a + T^+$ (*positive phase*), drops to a partial vacuum of value $p_0 - P_s^-$ (*negative phase*), and eventually returns to the ambient pressure p_0 in total time $t_a + T^+ + T^-$.

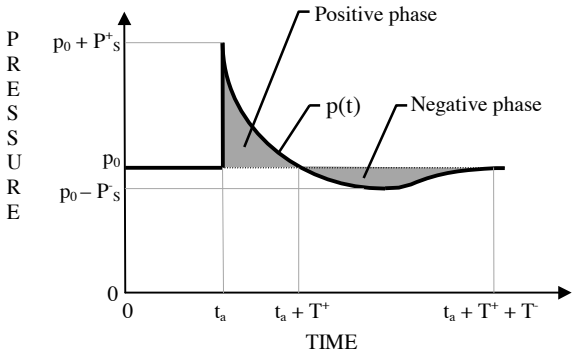

Figure 2-1. Pressure-time curve of an ideal blast

The pressure profile curve can be accurately described by the modified Friedlander equation [Baker73]:

$$p(t) = p_0 + P_s^+ (1 - t / T^+) \; e^{-bt/T^+} \qquad (2.1)$$

where time t is measured from time of arrival t_a. The blast wave parameters P_s^+, t_a, T^+, and b allow freedom to customize the pressure profile curve and the initial decay rate for different explosions, at different distances from the source.

Given an ideal point source explosion, the peak overpressure P_s^+ initially decays as the inverse of the cube of the radius $1/R^3$, and as the shock approaches the strength of a sound pulse, the decay approximates the inverse of the radius $1/R$.

2.2 Dynamic Fracture of Objects

The mechanism of body fracture and fragmentation under sharp blast wave loads is different from that under ordinary static loads. Blast wave propagation creates strong tensile stresses inside a solid object. Since most materials are substantially weaker under tension than under compression, the process of dynamic fracturing, called *spalling*, occurs. Spalling explains the effect of multiple, nearly uniformly distributed fracturing of solid bodies hit by a blast wave. Further propagation of the wave causes the fractures to grow, and the body is eventually fragmented into many parts. Due to the initial presence of multiple micro-fractures, called *flaws*, existing in most materials, spalling usually occurs for stresses lower than theoretical predictions [DGS96].

Davison and Stevens [Meyers94] proposed the concept of a continuum measure of spalling, based on reviewing and systematizing existing spall criteria. Davison and Stevens' theory is phenomenological in the sense that no detailed mechanisms for the initiation and propagation of fracture is incorporated. According to their compound-damage accumulation hypothesis, the number of cracks (damage D) inside a flawed microstructure grows exponentially:

$$D = BD^* [\exp(C \cdot t) - 1] \qquad (2.2)$$

$$\sigma = \frac{1}{2}(\sigma + |\sigma - \sigma_0|) \qquad (2.3)$$

where D^* represents the damage at total separation, or maximum possible damage, coefficients B and C are material-dependent constants, σ is the actual stress caused by the blast wave load, and σ_0 is a critical stress below which there is no damage.

3 Modeling

3.1 Blast Wave Model

The blast wave model employs the Friedlander equation (Equation 2.1) to compute physically accurate pressure changes at any distance R from the source of the explosion. The blast wave parameters required by (2.1) are obtained from experimental data for a reference explosion of one ton of TNT (trinitrotoluene) in a standard atmosphere [Kinney62]. The experimental data contains values for peak overpressure P_s^+, expected arrival time t_a, positive phase duration T^+, and the pressure decay coefficient b, measured at certain distances R_1, R_2, …, R_n, from the source. The corresponding parameters for any arbitrary distance R are obtained by interpolation.

Time and distance related parameters for explosions with different yields than one ton of TNT are derived from the reference explosion data by using an appropriate scaling factor as determined by the *scaling laws* [Kinney62]. This scaling factor is normally equal to the inverse of the cube root of the energy of the derived explosion.

In order to use our blast wave model in a dynamics simulator, the forces exerted on objects have to be derived from the pressure values. This process, however, is better understood after discussing object representation. The force computation is presented in detail in section 4.1.

3.2 Object Representation

Modeling exploding objects requires creating a structure that incorporates some connectivity information, in order to simulate the object's fracture and fragmentation. A good example of a connectivity-based representation is the spring-mass particle model, which was initially proposed by Terzopoulos [TPBF87]. A well-known problem with the spring-mass particle model is its poor ability to model rigid, inflexible objects. Increasing the spring stiffness coefficient to make the object more rigid leads to the "stiffness" of the governing ordinary differential equations (ODEs), which become difficult to solve. In such cases, the slightest imbalance in the system eventually causes the system to crash.

Another difficulty with the spring-mass particle model is the fact that it represents only the object's skeleton and requires the association of a surface to the particles before the modeled object can be rendered. This problem can be avoided when the simulation speed is not important, or when the number of objects in the scene does not change during the simulation and the surfaces can be computed during the initialization step. Simulating an explosion, however, entails the continual creation of new objects—debris—during the simulation. Computing surfaces to these multiple bodies dynamically can get unacceptably expensive for the simulation because the debris usually has a complex shape, and the accurate computation of the exact surfaces is non-trivial.

To overcome the difficulties described above, we propose modeling exploding bodies with connected voxels. The connected voxel model is a simple volumetric representation of objects, based on spatial occupancy enumeration [Foley90]. Connectivity information is preserved by the introduction of *links* between adjacent voxels. Links can be thought of as springs that are made infinitely stiff (in physical terms), and which do not allow any flexibility in the body, keeping adjacent voxels attached firmly together.

Figure 3-1. Connected voxels

Objects represented with connected voxels are animated in a different way than objects represented with the spring-mass particle model. In particle dynamics, each particle is considered an independent moving object, which does not make sense for connected voxels. Connected voxels are immovable with respect to each other and can be grouped into more complex structures, namely bodies. The world position of a particular voxel is computed from the position and orientation of the body to which that voxel belongs.

Voxel-represented objects can be displayed by associating a desired shape to voxels, for example, a cube. Rendering time is linear on the number of voxels; however, only boundary voxels are visible and require rendering. Since our representation already contains the adjacency information, the determination of visible surfaces is simply based on the observation that boundary voxels lack at least one neighbor out of six. The corresponding face is visible and has to be rendered.

Voxels, in principle, can be made as small as required in order to reduce the "staircased edges" problem. If simulation speed is not of primary concern, either the final image can be anti-aliased, or more sophisticated volume-rendering algorithms can be used [Kaufman91].

3.3 Simulating Fracture with Connected Voxels

The fracture of an object is simulated by "breaking" links in the connected voxels structure, which imitates the formation of cracks inside a solid. When the number of broken links grows, the object may be split into fragments. In order to compute the split fragments, the entire scene is represented with a connectivity graph. Its nodes are associated with voxels and its arcs with the links that connect voxels. The connected components of such a graph correspond to the independent bodies in the scene.

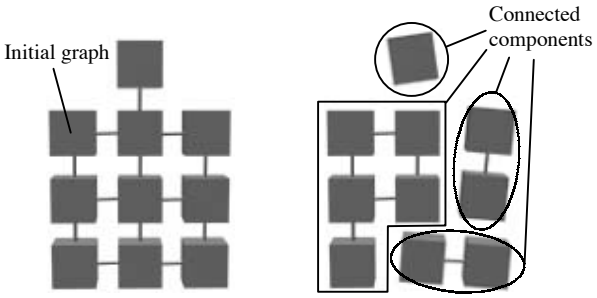

Figure 3-2. Graph representation of the connected voxel model

Although simulating fracture with connectivity-based models, in particular with spring-mass particle models, is not new to computer graphics ([TF88], [Norton91]), simulating fracture with connected voxels is different in the sense that using links requires finding new criteria for making a decision on when to break a link. We utilize the capabilities of our blast wave model to compute the pressure at the midpoint of each link. A link is broken whenever the pressure exceeds the link's *yield limit*, which is the maximum pressure that the link can withstand before breaking.

In our work, we have experimented with several fracture algorithms. These fracture algorithms produce different kinds of debris, and therefore the most appropriate one may be chosen for any particular case.

The simplest one relies on the variable link yield limit, which is computed by perturbing some random value, generated within a certain range, around some mean value [TF88]:

$$yield\_limit_1 = mean\_yield \pm \text{rand}(var\_yield) \quad (3.1)$$

Assigning a variable yield limit creates a portion of weak links, which naturally simulates the existence of flaws in the material structure.

In the improved algorithm, links that encounter the first explosion are weakened explicitly by reducing their yield limit, which makes them an easy target for subsequent explosions. This method simulates partial damage to the object, which does not necessarily result in link breakage.

A further improvement of the fracture algorithm accounts for the orientation of the links relative to the direction of the blast wave. In real explosions, the tensile forces created by the blast wave inside an object are much stronger in the direction parallel to the direction of the wave than normal to it. Based on this observation, links that are parallel to the direction of the wave are weakened by the orientation factor, computed as a dot product of the wave's radius vector and the link vector.

A different approach to simulating fracture is an algorithm that does not rely on the link yield limit, but rather on observed fracture patterns. The algorithm employs the compound-damage accumulation hypothesis of Davison and Stevens, discussed in section 2.2 (Equations 2.2 and 2.3). The incremental damage D is computed for each voxel to determine the number of links that have to be broken at the current simulation step in order to produce a realistic simulation of dynamic fracture. Here, D^* represents the maximum number of links that can be broken (six per voxel), and the parameter α is calculated as follows:

$$\alpha = \frac{1}{2}\left(p(t) - mean\_yield + \left|p(t) - mean\_yield\right|\right) \quad (3.2)$$

where $p(t)$ is a function that returns the pressure value (see section 3.1) at the center of a given voxel at time t.

The fractures inside a solid object are computed dynamically. At each time step, the fracture simulator is run in order to determine whether some links should be broken. If at least one link is broken at this stage, we run a breadth-first search to compute the fragments.

4 Implementation

4.1 Animation

A blast wave exerts forces on an object similar to those created by a very strong wind. Our model evaluates these forces at the centers of the voxels, as a product of the pressure generated by the wave and the area of the voxels projected onto the surface of the wave. Assuming a spherical blast wave, the force vector is congruent to the blast wave radius vector:

$$\vec{F} = p(t) \cdot A \cdot \frac{\vec{R}}{\left\|\vec{R}\right\|} \quad (4.1)$$

where $p(t)$ is a function that returns the pressure value (see section 3.1) at the center of a given voxel at time t, $\vec{R}$ is a radius vector from the source of the explosion to the voxel, and A is the projected surface area of the voxel. Simultaneously, we compute the torque on the body as a result of the force acting on a specific voxel of the body:

$$\vec{T} = \vec{r} \times \vec{F} \quad (4.2)$$

where $\vec{r}$ is the relative position of the voxel to the center of mass of the body. The force and torque vectors only need to be computed for boundary voxels, which approximate the surface of the body. The dynamic simulator then uses the summed up force and torque vectors to update the body position and orientation respectively.

In our model, we used the generic rigid body motion simulator described in [BW97]. We do not present the basic mechanics equations here. The reader can refer to [BW97], [Haug92], and [Shabana89] for a complete description of the rigid body dynamics. In order to solve the governing ODEs, a desirable integration method can be chosen. We opted for Euler's method, the simplest one, because we wanted to see how the model behaves in the basic case.

Using quaternions to represent rotation instead of rotation matrices is more advantageous, because the quaternion normalization step can be used to eliminate the rotation accumulation error, which may occur due to the integration process. More information on quaternions can be found in [Shoemake85] and [BW97].

The linear and angular momentum of the newly created body fragments are computed as weighted portions of the initial momentum of the body before it has been fragmented, which agrees with the law of conservation of momentum and ensures smooth motion of the split fragments.

4.2 Collision Detection

A discrete representation of the bodies using voxels is employed to simplify collision detection. In this work, we chose not to concentrate on efficient collision detection, and instead used a simple pair-wise test between body voxels. Collision between two bodies is considered to occur when the distance between their voxels becomes less than the expected limit. The distance between two voxels is defined as the length of the distance vector connecting the centers of the voxels.

To prevent inter-penetration of the bodies, the proper collision response is computed. The collision response algorithm changes the linear and angular momentum of the bodies discontinuously ([BW97], [MC95]), simulating the collision with the little to no deformation of the colliding objects specific to rigid bodies. The computation of collision response requires the determination of the collision region, which includes computing the point of collision and the vector normal to the body surfaces at this point. To avoid the time-consuming procedure of surface reconstruction, we assume that the surfaces of the bodies are smooth. We approximate the collision normal with the distance vector between the colliding voxel, and the collision point with the midpoint of the distance vector (Figure 4-1).

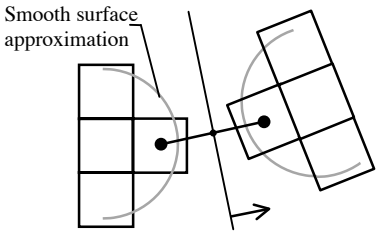

Figure 4-1. Collision approximation

Due to the extremely high velocities of the bodies hit by the blast wave, some bodies may interpenetrate noticeably in a single time step. This problem is solved by reducing the time step adaptively until the interpenetration is kept within a certain tolerance.

The voxel enumeration algorithm for collision detection can be optimized by limiting collision testing to only boundary voxels (see section 3.2). Further optimization of collision detection with *buckets* ([Norton91], [MC95]) reduces the total number of required tests significantly. In this approach, the scene is spatially subdivided into smaller regions called buckets, and each voxel is assigned the number of the bucket it is in at the current moment in time. Only voxels from the same or adjacent buckets need to be tested for collisions.

Unfortunately, optimization methods that rely on any kind of pre-computation step offer little to no benefits for explosion simulation because the number of bodies in the scene changes continually due to body fragmentation. Also, techniques that operate on polyhedral objects—such as I-collide [Cohen95] and V-Clip [Mirtich98]—are not directly applicable to our voxel-represented model without the additional step of associating polygon surfaces to the bodies, which can be expensive.

4.3 User Tools

There is an important trade-off between complete physical accuracy and increased user control over the simulation. It is frequently the case that some unrealistic simulation parameters can be chosen to produce more spectacular visual effects. In our implementation, the user can control the power of the explosion, as well as the model parameters, such as the mean and variable components of the link yield limit. Other controls include the base simulation time step, voxel mass, drag and gravity coefficients, etc.

5 Results

We have generated several video clips showing a blast wave impact on structures modeled with connected voxels. The visual results of the simulation are very similar to actual footage shot during nuclear bomb testing. Depending on the fracture algorithm used, we were able to generate various types of explosions. Even though we used a simple Euler's integration method, the simulation was both smooth and robust. Selected frames from the animations created using the connected voxel model are shown in Figures 5-1 and 5-2.

For simple models, the dynamic simulator is capable of generating frames over the real-time threshold. The bottleneck of the simulation is collision detection. Without collision detection, the computational cost of the dynamic fracture simulation process has an upper bound of the time required to compute connected components, which is $O(n)$, where n is the number of voxels in the scene. Collision detection, however, increases the simulation time up to $O(m^2)$, where m is the number of boundary voxels of all bodies in the scene. As the number of bodies grows due to fragmentation, so does the number of boundary voxels, and the simulation slows down. The use of buckets improves the frame rate by approximately a factor of $\log m$ over the non-optimized collision detection (see Table 5-1).

Number of boundary voxels	Frames/sec		
	No collision detection	Collision detection without buckets	Collision detection with buckets
1 – 10	> 100	> 100	> 100
10 – 100	70	20	15
100 - 1,000	15	0.25	5
1,000 - 10,000	< 1	< 0.017	< 0.33

Table 5-1. Frame rates for connected voxel representation. The experiments were performed on an Intel 333 MHz Pentium II, Windows NT 4.0 SP 4, FireGL 1000 Pro graphics accelerator.

6 Conclusions and Future Work

The paper has proposed a new model—connected voxels—for the realistic animation of a blast wave impact on solid objects. Visual realism of the animation is achieved due to both the volumetric representation of an exploding object, which allows the creation of convincing solid debris, and the use of physically-based methods to compute object fracture and motion. Improved fracture algorithms allow the simulation to take into account the damage of multiple explosions. The model can be successfully used in games, virtual reality applications, and visual effects, by choosing the appropriate level of accuracy in the object representation and the simulation methods.

Our current blast wave model leaves room for extensions and improvements, which would allow the simulation of more complex wave interactions with the environment. Possible refinements include the simulation of non-uniform pressure over the wave front due to shock absorbency of intervening objects, and modeling the compression of air in front of impacted objects. The implementation can be improved with more efficient collision detection and more sophisticated fracture simulation algorithms.

7 References

[Baker73] Baker, W.E., *Explosions in Air*, University of Texas Press, 1973.

[Baraff89] Baraff, D., "Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies", *SIGGRAPH 89*, pp. 223-232, 1989.

[BW97] Baraff, D., and Andrew Witkin, "Physically-based Modeling: Principles and Practice", Course Notes, *SIGGRAPH 97*, 1997.

[BW98] Baraff, D., and Andrew Witkin, "Large Steps in Cloth Simulation", *SIGGRAPH 98*, pp. 43-54, 1998.

[Cohen95] Cohen, J.D., M. C. Lin, D. Manocha, and M. K. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scaled environments", *Proceedings of the ACM Symposium on Interactive 3D Graphics*, pp. 189-196, 1995.

[DGS96] Davison, L., Dennis E. Grady, and Mohsen Shahinpoor, editors, *High Pressure Shock Compression of Solids II: Dynamic Fracture and Fragmentation*, Springer-Verlag, 1996.

[Foley90] Foley, J. D., Andries van Dam, Steven K. Feiner, John F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, pp. 549-550, 1990.

[Hahn88] Hahn, J. K., "Realistic Animation of Rigid Bodies", *SIGGRAPH 88*, pp. 299-308, 1988.

[Haug92] Haug, E. J., *Intermediate Dynamics*, Prentice Hall, 1992.

[Kaufman91] Kaufman, A., *Volume Visualization*, IEEE Computer Society Press, 1991.

[Kinney62] Kinney, G. F., *Explosive Shocks in Air*, The Macmillan Company, 1962.

[Lawn75] Lawn, B. R., Wilshaw T. R., *Fracture of Brittle Solids*, Cambridge University Press, 1975.

[MC95] Mirtich, B., Canny, J., "Impulse-based Simulation of Rigid Bodies", *1995 Symposium on Interactive 3D Graphics*, pp. 181-188, 1995.

[Meyers94] Meyers, M. A., *Dynamic Behavior of Materials*, John Wiley & Sons, 1994.

[Mirtich98] Mirtich, B., "V-Clip: Fast and Robust Polyhedral Collision Detection", *ACM Transactions in Graphics*, Vol. 17, No. 3, pp. 177-208, July 1998.

[MW88] Moore, M., and Jane Wilhelms, "Collision Detection and Response for Computer Animation", *SIGGRAPH 88*, pp. 289-298, 1988.

[Norton91] Norton, A., Greg Turk, Bob Bacon, John Gerth, Paula Sweeney, "Animation of Fracture by Physical Modeling", *Visual Computer*, pp. 7:210-219, 1991.

[PB88] Platt, J.C., and A.H. Barr, "Constraint Methods for Flexible Models", *SIGGRAPH 88*, 279-288, 1988.

[Reeves83] Reeves, W.T., "Particle Systems—A Technique for Modeling a Class of Fuzzy Objects", *SIGGRAPH 83*, pp. 359-376, 1983.

[Shabana89] Shabana A. A., *Dynamics of Multibody Systems*, John Wiley & Sons, 1989.

[Shoemake85] Shoemake, K., "Animating Rotation with Quaternion Curves", *SIGGRAPH 85*, pp. 245-254, 1985.

[TF88] Terzopoulos, D., and K. Fleischer, "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture", *SIGGRAPH 88*, pp. 269-278, 1988.

[TPBF87] Terzopoulos, D., John Platt, Alan Barr, and K. Fleischer, "Elastically Deformable Models", *SIGGRAPH 87*, pp. 205-214, 1987.

[WK88] Witkin A., and Michael Kass, "Spacetime Constraints", *SIGGRAPH 88*, pp. 159-168, 1988.

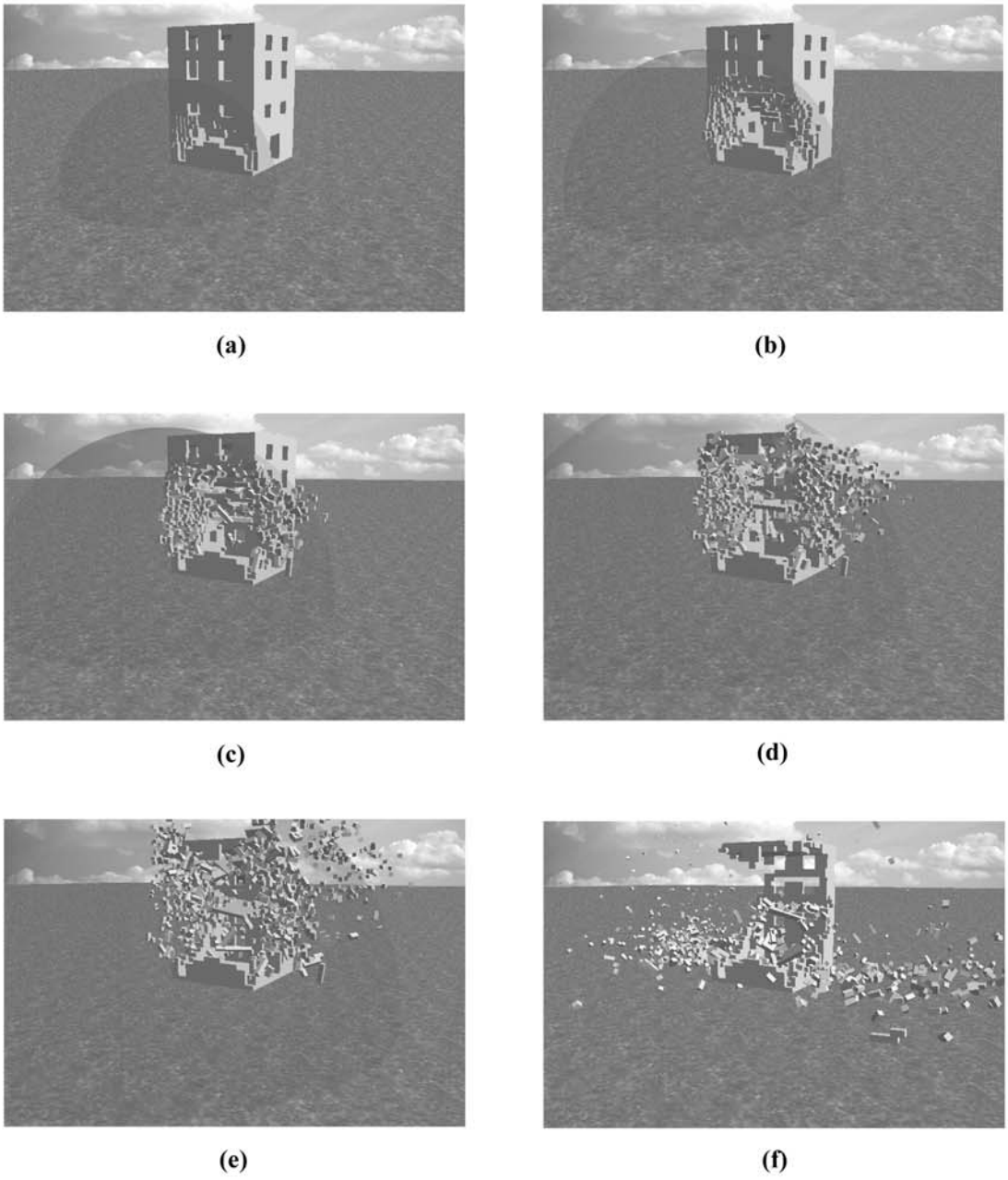

Fig. 5-1. Explosion of a rigid body represented by connected voxels

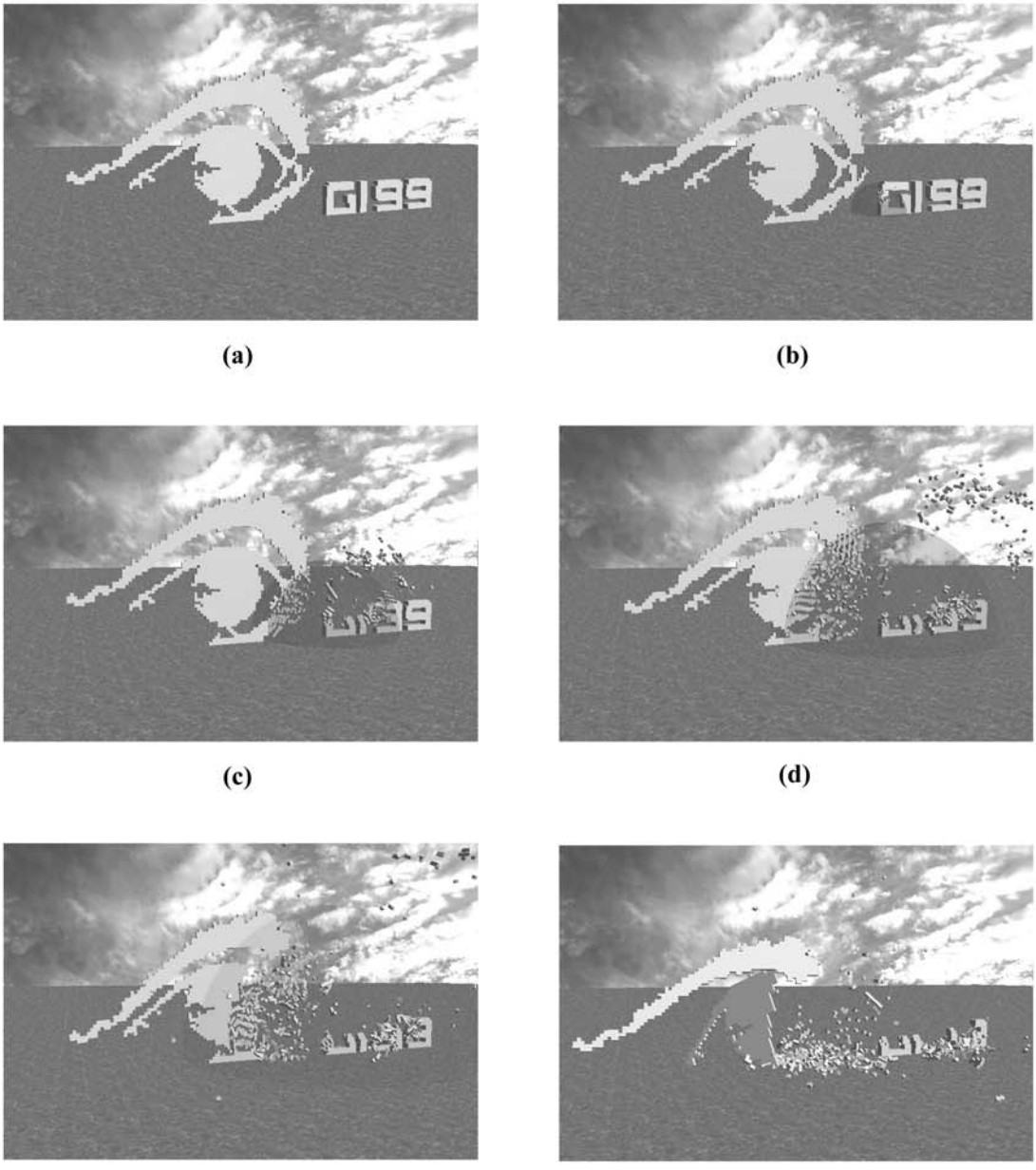

Fig. 5-2. Explosion of multiple rigid objects

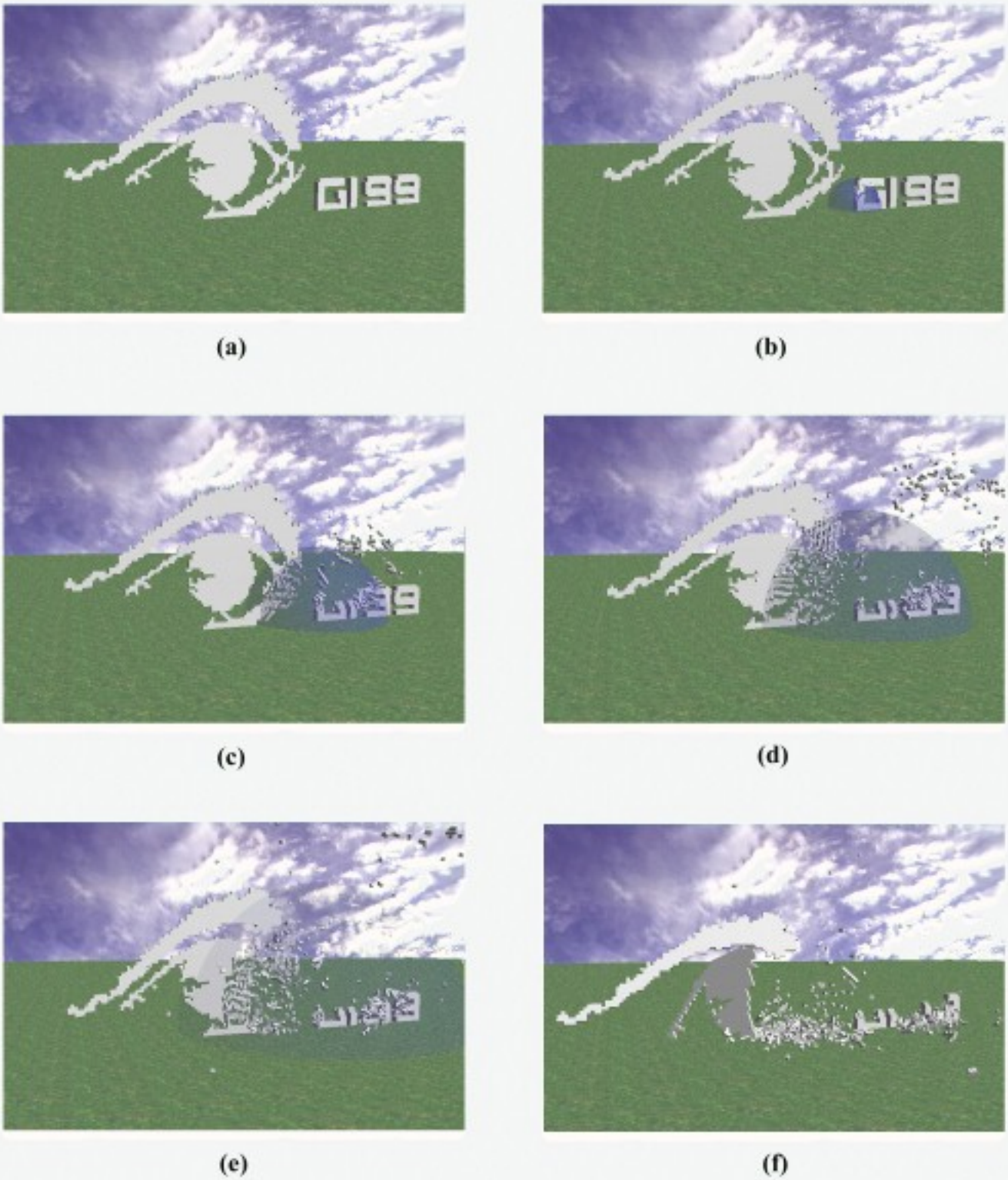

Fig. 5-2. Explosion of multiple rigid objects

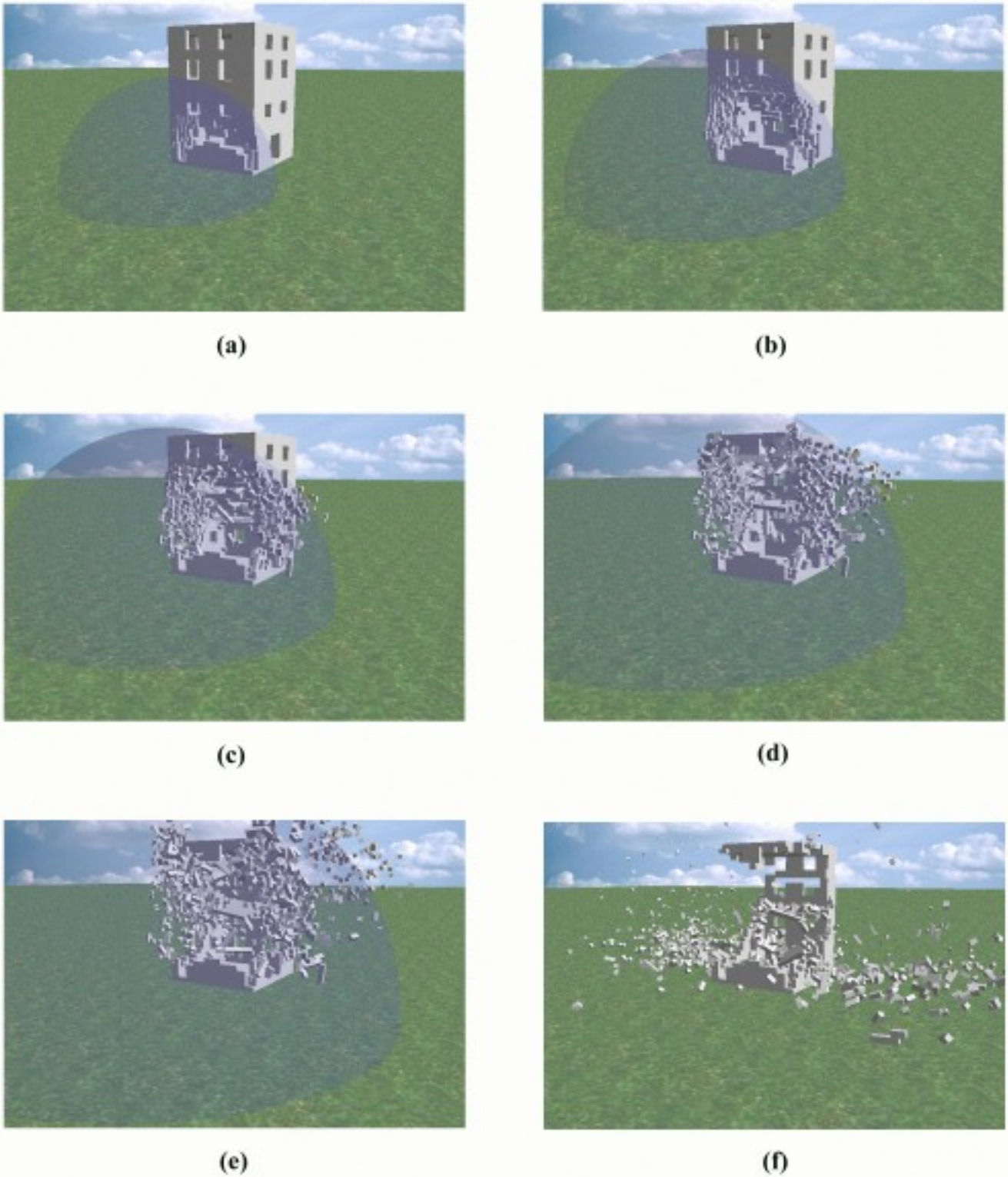

Fig. 5-1. Explosion of a rigid body represented by connected voxels