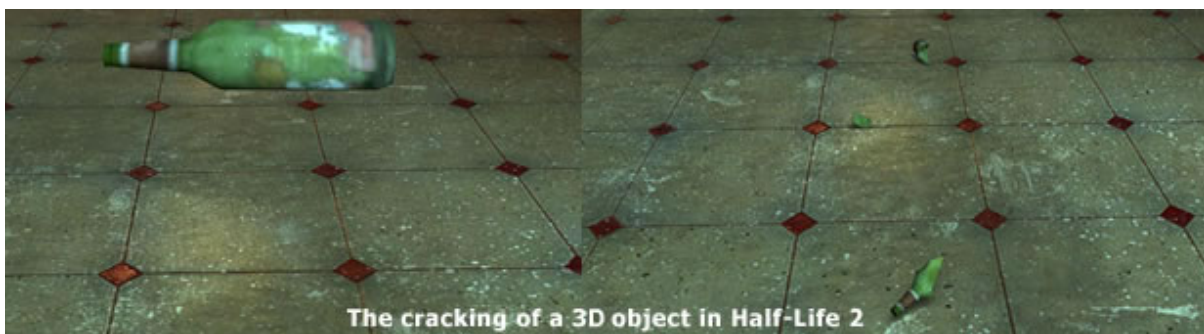


# A cracking algorithm for exploding objects

**Name:** Andrew Lowndes **Supervisor:** Dr Steve Maddock **Date:** 7th October 2008

## Introduction

Since the release of *Doom* <sup>[16]</sup> and *Star Wars* <sup>[17]</sup>, the video game and film industries has seen a rise in the need for special effects and better graphics; both vital for a realistic and immersive experience. The breaking of objects has been both a method for achieving that realism and a problem in 3D computer graphics. Various techniques have been used to simulate the cracking of objects. *Doom* featured 2D animated sprites to simulate the breaking of objects. More recently, *Half Life 2* <sup>[18]</sup> uses a predetermined cracked 3D object, which replaces the original object to simulate the cracking process. Extending this, *BlackSite Area 51* <sup>[19]</sup> features multiple sets of predetermined 3D objects to simulate cracking. All of these techniques add to the sense of realism in games.



After cracking an object multiple times you lose that sense of realism as you recognise the pattern in the cracking. There is a need to solve this and keep the player immersed in the game. A recent technology called DMM (Digital Molecular Matter) <sup>[20]</sup>, used in *Star Wars: The Force Unleashed* <sup>[21]</sup> already solves this problem by combining physics-based solutions and transforming the 3D object accordingly. Fractures are simulated in the DMM model by constructing the insides of 3D objects as tetrahedrons, applying transformations and splitting them apart. However, the tetrahedrons give a 'blocky' effect to the insides of the 3D object and, as the fractures are determined by the resolution of the inner tetrahedrons, the resulting crack is not as realistic as it could be.

The DMM model was built from work by O'Brien, who also developed techniques for fracturing a 3D polygonal object using tetrahedron-based lattices and physics, for his PhD thesis <sup>[7]</sup>. He later expanded the graphical modeling and animation of brittle fractures to ductile fractures <sup>[8]</sup>. O'Brien's PhD thesis was built upon the work by Norton et al, who worked on fracturing using a physics-based model <sup>[9]</sup>. This physical approach was later improved by work done in 2001 by Smith et al, who added constraints to the physics in favour of higher frame rates <sup>[11]</sup>. Similarly, Müller et al worked on deforming and fracturing stiff materials at real time speeds <sup>[13]</sup>. Different approaches to the cracking process include work done by Martinet et al, who used a procedural method for modeling cracks <sup>[12]</sup> and work by Raghavachary, who modeled fractures using Voronoi polygons <sup>[14]</sup>.

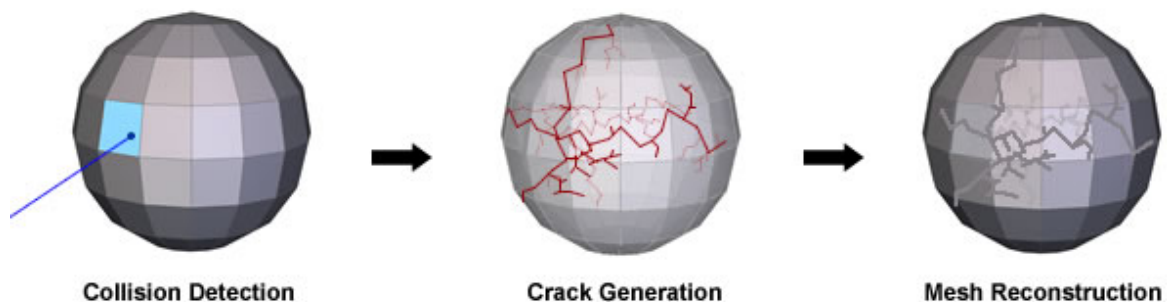
The purpose of this project is to provide a realistic-looking simulation of cracking a 3D object into pieces while running at real time speeds and achieving results that are competent with current working solutions.

## Analysis

There are three main stages to the cracking process. The first is detecting the collision between the projectile (a line) and the 3D object (triangles). The second is to determine the path the crack should take and generate it. The last step is to construct the inner mesh, which has been cut out of the 3D object. The animation of both the resulting pieces and the initial object during and after the cracking process is not covered in this algorithm. This means that the cracking algorithm can be executed separately from other physics being applied on the object.

The algorithm will be a procedural function that cracks the object at runtime. The object must be polygonal, be made up of triangles and contain no holes. The impact point will only be on one triangle of the object at any one time. Only one cracking process will run on a single object at one time. The input vector (the line), its magnitude and angle of impact will be used to determine the cracking path within the object. A depth map of the object generated upon impact (using ray casting) and the combined mechanical properties of the material (such as its toughness and porosity) will both be used to determine the crack path, allowing different materials to experience different crack patterns.

An object may not always crack; this depends on the material attributes. When cracking the object, the additional geometry used to fill the cracked pieces will be generated regardless of the crack going all the way through the object. As with all real time systems, there needs to be certain restrictions on the number of polygons visible on screen. Therefore, there will be a minimum size restriction on the cracked parts that are generated and a maximum limit on the number of polygons generated. Any polygons that exceed the maximum limit will be discarded.



This technique varies from others as it takes into account the shape of the 3D object, the angle and force of the impact and the material properties of the object, while working with the original geometry. The physical approach used in [9] constructs an inner mesh offline then manipulates that mesh at runtime. In contrast, the proposed technique constructs an inner mesh online. Although this will be slower than creating the mesh offline, the creation of the crack will be quicker than applying physics to the inner mesh and so should be quicker overall. Other procedural approaches such as that proposed in [12] utilizes predefined crack parameters for an object to generate a crack at runtime. The proposed technique varies from this as the crack parameters are generated at runtime and specific to that 3D object. Although this will be slower, it means that the crack can be generated based upon the impact location and angle and should provide more realistic results. The proposed technique is similar to [12] in that the resolution of the crack is independent of the resolution of the model. This means that the detail of the crack can be varied in favour of higher quality or frame rates.

The most difficult part of this project is constructing the inner mesh of the 3D object as it is being cracked and a number of problems may arise while developing this. The construction algorithm must not leave holes in the 3D object as otherwise the object would not be able to be cracked again. Another problem that may occur is that cracks may overlap and interfere with surface normals when recreating the inner polygons.

The future of this project could include integrating it into existing physics technologies such as the NVIDIA PhysX SDK <sup>[22]</sup>, in order for it to be accelerated on the GPU. It could also be made into a geometry shader and combined with existing vertex and fragment shaders to create fully-featured realistic materials. This would also accelerate the process as it would run on the GPU.

The result of this project will be a program demonstrating a range of simple polygonal objects in 3D space, which crack into pieces. The range of simple polygonal objects will include a sphere, a cube, a cone, a torus and a teapot. There may also be an option to import a standard .3DS object model. The material properties will consist of wood, metal, glass and marble. The materials will only affect the cracking pattern of the object and not its look or physical attributes.

To test the success of this project, both frame rate and visual quality (which will be assessed by an impartial adjudicator) will be compared with current solutions. Real life cracking of 3D objects will also be compared to see how realistic and accurate the results are.

## Plan of action

The project can be broken into small problems and researched individually. Below is a table of the action that will be taken to complete the project:

Week starting	Research	Action
13 <sup>th</sup> October	Crack patterns in real world objects	Research and analyse current techniques
20 <sup>th</sup> October	Material properties	Analyse findings and write up literature survey
27 <sup>th</sup> October	Required mathematics	Break down project into smaller chunks and write up requirements
3 <sup>rd</sup> November	OpenGL and C++	Start implementing project
10 <sup>th</sup> November	3D Objects	Evaluate progress and write up
17 <sup>th</sup> November	Animation	Finish draft and send to supervisor
24 <sup>th</sup> November	Collision detection	Continue implementing project
1 <sup>st</sup> December	Collision detection	Finalise Survey and analysis report
8 <sup>th</sup> December	Ray casting and tracing	Continue implementing project
15 <sup>th</sup> December	Ray casting and tracing	Review Survey and analysis report
7 <sup>th</sup> January - 3 <sup>rd</sup> February	---	Continue implementing project
4 <sup>th</sup> February - 2 <sup>nd</sup> March	---	Test and evaluate project
3 <sup>rd</sup> March - 30 <sup>th</sup> March	---	Write up report
31 <sup>st</sup> March - 6 <sup>th</sup> May	---	Finalise Final report

## Reading Material / References

- [1] Alan Watt, (2000), 3D Computer Graphics (3<sup>rd</sup> edition)
- [2] Eric Lengyel, (2004), Mathematics for 3D Game Programming & Computer Graphics (2<sup>nd</sup> edition)
- [3] Randima Fernando and Mark J Kilgard (2003), The CG Tutorial: The Definitive Guide to Programmable Real-Time Graphics
- [4] Daniel Fox, (2003), Demolishing Objects in Computer Games, The University of Sheffield
- [5] Adam Miller, (2004), A Cracking Algorithm for Exploding Objects, The University of Sheffield
- [6] Steven Workman, (2006), A Cracking Algorithm for Destructible 3D Objects, The University of Sheffield
- [7] James F O'Brien and Jessica K Hodgins (2000), Graphical Modeling and Animation of Brittle Fracture
- [8] James F O'Brien, Adam W. Bargteil and Jessica K Hodgins (2002), Graphical Modeling and Animation of Ductile Fracture
- [9] Alan Norton, Greg Turk, Bob Bacon, John Gerth and Paula Sweeney (1991), Animation of fracture by physical modeling
- [10] CNET Networks Inc, UC professor creates the Dark side, [http://findarticles.com/p/articles/mi\\_qn4176/is\\_20080704/ai\\_n28061532/pg\\_1](http://findarticles.com/p/articles/mi_qn4176/is_20080704/ai_n28061532/pg_1) (accessed 08/10/08)
- [11] Jeffrey Smith, Andrew Witkin and David Baraff (2001), Fast and Controllable Simulation of the Shattering of Brittle Objects
- [12] Aurélien Martinet, Eric Galin, Brett Desbenoit, Samir Akkouche (2004), Procedural Modeling of Cracks and Fractures
- [13] Matthias Müller, Leonard McMillan, Julie Dorsey, Robert Jagnow (2001), Real-Time Simulation of Deformation and Fracture of Stiff Materials
- [14] Saty Raghavachary (2002), Fracture generation on polygonal meshes using Voronoi polygons
- [15] Wikimedia Foundation Inc, Voronoi Diagram, [http://en.wikipedia.org/wiki/Voronoi\\_diagram](http://en.wikipedia.org/wiki/Voronoi_diagram) (accessed 08/10/08)
- [16] id Software, DOOM (1993) [Video game]
- [17] George Lucas, Star Wars IV: A New Hope (1997) [Film]
- [18] Valve Corporation, Half Life 2 (2004) [Video game]
- [19] Midway Studios Austin, BlackSite: Area 51 (2007) [Video game]
- [20] Pixelux Entertainment, DMM: How it works, [http://www.pixeluxentertainment.com/engine\\_tech.html](http://www.pixeluxentertainment.com/engine_tech.html) (accessed 08/10/08)
- [21] LucasArts, Star Wars: The Force Unleashed (2008) [Video game]
- [22] NVIDIA Corporation, NVIDIA PhysX Physics Simulation for Developers, <http://developer.nvidia.com/object/physx.html> (accessed 08/10/08)