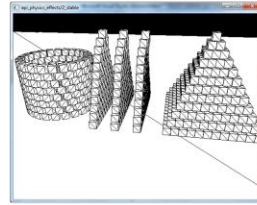
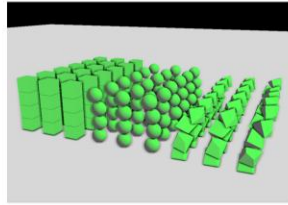
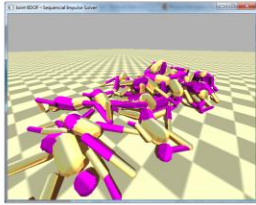


Destruction

Erwin Coumans, AMD
<http://bulletphysics.org>

Introduction



- Bullet Physics SDK, <http://bulletphysics.org>
- OpenCL/DirectX11 GPU physics research
- Sony Computer Entertainment Physics Effects



Hello, my name is Erwin Coumans.

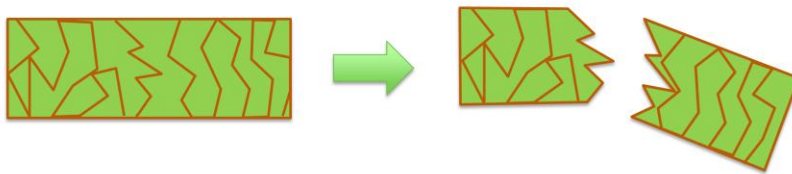
I'm creator of the open source Bullet physics engine, which is used in game and film production. I started Bullet while working at Sony Computer Entertainment America. I am now principal member of technical staff in the office of the CTO at Advanced Micro Devices (AMD) in California, and continue to work on Bullet. Several companies and developers contributed to this open source project.

In this presentation I will discuss some common methods related to real-time destruction in games and also film production. When possible we will refer to open source implementations.

- Geometry **preparation** and artist tools



- Runtime **destruction** methods



The destruction process can be divided in two parts:

- 1) the preparation of the 3D geometry, usually done during the creation of a game. This step pre-fractures a model into pieces that can break apart. Often this involves 3D content creation tools such as Autodesk Maya, 3ds Max or the open source Blender 3D modeler.
- 2) the breaking of objects when playing the game. This task is performed by the physics engine, as part of the game engine.

I will first discuss the geometry preparation and then the runtime destruction methods at the end.

Geometry Preparation



Geometry Preparation	Runtime Destruction
Voronoi shatter, slicing	Canned animation
Boolean operations	Real-time Booleans
Convex decomposition	FEM, particle based
Tetrahedralization	Rigid body & Hybrid



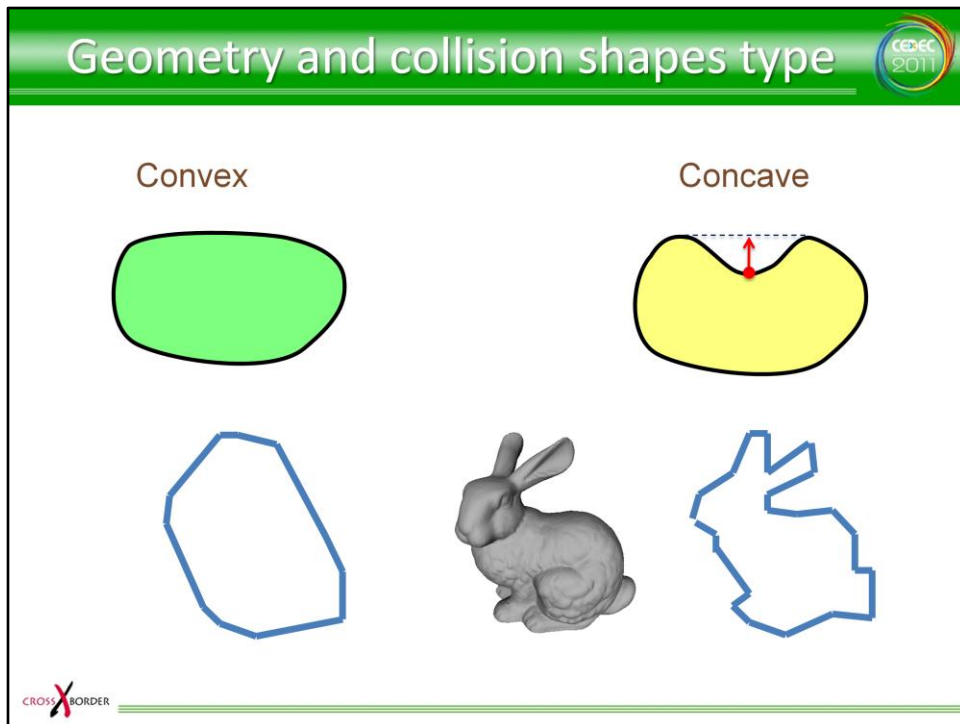
There are many different ways to prepare the 3D geometry for fracture. In the following slides, I will briefly discuss each method.

You can cut a model, represented as a closed 3D mesh into pieces using voronoi diagrams.

Another way of cutting geometry is using Boolean operations.

A third way is using convex decomposition, this can be performed by artists by hand, or using automatic tools.

It can also be done by converting the 3D model into tetra, this process is called tetrahedralization.



The 3D mesh geometry used for graphics rendering is often simplified when performing physics simulation.

1) For geometry that is not changing and not moving, such as world environment, buildings we often use the 3D concave triangle meshes as collision geometry. In some cases it is simplified for collision performance, but it is still a concave mesh.

Dealing with concave triangle meshes for moving objects is often avoided in real-time physics engines. It is hard to maintain a high performance, when two concave triangle meshes collide, and creating good quality contact points between 3d concave triangle meshes is an open research area.

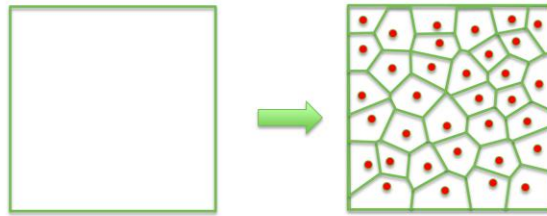
It is easier to compute contact points when dealing with convex shapes: the separating vector and penetration distance is well defined. You can use known algorithms such as the separating axis test or GJK to compute contact information.

Simple convex shapes such as boxes, spheres and capsules give the best performance and stability.

If you need a better approximation than those implicit primitives, you can use the convex hull of a 3D mesh. This convex hull can be represented as a point cloud when using certain collision algorithms, such as GJK.

Multiple convex hulls or convex primitives can be combined into a composite, or compound: this is a single rigid body with multiple convex collision shapes. Such composite rigid body can approximate the shape of the geometry better than a single convex primitive, while still providing good performance and stability.

Voronoi shatter steps



- Distribute points / particles inside the 3D model
- Create Voronoi regions enclosing those points



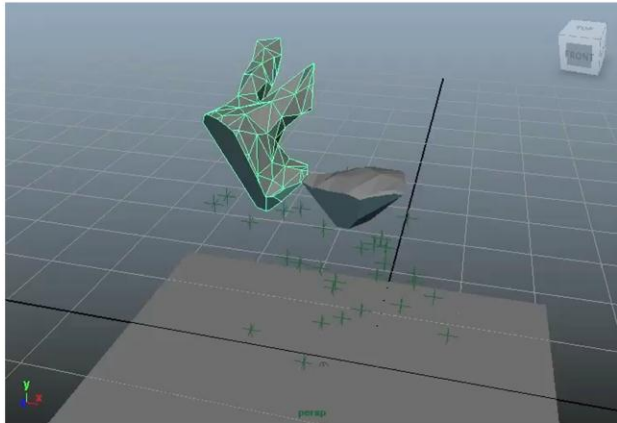
Cutting a 3D model into pieces using voronoi shatter is very popular. For example it is used by the Epic Unreal Engine.

You first create particles inside the 3D model. Then you create Voronoi regions that enclose those points.

Some good algorithms are developed to compute voronoi regions efficiently, given a closed mesh and enclosed points, for example Fortune's algorithm.

There is an open source Python script for Maya written by [Dave Greenwood](#). It uses a brute-force approach to build the voronoi regions.

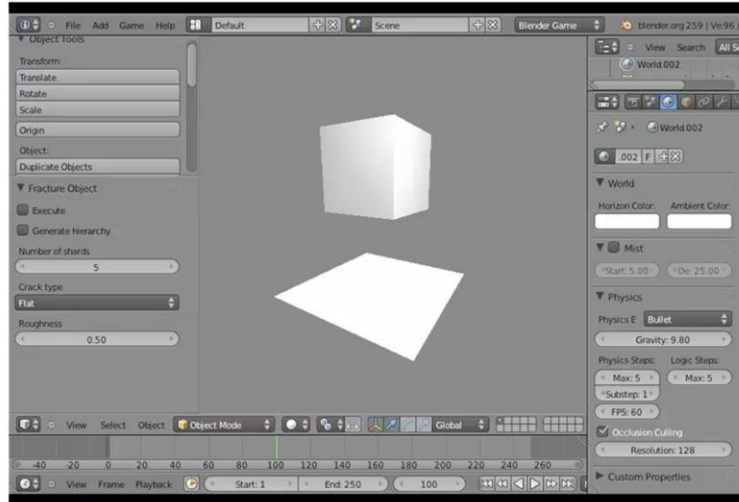
Voronoi scatter in Maya



David Greenwood, <http://bit.ly/q4nUKp>
<http://dynamica.googlecode.com>



Blender Fracture script

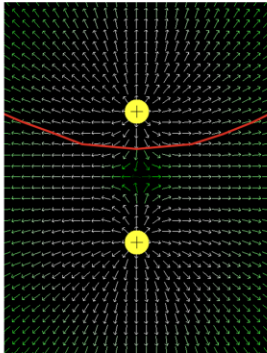


<http://www.directcg.info/how-to-use-fracturetools-inside-blender.html>

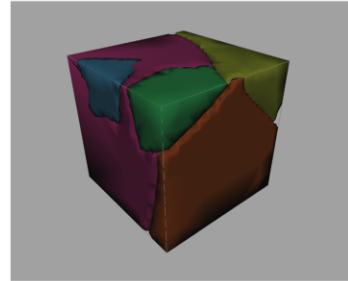


<http://www.directcg.info/how-to-use-fracturetools-inside-blender.html>

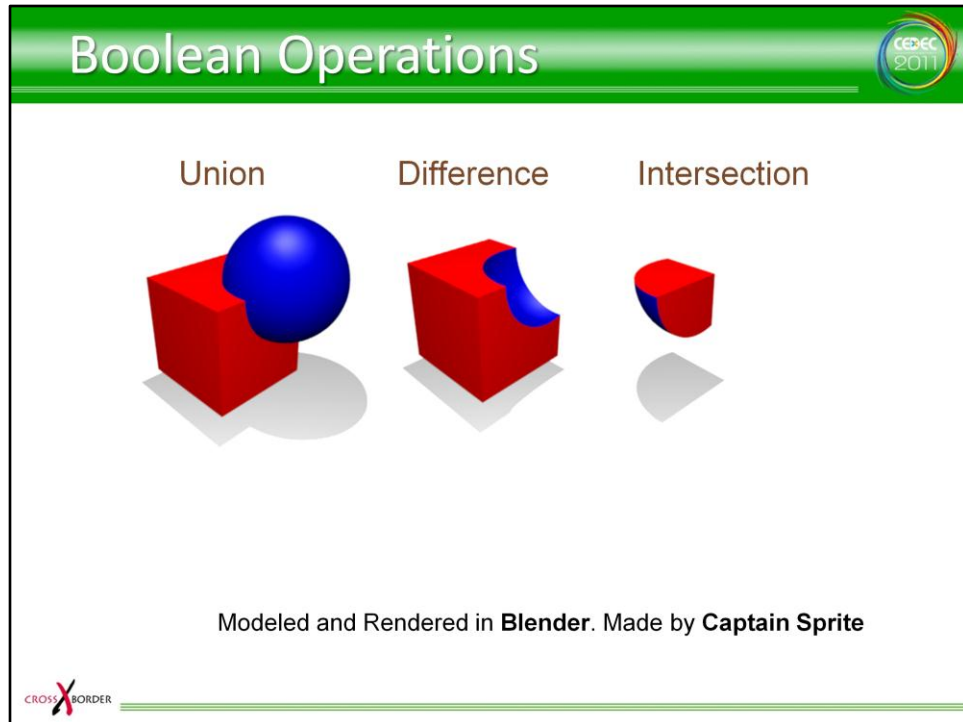
Electric fields



$$E = \sum_{i=1}^n \frac{q_i}{4\pi\epsilon_0 r_i^2} \hat{r}_i$$



- Points can have different charge, concave boundaries
- Use Dual Contouring to generate the boundary mesh
- See the 'Fragged' article in GDMag 2010, December



Boolean operations, also known as constructive solid geometry (CSG) is a way to perform volumetric operations between 3D models.

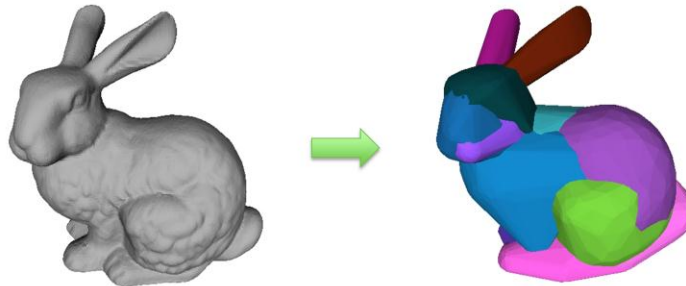
For example you can add two volumes together, or compute the difference between two objects, or take the intersection.

Those operations allow you to break original 3D models into smaller pieces, similar to a cookie cutter.

Convex Decomposition



- (Semi) Automatic physics shape generation



We can also create a convex decomposition of a concave triangle mesh using convex decomposition.

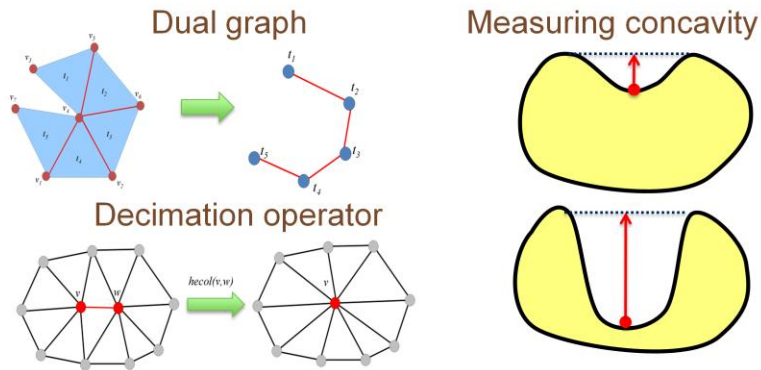
A artist can create a convex decomposition manually, using simple convex primitives such as boxes spheres and capsules.

It is also possible to automatically create a convex decomposition. This process might involve the tweaking of parameters, so it is not fully automatic.

There are some free implementations, one is by John Ratcliff. This is a top down aproach: it recursively breaks down a concave mesh into parts, until each part is convex.

Khaled Mammou was inspired by John's work and developed a bottom up approach called HACD.

HACD in one slide



- Hierarchical Approximate Convex Decomposition
by Khaled Mammou, ICIP 09
- <http://sourceforge.net/projects/hacd>



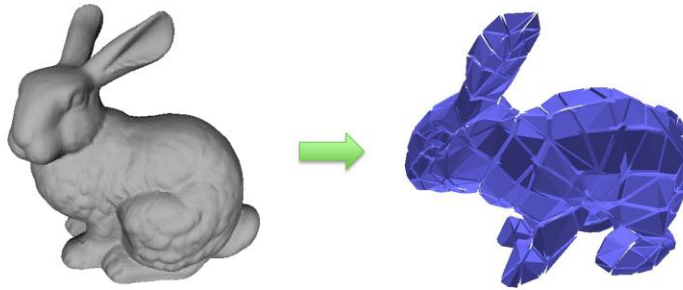
Khaled Mammou's Hierarchical Approximate Convex Decomposition (or HACD) is a bottom up approach.

Given a triangle mesh, it will compute a dual graph. The nodes in the dual graph represent triangles in the original mesh, while the links in the dual graph represent shared edges in the original mesh.

We create and maintain a set of clusters. Initially each triangle (or node in the dual graph) is its own cluster.

We will keep on merging neighboring nodes in the dual graph, based on some concavity measurement.

Creating Tetrahedra



- Netgen, <http://sourceforge.net/projects/netgen-mesher>
- Tetgen, <http://tetgen.berlios.de>
- Maya 2012 DMM plugin by Pixelux



A mesh can also be decomposed into tetrahedral elements using Delaunay triangulation, this is the dual of voronoi regions.

There are some open source implementations available, including Netgen, Tetgen.

The Maya 2012 Digital Molecular Matter plugin by Pixelux, uses Netgen internally to perform a tetrahedralization.

Runtime Destruction



Geometry Preparation	Runtime Destruction
Voronoi shatter, slicing	Canned animation
Boolean operations	Real-time Booleans
Convex decomposition	FEM, particle based
Tetrahedralization	Rigid body & Hybrid



Once we have performed the preparation, we can perform the runtime destruction.

Many games still use canned animation to perform destruction and fracture effects. This means that the fracturing effect has been simulated as a preparation step, and at runtime we just trigger the playback of an animation. This works well as long as there is no two-way interaction required with the fractured parts (debris).

Other methods that I will briefly discuss in the following slides include real-time boolean operations, particle based methods, finite element method based method, and rigid body based methods.

Real-time Booleans



- <http://melax.googlecode.com>



Boolean operations can be used in realtime in games. Some good examples are the Red Faction games, by Volition. Their technology is called Geo Mod. <http://en.wikipedia.org/wiki/Geo-Mod>

Stan Melax has an open source implementation of real-time boolean operations. It is available from <http://melax.googlecode.com>
His work is based on the paper "Merging BSP trees yields polyhedral set operations" by [Bruce Naylor](#), SIGGRAPH '90

There are many challenges when using boolean operations in games. One of the challenges is related to level design: you need to create levels keeping in mind all the possible modifications by players.
After the boolean operations perform the destruction, you might need to perform some physical simulation to check if disconnected pieces need to fall down.

Connected Particles

- Spring damper systems or Position Based Dynamics
Game Physics Pearls, Thomas Jakobsen, Matthias Müller
- Extra links for bending, volume preservation
Kenny Erleben, Jan Bender

CROSS BORDER

Particle based methods can also be used for destruction and fracture.

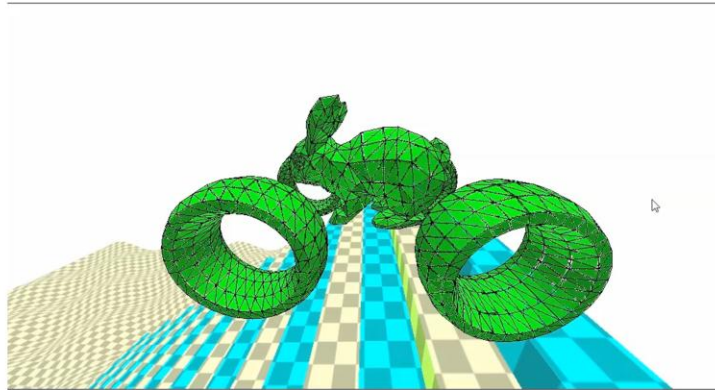
You can connect two neighboring particles by links to form a 1D rope, or you can connect 3 particles to form a 2D triangle, and connecting 4 particles to create a tetrahedron.

It is possible to use additional links between particles to simulate bending constraints and shearing constraints.

There are many ways of simulating the motion of such connected particles. One popular way is based on spring-dampers, using a formulation based on forces and accelerations.

Another way is to use a position based formulation. You can use a verlet style integrator, where the velocity is implicitly defined by the change in position between the current and previous frame.

If constraints, such as collision constraints or lengths of links are violated, they can be corrected directly changing the position of particles. This is also called projection.



- Bullet, <http://bulletphysics.org>

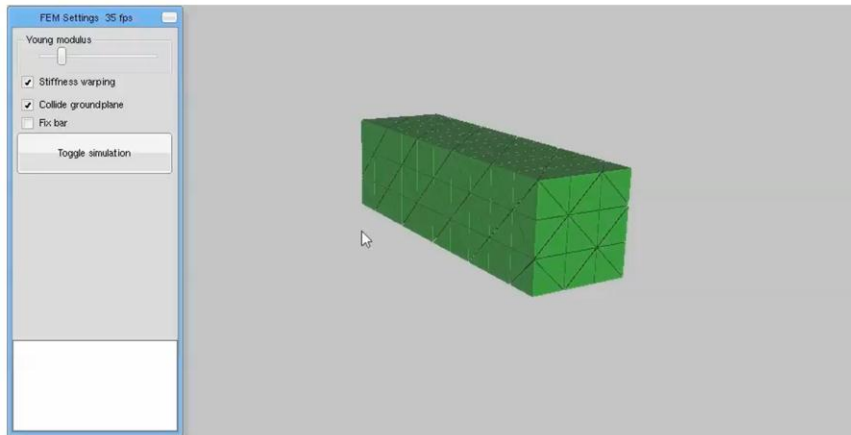


The position based dynamics method, as described by Thomas Jakobsen and later refined by Matthias Mueller, has been implemented in our open source Bullet library. This movie shows a deformable bunny, with 4 deformable wheels. The bunny and wheels are connected by special hinge constraints between soft bodies.

The collision detection for deformable objects is interesting in this case: instead of simply colliding with the surface triangles, the collision happens using convex clusters.

An efficient acceleration structure based on dynamic AABB trees is used.

Finite Element Method (FEM)



<https://github.com/erwincoumans/experiments>




A more physically correct way of simulating deformation and fracture is based on continuum mechanics, and it is called the finite element method.


A 3D mesh is approximated using a collection of elements, usually tetrahedra. The strains, stress and stiffness matrix is used to compute the effect of forces and deformations.

Here is an open source implementation, originally from the Open Tissue library, a project lead by Kenny Erleben from Denmark University.


FEM on the GPU



FPS: 94.3
GPU: GeForce GTX 580
particles: 13443
elements: 45206
CG iterations: 25



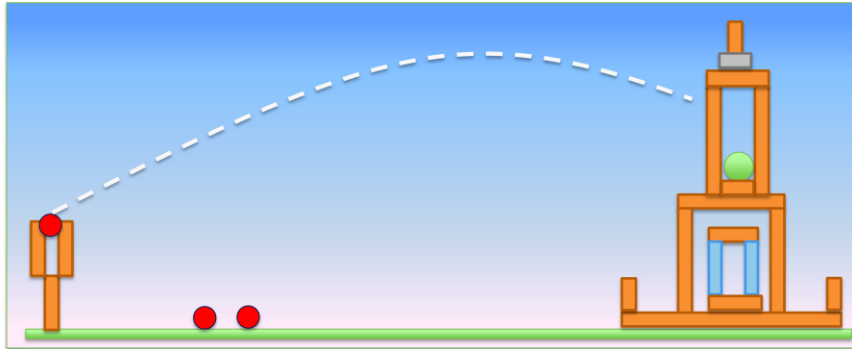
<http://sofa-framework.org>



This year, some researchers from Inria, France, showed the finite element method running on the GPU.

This way you can simulation tens of thousands of tetrahedra.

Rigid Bodies



- <http://box2d.org>, Erin Catto



When using rigid bodies, the most simple approach is to rely purely on the default simulation destruction.

For example the popular game Angry Birds is using the open source Box2D rigid body engine to simulate the destruction of the structures.

Breakable rigid bodies



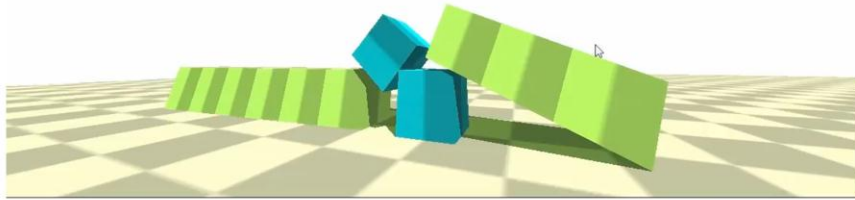
- Composite single rigid body
- Breakable constraints
- Hybrid methods



There are methods on top of rigid body dynamics, that give some more control when and where the fracture happens.

In the following slides I will briefly discuss the composite rigid body method, breakable constraints and hybrid methods.


Composite single rigid body




Here is a video that shows a breaking bar using the composite rigid body method.

You can see that the objects are perfectly rigid, before and after the fracture.

Steps for the composite method



- Break an object into parts
- Automatically create connections
 - based on contact points (collision detection)
 - assign a breaking threshold to connections
- At run-time propagate a collision impulse
 - break connections if the impulse > threshold
 - determine disconnected parts using union find
 - create new rigid bodies for each disconnected part
 - update inertia tensor and velocity



Here are some steps that describe the composite rigid body fracture method.

First we prepare the geometry into fractured pieces and now we need to 'glue' them together:

We need to create some connections between those pieces. There are several ways to do this.

One way is to define connections between every piece and every other piece. This gives the most control, but the performance can be slow due to the many connections.

Another way to compute connections is to automatically compute them based on collision detection: compute the contact points between touching pieces, and only create connections when there is a contact point. Then you can create a breaking threshold for those connections.

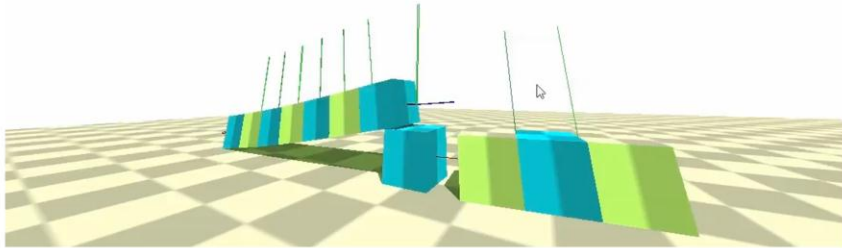
Once we glued the pieces into a single rigid body, we can perform the runtime fracture:

If there is a collision impact, we compute the impulse. If this impulse is larger than a chosen threshold, we propagate this impulse through the connections. Those connections can be weakened or broken.

After this, we need to determine the disconnected parts, we use the union find algorithm for this, and then create new rigid bodies for each separate part. Of course the inertia matrix needs to be updated properly.

In the open source Bullet physics library there is a demo that shows how to use this:
It is in `Bullet/Demos/BulletFractureDemo`

Breakable constraints



We can also simulate the object by using a rigid body for each fracture piece, and connect the pieces using breakable constraints.

For example this can be a fixed rigid body constraint that connects two pieces. You can see that the object is bending a little bit after the collision, because the constraints are not perfectly stiff.

This bending effect can be desired in some cases, and this method has been used in some movies, for example the 2012 movie by Sony Pictures Imageworks.

Steps for breakable constraints



- Break an object into parts
- Automatically create constraints
 - based on **contact points** (collision detection)
 - assign a **breaking threshold** to constraints
- At run-time propagate a collision impulse
 - break connections if the **impulse > threshold**



The setup is similar to the composite rigid body method.

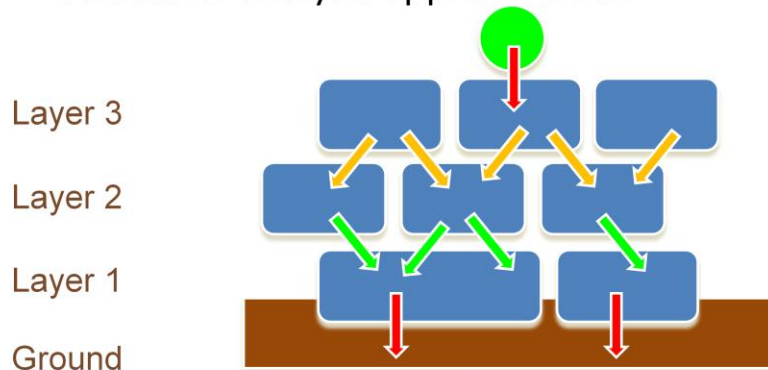
Instead of connections, we create constraints between neighboring rigid bodies. We can use the collision detection to determine if objects are touching.

Then the breaking threshold of the constraints can be chosen, to control when and where the parts will break off.

The run-time fracture is much simpler, because it relies on the existing rigid body simulation:

once an impulse in the constraint solver exceeds a breaking threshold, the constraint will be deactivated.

- Structural analysis approximation



Hybrid method



- Composite rigid body with static FEM analysis
 - Matthias Müller et al. Eurographics CAS 2001



It is also possible to use a hybrid method based on rigid body and finite element method.

The propagation of a collision impulse in the composite rigid body method is not a very good approximation of strain and stress.

For example, it will not handle structural fracture, where the object breaks due to its own weight.

The finite element method to compute the strain and stress and if the FEM analysis determines that the object should break, we can break the rigid bodies into multiple pieces.

This hybrid method has been discussed in a paper by Matthias Müller at the Eurographics conference in 2001.

References



Bullet Physics website

- <http://bulletphysics.org>
- <http://bulletphysics.org/siggraph2011>
- <http://bullet.googlecode.com>
- <http://github.com/erwincoumans>

Other web sites:

- <http://bulletphysics.org>
- <http://iphys.wordpress.com>
- <http://graphics.ewha.ac.kr>
- <http://gamma.cs.unc.edu/research/collision>
- <http://matthiasmueller.info>
- <http://box2d.org>

Books:

- Game Physics Pearls, AK Peters
- Physics-Based Animation by Kenny Erleben et al.
- Real-Time Collision Detection by Christer Ericson.



Thanks a lot for attending my presentation.

Please try out our open source Bullet physics library, you can download it from <http://bullet.googlecode.com>

Also, if you have questions, you can visit the Bullet physics forums at <http://bulletphysics.org> or email me directly at erwin.coumans@gmail.com

Thank you