
SIGGRAPH2011

Overview of destruction and dynamics methods

Erwin Coumans, AMD

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 overview 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.

Agenda

- 2:00pm Course introduction and overview of techniques,
  Erwin Coumans, AMD
- 2:30pm Authoring destruction with the Dynamica Maya plugin,
  Michael Baker, Art Institute of Las Vegas
- 2:45pm Destruction and dynamics artist tools for film,
  Nafees Bin Zafar and Mark Carlson, Dreamworks Animation
- 3:30pm Break
- 3:45pm Deformable rigid bodies and fragment clustering for film,
  Brice Criswell, Industrial Light & Magic
- 4:30pm Procedurally generating fragmented meshes for games,
  Phil Knight, Disney Interactive Studios
- 4:45pm Accelerating rigid body simulation using GPU,
  Takahiro Harada, AMD
- 5:15pm End

This is the agenda for todays course.

Offline versus runtime

SIGGRAPH2011

- 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 plauing the game. This task is performed by the physics engine, as part of the game engine.

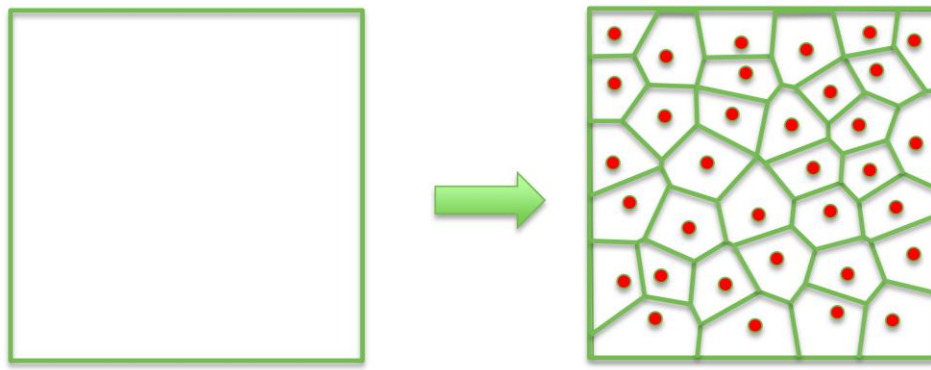
I will first discuss the geometry preparation and then the runtime destruction methods at the end.

Geometry Preparation	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.

Voronoi shatter



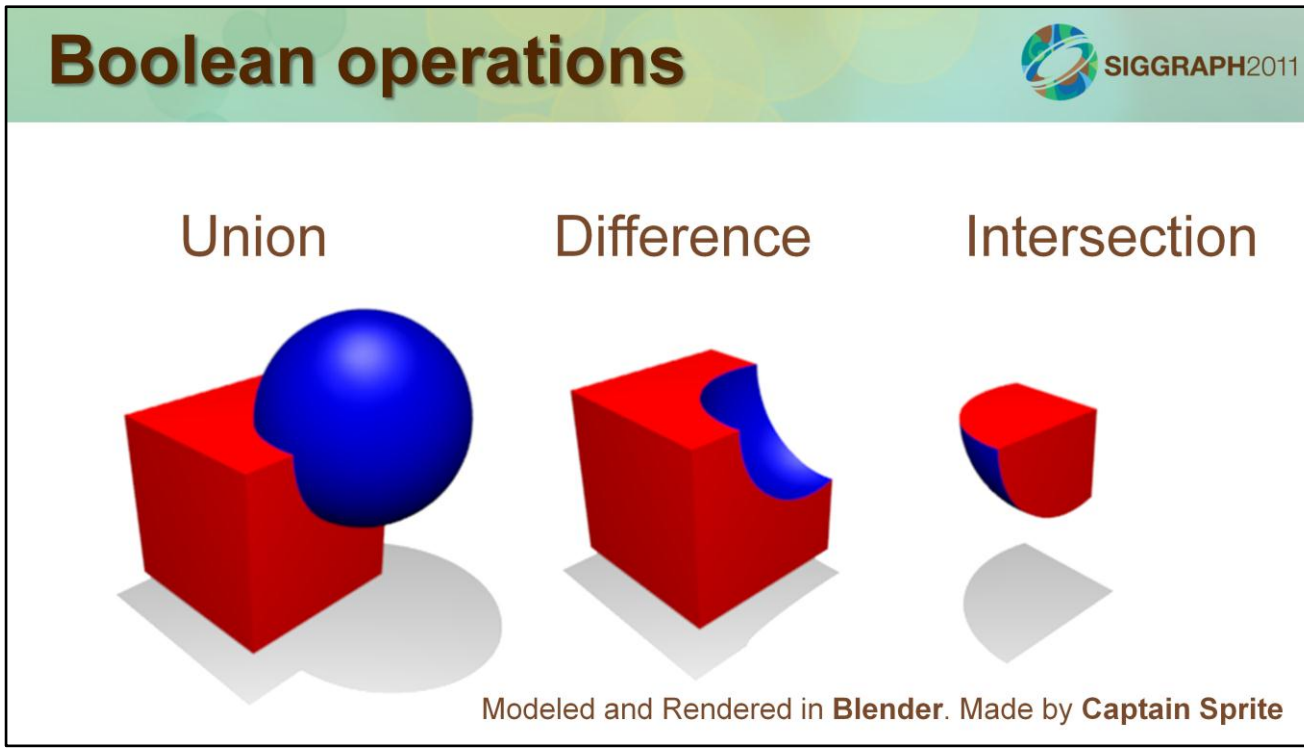
- Demo by Michael Baker using Dave Greenwood's Maya script

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.
See http://bit.ly/q4nUKp

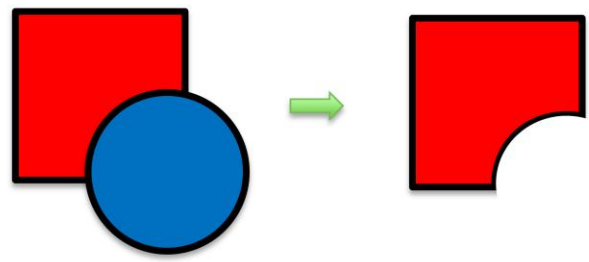


Boolean operations, also known as constructive solid geometry (CSG) is a way to perform volumetric operations between 3D models.

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.

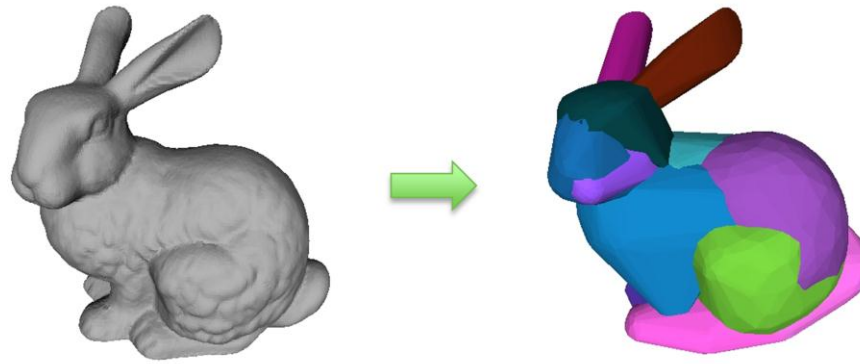
Polyhedral Booleans

- Merging BSP trees yields polyhedral set operations, Bruce Naylor, SIGGRAPH '90



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.

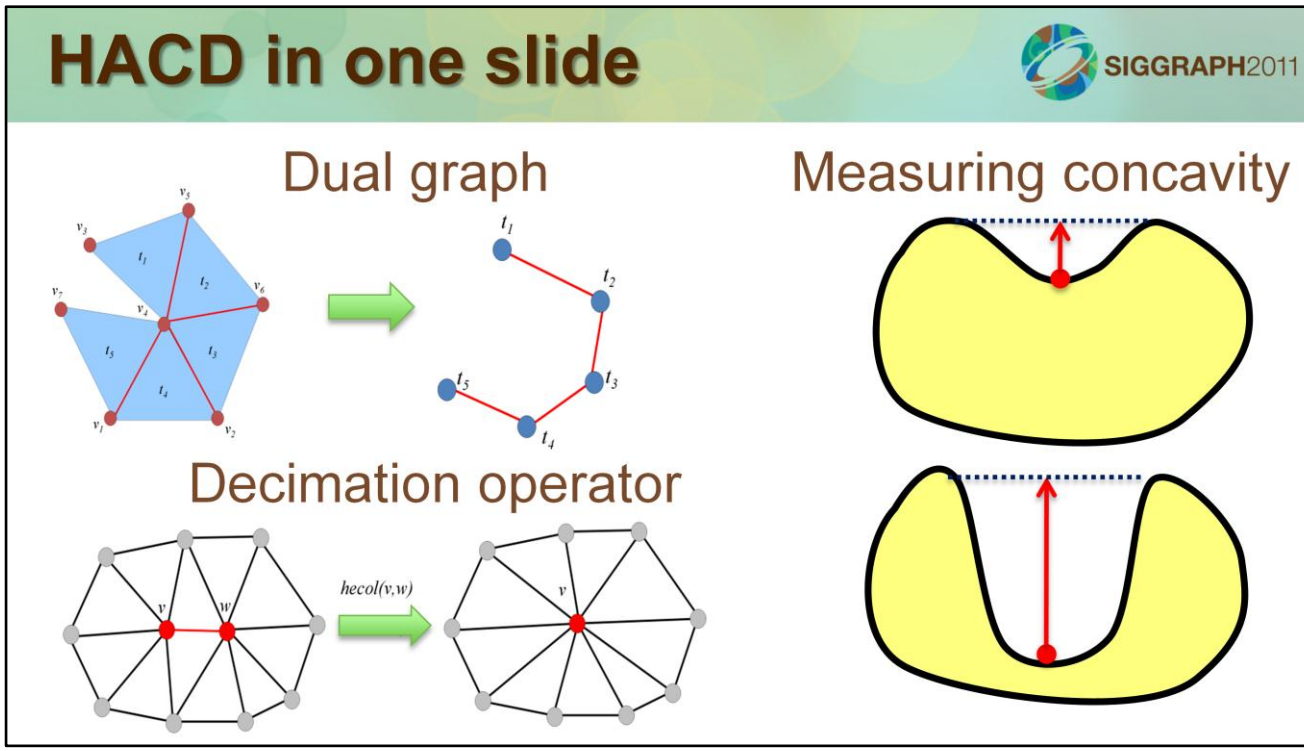
HACD

- Hierarchical Approximate Convex Decomposition by Khaled Mammou, ICIP 09
- Bottom up, merging convex clusters
- http://sourceforge.net/projects/hacd

Although the convex decomposition problem is NP hard, you can implement an approximate method using top down or bottom up.

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.

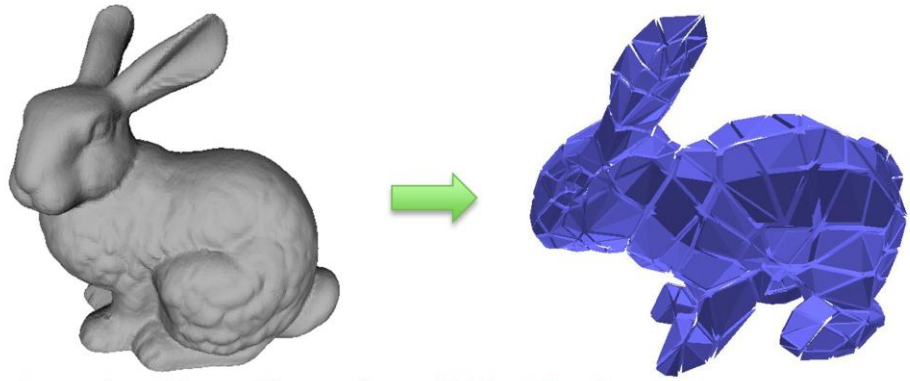


Given a triangle mesh, it will compute a dual graph. The nodes in the dual graph represent triangles in the original mesh,
while the links in the dual graph represent shared edges in the original mesh.

We create and maintain a set of clusters. Initally 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.
See *Game Programming Gems 8 - Chapter 2.8, p.202* or see http://bit.ly/p5VEAl

Tetrahedra Creation



- Tetrahedralization by 3D Delaunay Triangulation, Boris Delaunay, 1934

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.

Tetrahedralization

- Netgen, LGPL, http://sourceforge.net/projects/netgen-mesher
- Tetgen
- Maya 2012 with DMM plugin, Pixelux Digital Molecular Matter

There are some free open source implementations that can create a tetrahedral mesh from a 3d closed triangle mesh.

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

- Stan Melax, 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.

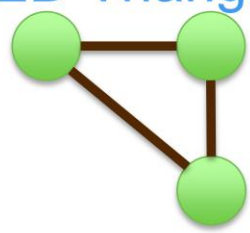
Connected particles

SIGGRAPH2011

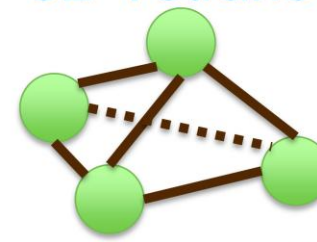
- Spring damper systems or Position Based Dynamics
  Game Physics Pearls, Thomas Jakobsen, Matthias Müller

1D Rope

2D Triangle



3D Tetrahedron



- Extra links for bending, volume preservation
  Kenny Erleben, Jan Bender

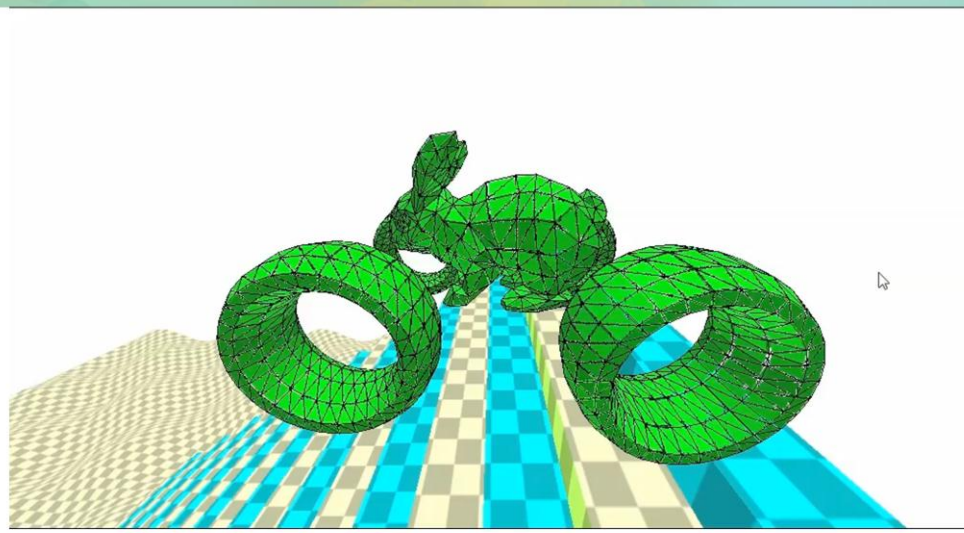
Particle based methods can also be used for destruction and fracture.

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.

Particles based deformables



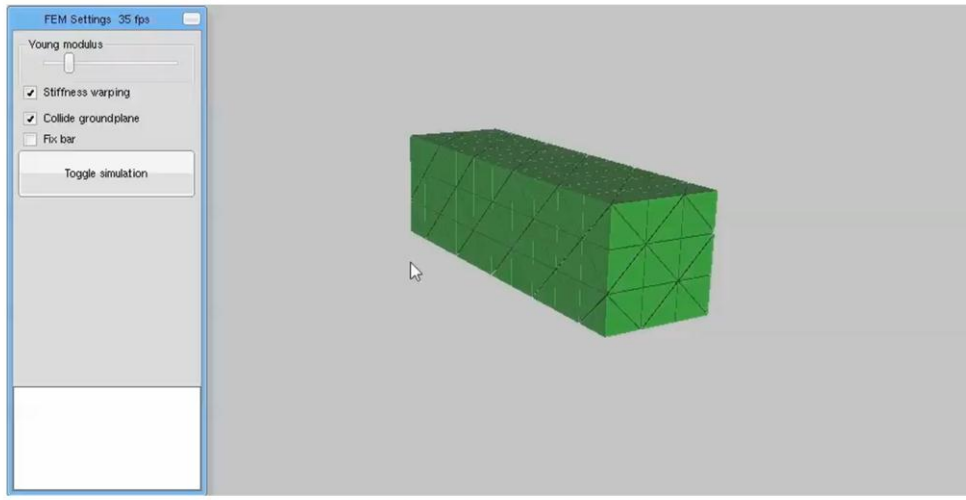
- Bullet, http://bulletphysics.org

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



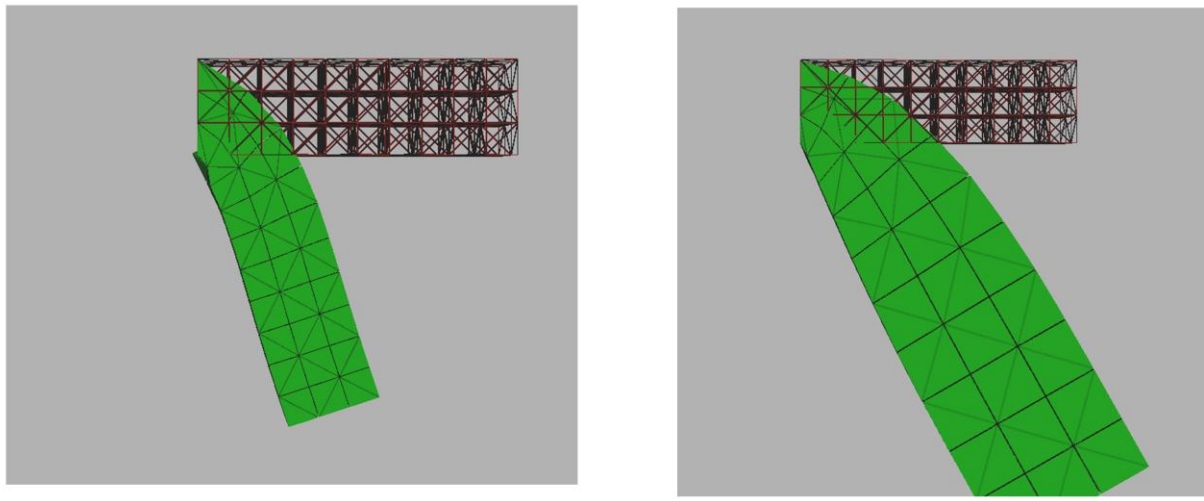
- OpenTissue, http://opentissue.org

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.

Stiffness warping



On the left side you can see the effect of stiffness warping, right it is disabled.

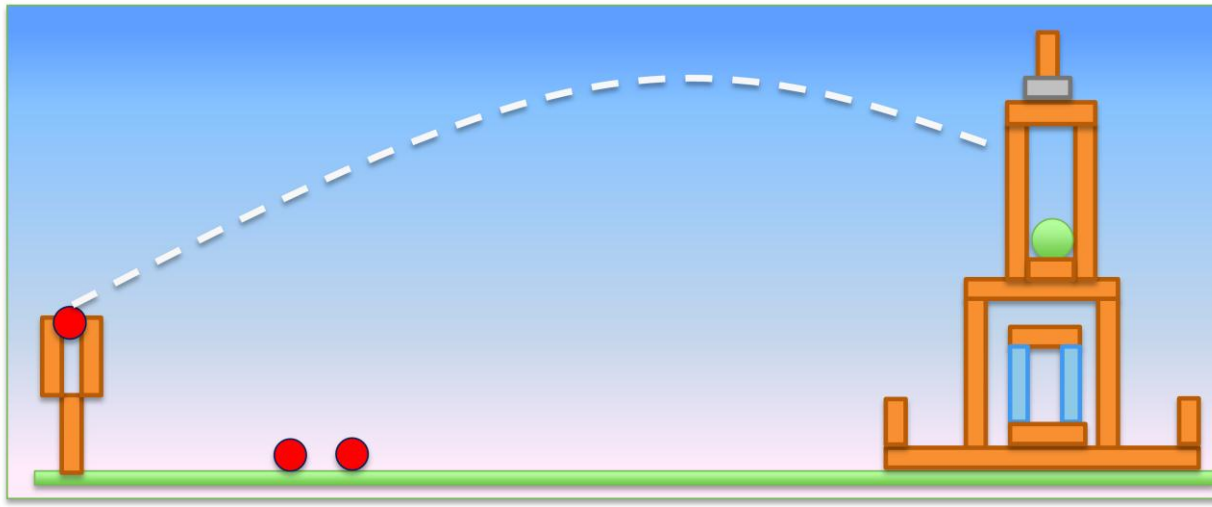
FEM on GPU

- SOFA, 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

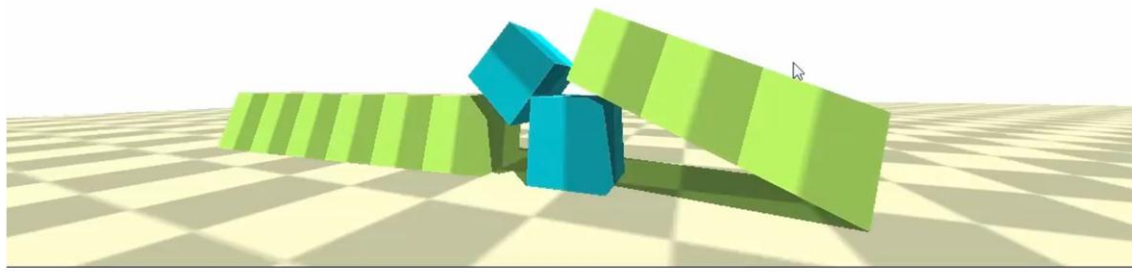
- Composite single rigid body
- Breakable constraints
- Hybrid methods

- See also Nafees Bin Zafar, Mark Carlson, Brice Chriswell presentations

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.

Composite single rigid body

- Break an object into parts
- Automatically create connections
  - based on contact points (collision detecion)
  - 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, 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.

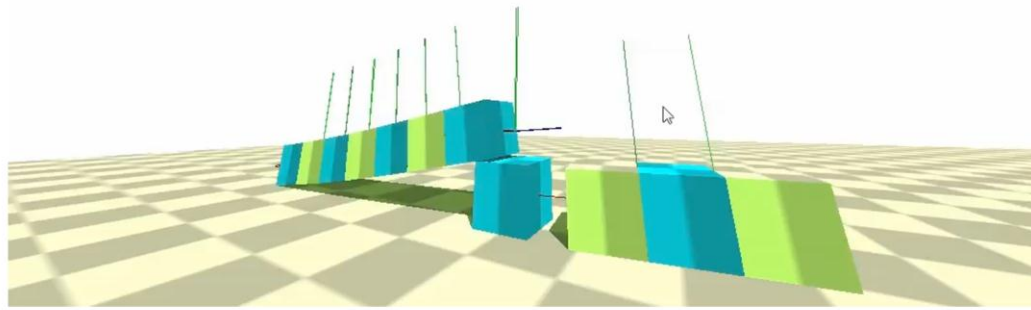
Once we glued the pieces into a single rigid body, we can perform the runtime fracture:

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.

Hybrid method

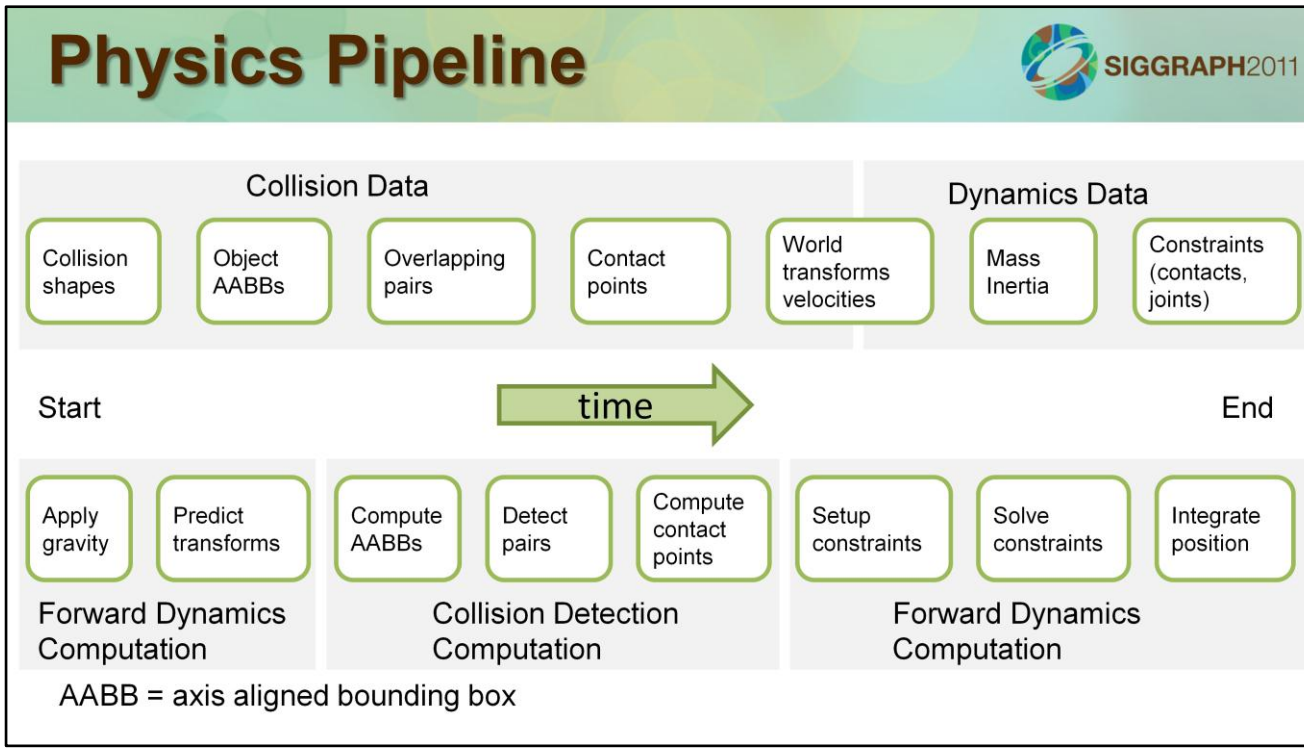
- Composite rigid body with static FEM analysis
  - Matthias Müller et al. Eurographics CAS 2001

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 an 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 Muller at the Eurographics conference in 2001.

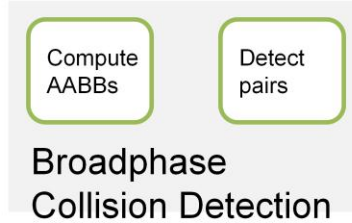


Here is in a nutshell a typical rigid body physics pipeline. The steps are performed in order from the left to the right.

At the top you can see the main data structures, and at the bottom the operations performed on this data. AABB stands for axis aligned bounding box, a bounding volume used for fast culling.

Broadphase CD

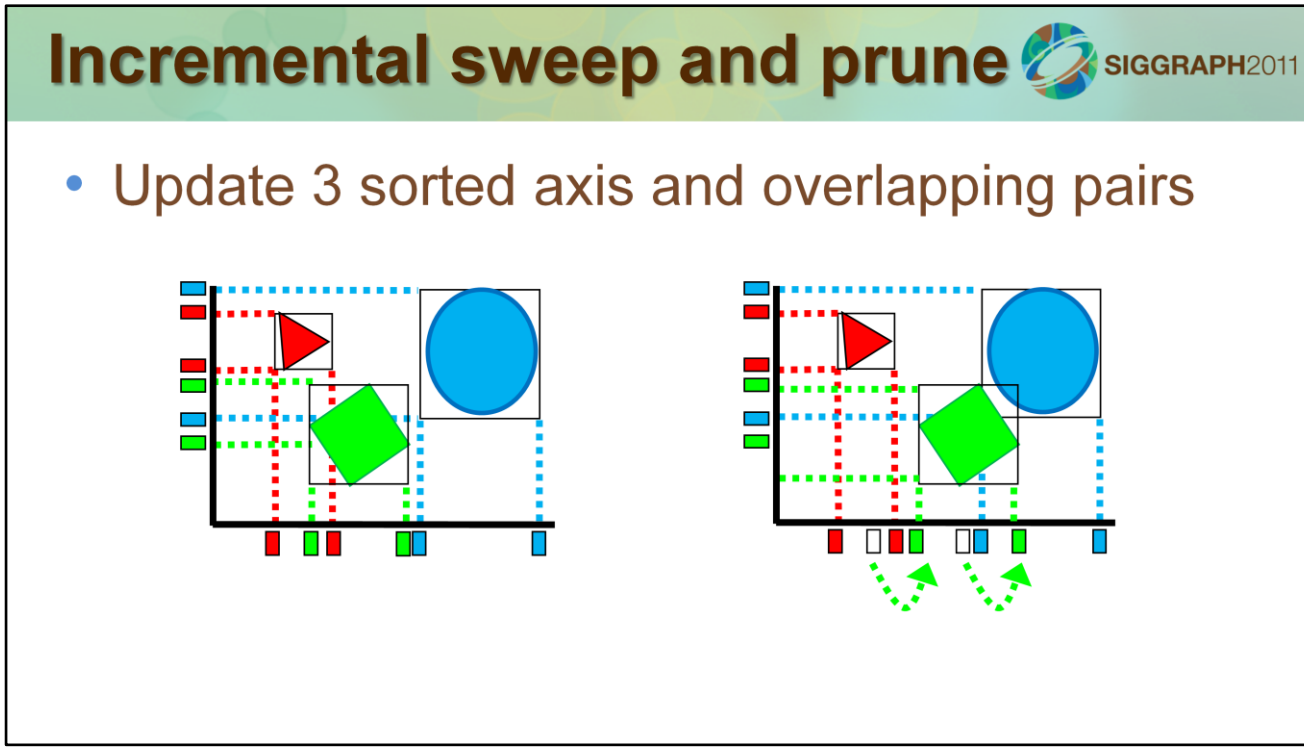
- Avoid brute-force N*N tests



- Input: world space BVs and unique IDs
- Output: array of potential overlapping pairs

The first collision detection stage perform culling, usually based on bounding volumes such as AABBs or bounding spheres.

Incremental sweep and prune



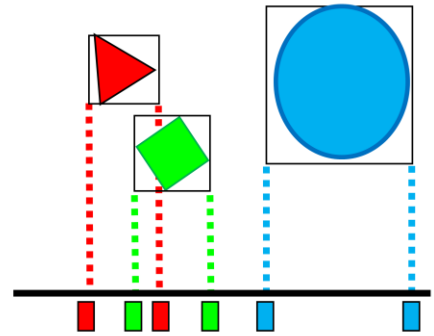
One way to compute overlapping pairs is to incrementally sort them on all 3 axis, and maintain the overlapping pairs incrementally,
adding and removing them once the projections of all begin and end points overlap in all 3 axis.

You only potentially need to add/remove pairs when you need to 'swap' the projected points to keep them in sorted order.

This algorithm is difficult to parallelize due to code complexity and data dependencies between the 3 axis.

1 axis sweep and prune

- From scratch sort 1 axis sweep to find all pairs



- Parallel bitonic sort and sweep in parallel

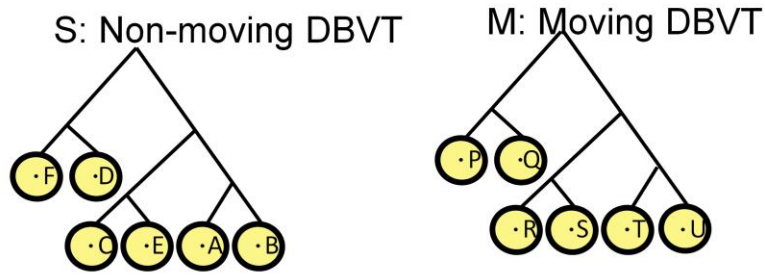
  Game Physics Pearls, 2010, AK Peters

We can also only maintain the sorted pairs on a single axis. We perform a full sort, instead of an incremental sort.

The implementation is simpler and it is easier to parallelize, but performance might be degraded if we can't choose a good axis.

Dynamic AABB trees

- Keep two dynamic trees



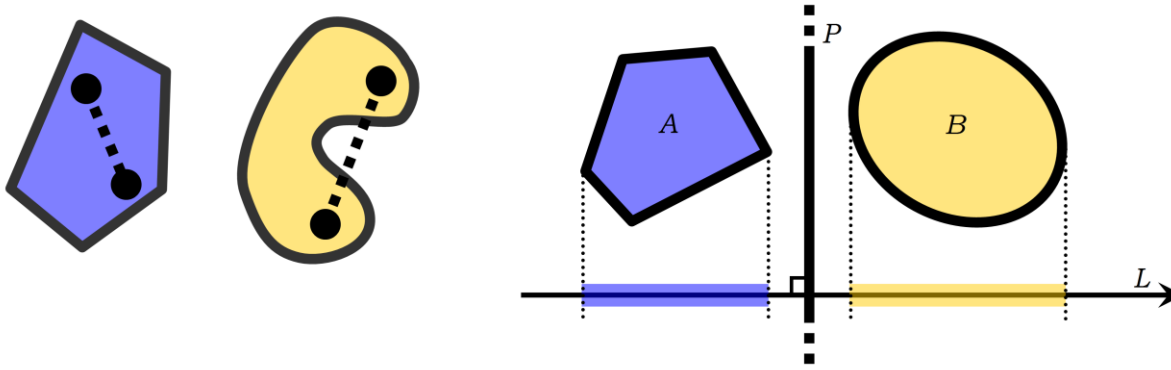
- Find overlapping pairs:
  - Overlap M versus M and Overlap M versus S

A dynamic AABB tree is a very versatile acceleration structure. It can be used to accelerate ray casts, occlusion culling and for finding overlapping pairs.
Here is a desciption in a nutshell:

One tree for moving objects, other for objects (sleeping/static)

If new AABB is contained by old do nothing
Otherwise remove and re-insert leaf
    Re-insert at closest ancestor that was not resized during remove
Expand AABB with margin
    Avoid updates due to jitter or small random motion
Expand AABB with velocity
    Handle the case of linear motion over n frames

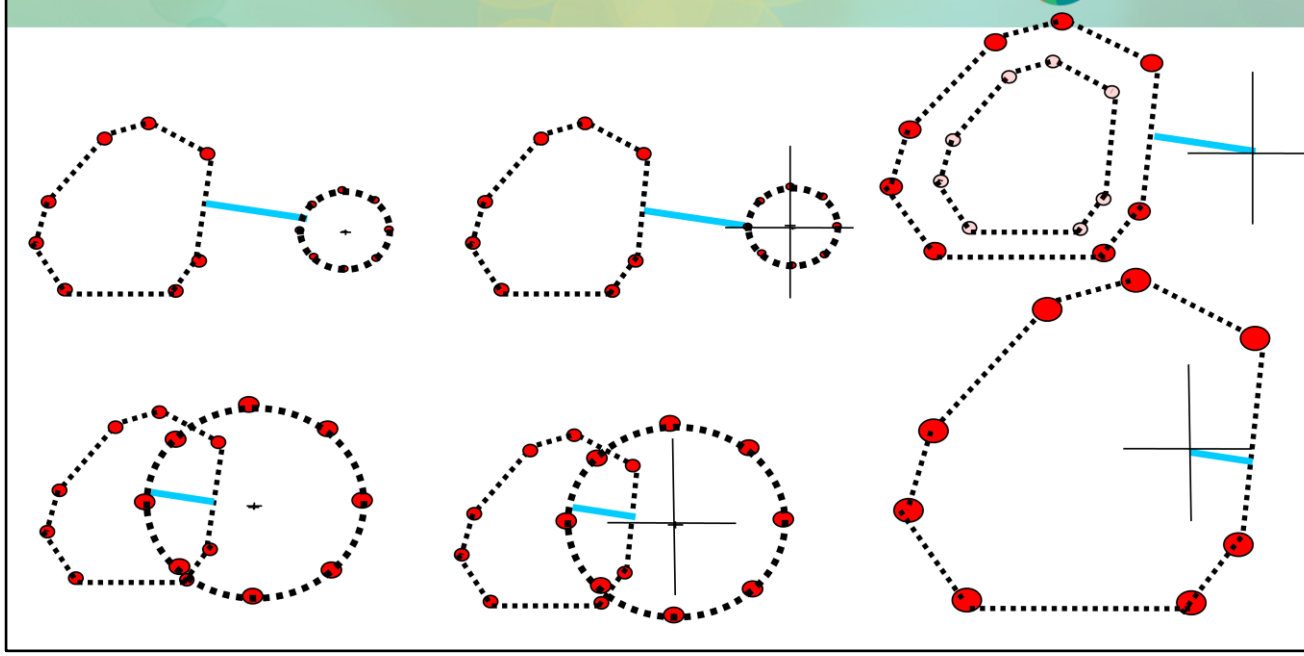
Convex CD: SAT



- Separating Axis Theorem
  - Real-time collision detection, Christer Ericson

One way to compute overlap, closest points, between convex objects is using the Separating Axis Theorem. You can find more details in Christer Ericson's book.

Convex CD: GJK and EPA



A more general method to compute the closest points and distance between convex shapes is called GJK, named after its creators.
Convex shapes are described using a support map, the vertex on the shape furthest away given a certain direction.
Then the distance query between two convex shapes is reduced to a distance query between a single convex object and the origin,
using the concept of Minkowski sum/difference.

GJK only works when objects are separate, so a companion algorithm is required for the overlapping case, such as the EPA, expanding polytope algorythm.

GJK and EPA

- Gilbert Keerthi Johnson, Expanding polythope algorithm
  - Collision detection in interactive 3D environments, Gino van den Bergen

GJK and similar algorithms are versatile and can also be used for ray intersection test and sweeping convex shapes against other convex shapes.

You can read more about GJK in the book by Gino van den Bergen.

Multiple contact points

- Sutherland Hodgman polygon clipping
- See also the Game Physics Pearls book
- Persistent contact caching
  - Contact Generation, Game Developers Conference 2010, Erwin Coumans, http://code.google.com/p/bullet/downloads/list

For stable resting bodies and stacking, it is best to compute multiple contact points. One way to compute multiple contact points is by using polygon clipping. Another way is to add one or more contact points to a point cache, and accumulate multiple contact points over multiple frames. See my GDC presentation at this link.

The open source Bullet library has an implementation of both methods.

Constraint solving

- Iterative LCP solutions, Projected Gauss Seidel (PGS), Successive Over Relaxation (SOR), NNCG
  - Physics Based Animation, Kenny Erleben
  - A Nonsmooth Nonlinear Conjugate Gradient Method for Interactive Contact Force Problems
- See Takahiro Harada's presentation today

We need to find the forces and velocities in such a way that at the end of the timestep the constraints are satisfied.

This includes contacts constraints and also other joint types such as hinge, ball-socket, joint limits and joint motors. Constraints can be formulated in many ways, one is using a mixed linear complementarity problem. This MLCP can be numerically solved using an iterative solver such as Projected Gauss Seidel and variations.

Takahiro will show how to solve such constraints on the GPU.

Thanks!

- Questions: erwin.coumans@gmail.com

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

References

- http://bulletphysics.org
- Physics-Based Animation by Kenny Erleben et al.
- Real-Time Collision Detection by Christer Ericson.
- Game Physics Pearls, AK Peters
- http://iphys.wordpress.com Kenny Erleben
- http://graphics.ewha.ac.kr Young Kim
- http://gamma.cs.unc.edu/research/collision
- http://matthiasmueller.info
- http://box2d.org