

Smash It! - simulating car crash in the Incredible Hulk

Tae-Yong Kim*
Rhythm and Hues Studios

David Horsley†
Rhythm and Hues Studios.



For the recent production of *The Incredible Hulk*, Rhythm and Hues Studios has developed a set of tools to simulate crushing and crumpling of metallic objects (such as car smashed by a hero character).

Metal Deformation

To simulate crumpling of metals, we revised our in-house cloth simulator in many ways. In reality, stiffness of metal is orders of magnitude higher than normal fabric. This presented challenges - cloth would stretch unrealistically and, more seriously, bending models would break simulation as we cranked up the stiffness of bending. In the limit, the cloth object had to move almost like a rigid object,

We derived the bending energy on dihedral angles between a pair of triangles as in prior works [Grinspun et al. 2003]. In such models, force jacobians (as used for implicit integration) are not always positive definite, and this accounted for most instability in the cloth solver when the bending stiffness had to be high (in fact, orders magnitude higher than the usual cloth materials).

We came up with a novel linearization technique that would transform these force Jacobians such that jacobians matrices stay always positive definite regardless of the material stiffness. The resulting simulator was very stable - in fact, we did not have any stability issue with even fixed time stepping scheme of as few as 4 steps per frame. To solve the equations of motion, we used the standard conjugate gradient method for lower stiffness and would turn to the efficient PARDISO solver for higher stiffness.

Plasticity

Once we could simulate very stiff material, the next step was to add a notion of shape memory, so that metals could preserve deformed shape after each crushing event. We added several control on plasticity, updating rest condition on both length and bending forces. When and how these conditions were updated depended on such parameters as amount of collision impact, impact propagation distance, weakening factor of the material upon collision (plastic softening), and hardening factor of the material after collision (plastic hardening). Such parameters could be painted over the cloth surface, yielding different behavior on different parts (e.g. a door would crush upon collision while a tire would bounce back after initial deformation).

Deformation Transfer

The simulator was relatively efficient (it could handle the whole simulation for 50000 triangles under a couple of hours). However,

actual rendered geometry had over a million triangles, and it was clearly not practical to run simulation on such complex geometry.

We started by a proximity based point transfer - for each point of the rendered geometry, we find the closest triangle on the cloth geometry, and store local coordinate of the point w.r.t the triangle. As the cloth gets deformed, we use the stored coordinate to update position of the rendered geometry. When a point is bounded to only one triangle, this tends to generate artifacts for points that were far away from the cloth geometry and/or around highly deformed region. As our cloth geometry was constantly wrinkled due to crushing, this artifact would distort the rendered geometry too much.

On the other hand, using global transfer technique such as Lattice (FFD) would smooth out too much of detail from the underlying simulation. In the end, we developed a tool that seamlessly transfers deformation that would sit somewhere in-between above two techniques. Instead of binding each point to a single triangle, we scanned a region of cloth geometry so that each point would be attached to a set of triangles. The amount of deformation transfer was then weighted by triangle normals and distance to the triangle. We called this technique *multi-proxy*, and it served our purpose well.

Rigid body simulation

When some parts of the car had to be detached (such as doors) due to collision, we turned to rigid body simulation that was run in Houdini. From a single cloth simulation, each breakable piece would be separately deformed using *multi-proxy* until breaking event would happen. Once detached, each piece was converted to a rigid body initialized with deformed shape at the moment, and then simulated with rigid body simulator for the remainder of the shot.

When two metal objects had to collide (such as hummer getting hit against a pole), we simulated heavier object first and then locked it as a rigid body, and ran a second pass simulation of another object while using the first one as collision object. Such simulation layering turned out to be more efficient both in computation and user time than a single simulation on combined objects.

Most of the car crash had to be accompanied with other destructive events, such as breaking window and falling pieces. Breaking window was handled similarly as other rigid objects, except in this case, each fragment of window was handled separately.

References

GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SHRODER, P. 2003. Discrete shells. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 62–67.

*e-mail: tae@rhythm.com

†e-mail: dfh@rhythm.com