Requirements list for the Fracture Modifier (FM) in Blender

by Martin Felke & Kai Kostack, September 2015

1. Subgeometry system:

Object shard management in the FM happens currently in form of shards for fracturing, and mesh islands for simulation. Desirable would be an integration of a mesh island concept (as fourth element in addition to vertices, edges, faces) into Mesh, DerivedMesh and Bmesh / BMEditMesh, so no special DNA structures like shard and mesh island would be necessary any more. This would be a better integration into existing data structures.

Shards should all be part of one single object, so that the depsgraph doesn't need to manage thousands of individual objects. This would lead to higher performance at cache playback and easier to handle by the user. Also it wouldn't clutter the Blender database with ID blocks and would keep the outliner clean.

Mesh islands would have direct references to the vertices which build up the shape of their rigid body. This is necessary for fast access when the vertices need to be transformed after the simulation step has occurred to update the visual render mesh.

2. Multi rigid body system:

When we have multiple independent rigid bodies per object, each rigid body shape can be constructed by the individual shard / mesh island it is assigned to. “Multi rigid body” could become a new rigid body type and should be the default. Single rigid bodies would be covered by just having only one mesh island.

However having two rigid body systems is not optimal. There would be a mapping step necessary between regular rigid bodies and shard rigid bodies across the simulation object group.

3. Caching:

For prefracturing there is a shard / mesh island cache necessary, so you don't need to refracture the object in each modifier evaluation step, even if no changes are being made to the mesh. This is currently implemented as special DNA structs and stored within the .blend file.

For dynamic fracturing either a dynamic cache which supports a growing count of elements and changes in mesh topology would be necessary, or a sequence of shard / mesh island lists, which would denote the mesh state at keyframes where fracturing events happen.

For simulation data currently a fixed point cache is used, which stores motion data (loc, rot) on per frame basis. However this is only sufficient for prefracture.

On a dynamic fracture simulation the cache needs to be dynamically extensible as well, to be able to store more and more elements as the simulation goes on. One idea for the FM was to distribute the cache among all participating objects (so each object has its own cache).

4. Storage:

Both mesh and simulation cache should be storable in the .blend file, or alternatively in an external file so the simulation is ready to start after loading the file. No additional fracture step or first uncached (slow) simulation step should be mandatory in case the cache is valid.