How Red Faction's Geomods Work

Background

I wrote Red Faction's Geomod engine, and rewrote portions of it for the RF2 engine. It basically took a good part of a summer to get it working completely, I did lots of research and learning. See the end of this article for some sources of information that I used.

Overview

Red Faction does a complete realtime subtractive boolean from the world Geometry. The "hole" that is cut out is a loaded mesh that gets randomly rotated to look different everytime. It could be any shape, we just liked how this one worked. It operates on a face basis, splitting some, deleting some, and adding in new ones. The input is a polygon mesh; the output is a polygon mesh.

The math to to the boolean takes place spread out over a few frames so that it doesn't impact framerate much.

All of out structures in the game are dynamic to handle this... the code to actually make the hole is only a minor part of the problem. For instance, our AI paths can update to reflect the hole, the rooms and portals update so a hole between two rooms is now a portal, and the collision detection structures for the world use recursive AABB, which get dynamically updated during the Geomod. Even things like slapping bullet decals on the new faces and making sure to remove decals on faces that get Geomoded away is a mess.

The Geomod itself works on n-sided convex faces, and tesselates and splits all faces as needed. It is complex enough that it works on all input meshes, meaning it can do a lot more than punch a hole through a wall. Neither the input mesh or the geomod mesh need to be convex or even composed of just one primitive.

The Geomod code maintains all face adjacency and fixes any t-joints.

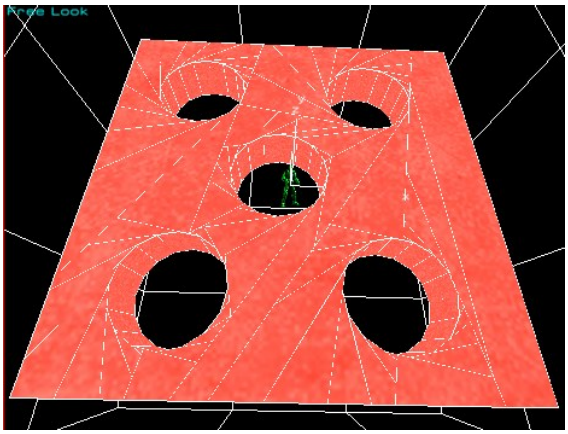
Once the Geomod has completed, the code does a search through adjacent faces and

finds any chunks of the world that are separated from everything else (like a bridge that has been blasted on each end) and separates it into a new piece of geometry and either deletes it or adds physics to it, depending on the version of the engine we're using. RF2 just deleted them, for framerate reasons.

As far as the world slowing down, it's usually not a large problem... if you do a lot of Geomoding, the face count increases, then gets to a point where every Geomod tends to remove about as much geometry as it adds.

In RF2 we actually did a trick, where, if you Geomoded a lot and we started running low on memory, we would remove brushes that were Geomoded a lot a long time ago, rather than limiting the max number of Geomods, like we did in RF1.

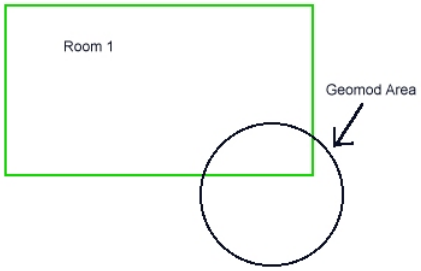
Here is an example of the faces created by RF2... if you used our Geomod and did four Geomods using cylinders against a 6-sided box, you'd get this:



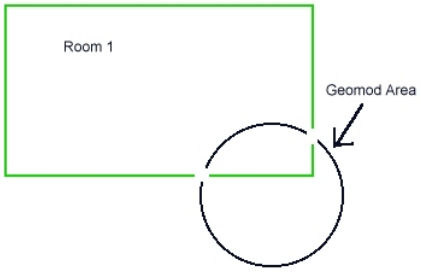
==========

CSG Boolean Operation in Detail

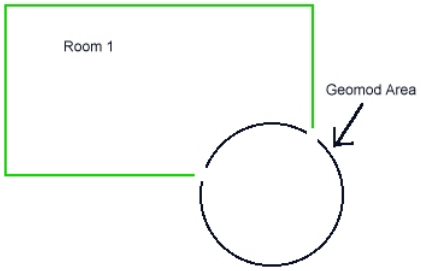
Consider the world "A" and the hole "B" ... assume that "B" is a cylinder made up of polygons with all their normals facing to the inside of it.



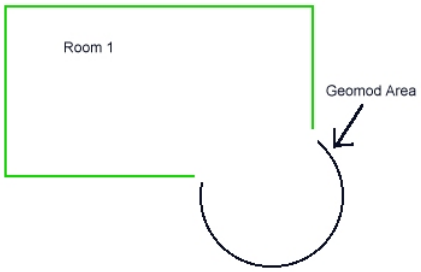
1) Find all the polygons from A and B that intersect each other and divide them into smaller polygons along the lines of intersection, so no polygons overlap, but only touch.



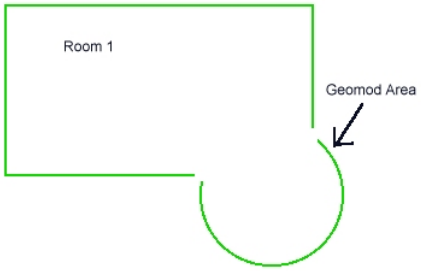
2) Then find all of the polygons from B that are not inside of A and delete them.



3) Then find all of the polygons from A that are inside of B and delete them.



And you're done!!



So the two hard steps are intersecting all of the faces, and determining if a point is inside of the other object.

And the rest is just making it work, and work fast!!!

Sources & References

Here are some good books and resources that I found useful for doing CSG, the Philip Hubbard paper is almost all you need to read to put CSG into a game engine:

Martti Mantyla, "An Introduction to Solid modeling", 1988

Christoph M. Hoffman, "Geometric and Solid modeling", 1989

Philip M. Hubbard, "Constructive Solid Geometry for Triangulated Polyhedra", 1990 (This paper can be found online, I highly recommend reading it.)

Michael Muuss & Lee Butler, "Combinatorial Solid Geometry, B-Reps, and n-Manifold Geometry" from the book "State of the Art in Computer Graphics", 1991

Back