Procedural Fracturing and Debris Generation for *Kung-Fu Panda*

Lawrence Lee and Nikita Pavlov
DreamWorks Animation

Figure 1: Bridge destruction sequence from Kung-Fu Panda

1. Abstract
Several sequences in *Kung-Fu Panda* called for large-scale destruction that encouraged the debris to be generated by procedural methods but also allowed a high degree of artistic control. In this sketch, we outline such a method for fracturing a model into debris and a system for generating secondary debris when the large pieces collide or break apart.

2. Primary debris
The goal was to allow the user to fracture a model by quickly roughing in the size and shape of the debris pieces, and then procedurally filling in the final shape using the fracture system.

2.1. User input
The system uses a 3D painting program as the primary method of user input because it allows us to match the fracture pattern given by the art director more accurately. Discrete colors are painted on the unfractured model to specify "regions" that correspond to debris pieces. The regions can be painted with as much or as little detail as the situation calls for. Unpainted regions are filled in procedurally in the next step.

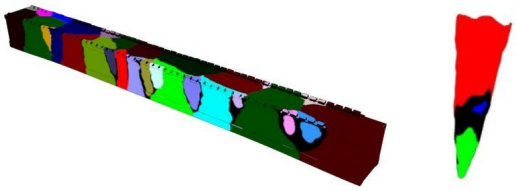

Figure 2: Painted regions for the bridge and a stalactite

2.2. Cutting volume generation
To generate the cutting volumes, we first set up a voxel grid based on the bounding box of the unfractured model. The unfractured surfaces are then voxelized such that the voxels are seeded with the painted color of the surface. Simple rules inspired by Cellular Automata (CA) concepts are used to "grow" the regions until all voxels are filled in. The voxels of each region are then converted to polygon meshes using the marching cube algorithm.

2.3. Final debris piece generation
Finally, the unfractured model is converted to a closed polymesh which inherits the texture coordinates and geometric normals. Intersecting the resulting polymesh with the cutters generated in section 2.2 using constructive solid geometric algorithm (CSG), we arrive at our final debris pieces. Procedural textures based on reference positions and normals are then applied to the newly exposed internal surfaces using triplanar projection. The end result would work regardless of how we fracture the original geometry.

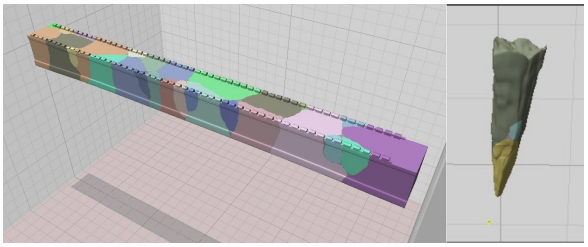

Figure 3. Fractured pieces (color-coded for visualization)

3. Secondary debris
We use the larger debris to procedurally generate smaller chunks of rock and pebbles coming off when these pieces collide and/or crack open.

3.1. Secondary debris from collisions
For any two pieces of debris, we compute the intersecting region, reduce it to the exterior edge, and store that as particle data. We define each point's normal as the average of normals at the intersecting surfaces, storing it in the particles to determine emission direction. The impact force is calculated based on per-vertex velocity of the intersecting surfaces near each particle, which is later used to scale debris emission velocity.

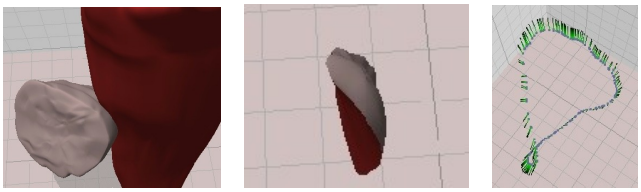

Figure 4: Intersection between two pieces

3.2. Secondary debris from separating pieces
For any two pieces of debris A and B that are close enough to intersect, we build a table that tracks, for every vertex in A, the distance to the closest vertex in B. As soon as this distance exceeds a certain threshold, indicating that the surfaces are separating, we place a particle between the two vertices, its normal facing outward as described above.

4. Additional credits
David Allen, Wes Burian, Saty Raghavachary.