Shattering in Maya

Marcus Lilja
Pelle Serander
Lasse Bergman

April 9, 2018

Abstract

The aim of this project was to create a shattering plug-in for Maya using Python. This report describes how to shatter 3D objects using a Voronoi method and the results obtained, followed by a discussion and future work section.

1 Background

Breaking materials into pieces is a very common process in the visual effects industry. It has a wide range of applications whether it is in explosions, falling debris, cracking materials, collapsing structures, creating dents in materials on impact and more. Although it is possible for users to manually define patterns for cracks along a mesh it is desirable to automatize the process for a less user intensive and more realistic result.

Shattering can either be physically based or non-physically based. Physically based methods are generally more advanced since they simulate reality and therefore rely on complex equations describing material property and similar. However in the VFX industry effects a simulation only need to look *good enough* and not be entirely accurate. Non-physically based methods can either be procedural or image based. Since a physically based method would be too advanced to implement due to time constraints a Voronoi-based procedural method was chosen instead.

2 Method

The chosen method is based on generating cutting points for the mesh. Each pair of cutting points is associated with a cutting plane which is defined using a center point and a rotation angle acquired from the randomized points. The cutting plane cuts a copy of the mesh into a user defined number of pieces and stores these into a *shard group*.

2.1 Collision Check

The implemented script checks for collision between rigid bodies each frame. When at least two rigid bodies collide the shattering process is run. The shattering is only performed on active rigid bodies and does not affect static rigid bodies.

2.2 Enforce Cut Points Inside Mesh

It is important that the points for the cutting plane are located inside of the mesh. If the whole cutting plane is located outside of the mesh no new shard will be created resulting in a runtime error. The cutting points in this project are generated within the bounding box of the mesh. Generating cutting points for more advanced geometry inside its bounding box can be a problem since the points can be outside of the geometry. A check is required to determine if a point is within the mesh. A ray-cast based method was used to perform this check described in algorithm 1. Even if all points are located within the mesh there is a risk that a plane can cut away a shard that exist of two separate shells. If this happens a runtime error will be the thrown since a mesh cannot be constructed by the two separate shells.

```
for all points p do
    shoot ray from $p$ in arbitrary direction.
    count number of ray intersections $n$ with mesh.
    if $n$ mod $2 \neq 0$ then
        return TRUE
    end
    else
        return FALSE
    end
end
```

Algorithm 1: Point inside mesh check

An even amount of intersections means that the ray goes in and out of the geometry the same number of times. Thus the point must be outside of the geometry. An uneven amount of intersections validates that the point is inside.

2.3 Voronoi Shattering

The Voronoi approach that was used is quite simple to implement and understand. It follows the Delauney triangulation which is often used for triangulating a Voronoi diagram according to Scharmen [1]. The general procedure is described in item algorithm 2.

By adding rigid body components to the shards they get affected by Maya's physics simulations. When adding a rigid body its initial velocity is set to zero. If a cannonball comes crashing onto a surface the pieces should inherit a fraction of the cannonballs physical properties. In this implementation the previous velocity of the original object is applied to the shards initial velocity using a damping factor. After applying the rigid bodies the simulation is run.

3 Results

In figure 1 a few frames of a simulation can be seen. In this scene the script was attached to two objects where one of them had an initial velocity.

```
for n number of meshes do
    generate random cutting points $p$ points inside the bounding box
    for all cutting points $p_i$ do
        regenerate if outside of mesh
    end
    for all cutting points $p_i$ do
        duplicate original object
        for all cutting points $p_j$ do
            calculate cut plane center at $p_{center} = (p_j + p_i)/2$.
            calculate angle between $p_j - p_i$ and vector(0,0,1) using python angleBetween to get a perpendicular vector
            create cut plane using center and rotation angle
            perform cut.
            remove shard geometry from temporary mesh.
        end
        add the shard into a shard group.
    end
    add a rigid body to each shard.
    add velocity from original mesh to all shards.
end
```

Algorithm 2: Voronoi Shattering

4 Discussion

The goals of the project were achieved. A Python-based plug-in was created with a graphical user interface. The user can select any number of objects in the scene which have the rigid body components attached to them. The amounts of shards can be chosen by the user.

The damping factor controls how much of the original objects velocity that should be inherited and applied to the shards on collision.

The Voronoi algorithm together with the function that checks if a cut point is within the mesh can shatter many basic manifold meshes as long as two shells are not created. Furthermore the material from the original object will be passed on to the shards after the shattering. However if a texture is used as the

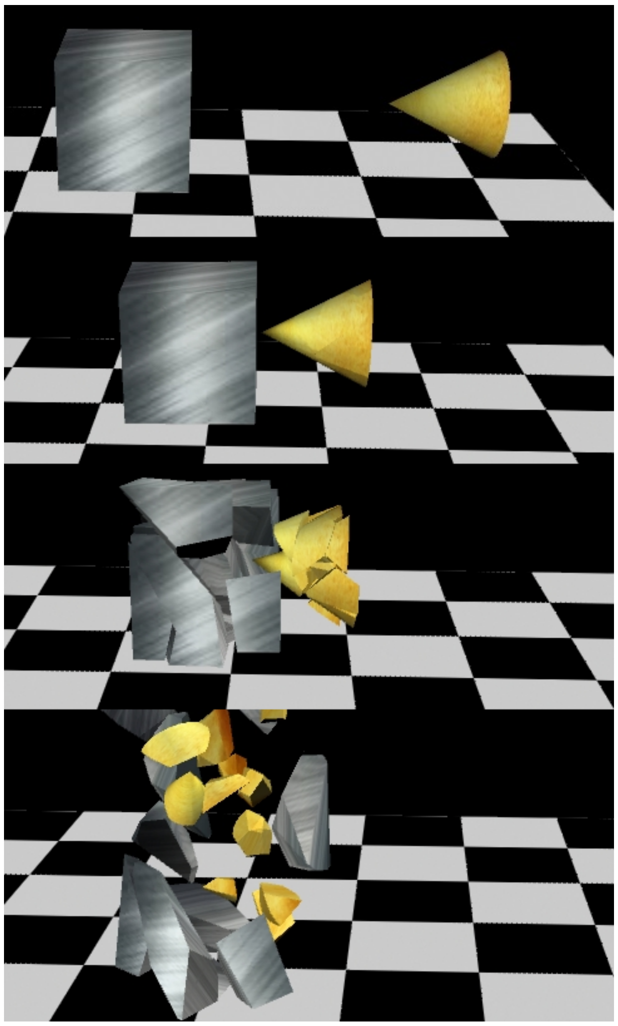

Figure 1: A cone hitting a sphere at rest, the shards from the cone inherits 50% of the original velocity from the parent object

material it will be stretched out on the shards, but it still works and looks decent.

The shattering plug-in implemented works as the project group intended it to and yields a promising result. However it is not completely realistic in some aspects and can still be improved to look better. Nevertheless the group is satisfied with the results and the experiences learned from working with both Maya and Python.

4.1 Technical Challenges

The main challenge that was encountered is how Maya create and handles scene objects. This is mostly due to the group's inexperience with working in the Maya environment. However there are some interesting traits that are not used in other developing tools that you would not expect from a professional developing tool. Other challenges have been working with Python and the Maya API and the limitations this comes with.

One specific problem was that when the object shattered into shards, the shards inherited the rigid body from the original object. However the inherited rigid bodies on the shards were unaffected by the gravity field. Therefore the rigid bodies inherited to the shards had to be deleted and then recreated and connected to the gravity field for it to work as intended.

Another Maya-specific problem was that the simulation was performed each time the timeline was scrolled through.

5 Future Work

There are several things that this plug-in could be extended to do. As of this time it only works for manifold meshes and not 2d surfaces. As mentioned previously there is also a risk of a runtime error if the mesh is constructed in a way that allows more than one shell if it is cut. Moreover it would be preferred to implement an impact force. This would prevent the mesh from shattering at impact if, for example, the fall height wasn't high enough to create the necessary force. Adding this property would also give the user control since they could alter the material solidity. There are also several other traits that are not inherited from the parent object, e.g. the angular velocity. This makes the implementation look a bit unrealistic since the new shards will travel in a straight path after a rotating object has shattered.

The pivot points of the shards are not in the center of the geometry but in the center of its bounding box.

This makes the shards rotate unnaturally. This was done automatically by Maya and could be fixed in the future.

Another thing that the group wanted to implement was to make the object shatter more at the collision area. This would yield a much more realistic result, since it would shatter more at the impact of the collision. The group talked about doing a UV-map or a non-uniform point distribution to solve this, but due to the time constraint this is yet to be implemented.

References

[1] Fred Scharmen. How to: Draw the Voronoi Diagram. 765.blogspot.se/2009/09/how-to-draw-voronoi-diagram.html, 2009.