LINKÖPING UNIVERSITY

TNCG13

# FRACTURING

PROJECT REPORT

Johan ELIASSON johel964@student.liu.se
Kristina ENGSTRÖM krien026@student.liu.se

December 17, 2015

# 1 Introduction

This project is a fracture script for Maya written in Python for creating fractures on objects using a non-physically-based basic procedural method.

## 1.1 Background

When doing fracturing there are two possible approaches, physically-based or non-physically-based[1]. The physical model mimics the real world in fracture behaviour and is commonly used to simulate the real world. For more artistic fracturing the non-physically methods are used. This can be implemented both as image-based models, procedural models and a combination of both.

# 2 Method

The approach is to use a procedural method[1] to split a geometry into sub-meshes is described in pseudocode bellow.

```
getCenterOfGeometry()
for numOfCuts{
    generateRandomPoint(
    centerOfGeometry - threshold,

    centerOfGeometry + threshold)
    cutGeometry(randomPoint,
    randomAngle)
}
separateToSubMeshes()
addRigidBody(submeshes)
```

The algorithm gets the center of the chosen geometry for fracturing. For a user specified number of iterations the algorithm creates a random seed point inside a limited area around the geometry center. From this seed point a cut operation is performed, at a random angle, and creates a new edge trough the original geometry. After all iterations has been completed, an operation splits the original geometry along the created edges into new sub-meshes. In the last step all sub-meshes are given a rigid body to interact with both each other and other objects in the scene, using mayas built-in physics.

This describes the most basic behaviour for the algorithm, which is possible to extended for interaction between seed points to create more realistic fracturing behaviour.

# 3 Result

The result of the method applied on a cube and a plane can been seen in figure 3.1 and 3.2.

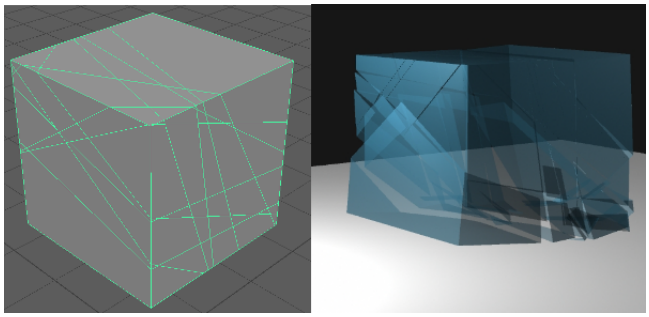

Figure 3.1: Left: Fracturing applied on a cube. Right: Cube rendered with glass material free falling on to the ground.

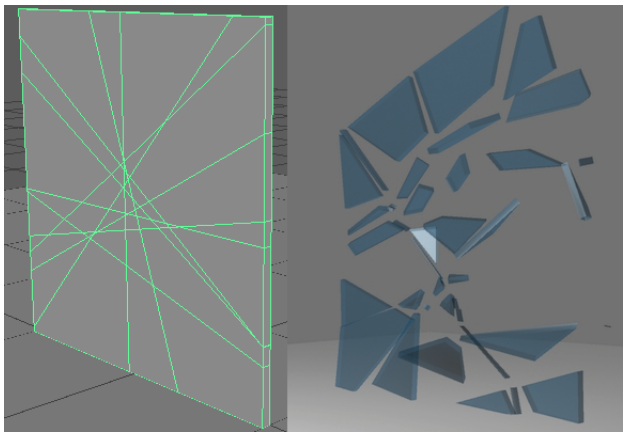

Figure 3.2: Left: Fracturing applied on a extruded plane. Right: Plane rendered with glass material and collided with a bullet.

# 4 Discussion

There has been many challenges during the projects. Much time has been spent trying to solve all these problems and many could have been avoided with more experience in using maya and Python.

## 4.1 Technical challenges

The *polyCut* function splits the object and extract new faces trough the whole object, instead of just making an edge between a seed point and another point or edge. This made a limitation in the appearance of a split as it would always be straight.

Generating faces for the inside of an object after the separation resulted in a lot of problems, which is still unsolved for 3d-geometries. Two approaches to solve the problem has been tested.
The first was to locate border vertices to connect them and generate a new face, this approach was too complicated as no method to determine border vertices was obtained.
The second approach was using the built-in function *closeBorder* to close holes, but the method was unstable and instead of generating a new face it could invert the normals of the geometry instead of generating a new face.

### 4.2 Theoretical challenges

To make a fully procedural algorithm there would be requirements for dependencies between the seed points inside the geometry. In this method each seed point is created without knowledge of the location of the others. This could result in many points being placed close to each other in some places and spread out in other places, without the user having any control of where these hotspots would be located.

## 5 Improvements

Instead of writing the fracturing as a python script is should have been written as a c++ plugin. Both to optimize its performance and also get more more control of the objects structure.

The procedural method could be improved in many ways. The main improvement to be made is to make each seed point aware of each other. This would give the user the possibility to create hotspots and empty regions by letting certain locations and/or points repel and attract other points. To provide better looking results the algorithm should spawn points inside the geometry and then make cuts between the inside points and surface points. This would make the fracturing look more realistic.

To make it simpler for a user to control the algorithm there would be an implemented user interface. This would give more freedom to the user and would not require programing knowledge to change the look of the fracturing.

## 6 Lessons

The choice of making a Python script to Maya was that it sounded like the simplest way. Unfortunately this was not the case. Maya's API for Python limited the project by only letting us use the tools provided in maya. If we would have implemented this as a c++ plugin it would have given us more power to manipulate and create the vertices as we wanted. This would hopefully led to a more realistic and good looking result, both for the cracks of the fracture but also enabled us to create faces on the inside of the geometry.

## References

[1] Lien Muguercia Torres. Fracture Modeling in Computer Graphics. *A thesis submitted in partial fulfilment of the requirements for the degree of Master universitario en Informatica Industrial, Automatica, Computacion y Sistemas*, Advisors: Dr. Gustavo A. Patow and Dr. Carles Bosch, Universitat de Girona, 2011.