A TOOL FOR PROCEDURAL DESTRUCTION IN HOUDINI SHATTERING + DYNAMICS

MASTER'S THESIS

EMANUELE GOFFREDO

N.C.C.A BOURNEMOUTH UNIVERSITY

20th August 2010

Contents

List of Figures

Chapter 1

Introduction

This dissertation describes and outlines the Master Project by Emanuele Goffredo as a completion of the MSc Computer Animation and Visual Effects at Bournemouth University in 2010.

The first aim of the Master Project was to develop a tool to shatter geometries procedurally and in particular applicable in Procedural Building Destructions in the form of a Houdini Digital Asset (HDA). The second aim was the creation of another tool to drive the shattered pieces, using a mix of Particle System and Rigid Body Dynamics simulations.

Chapter 2 places the work in the context of film production, describing problems and solutions which artists have found from the early times to the present with the introduction of Computer Graphics. Some recent academic papers - which have been utilized as a point of departure for the development of the project - are cited and described along with some CG tools available on the market.

Chapter 3 introduces the relevant technical concepts behind the project. In particular Perlin Noise, Particle System and Rigid body Dynamics are described and explained.

Chapter 4 explains the actual implementation of the HDA and the main techniques utilized practically in Houdini for its creation. It also outlines the user interface available. Then it describes some of the problems that affect the application.

Chapter 5 concludes the dissertation highlighting the results of the project and individualizes future work required to improve the tool.

The appendix contains an User Guidelines that explains the main steps to use efficiently the tool.

The Houdini Digital Asset in the form of an otl file, as well as the internal help page and some example files, are included in the DVD handed in with this master's thesis.

Chapter 2

Previous Work

2.1 Shattering and Cinema

Shattering objects with the aim of creating spectacular visual effects, has always been one of the most sought effects in the film productions. The best directors have always dreamed and loved to put some spectacular destruction in their films to thrill their audience. Unfortunately before the born of the Computer Graphics, the type of effects allowed by budgets and practicality were very limited. In the early times these effects were done like practical effects, where big structures of wood, glass or polystyrene were used to create objects, statues and buildings to be destroyed; more recently some effects were created making high realistic small-scale models of real buildings to be blown up and recorded by high speed cameras. Obviously these techniques were and are difficult to direct, to make appear real and difficult to try several time before everything could work in a nice and realistic way; for some of them there is just one shot possibility.
One particular brunch of the shattering in film has been the destruction of buildings. It has been always extremely difficult to create nice and realistic effects in this field since about 10-15 years ago when the Computer Graphics started to be used extensively in big film productions to create always more spectacular effects, especially in the Hollywood high budget film productions.
Examples of shattering buildings are spread in all the history of cinema: from the '30 with King Kong [1933] through the '50 and '60 with the Hollywood Historical epics like Samson and Delilah [1949] with the spectacular destruction of the temple of Dagon in the final scene or the Sword-and-sandal films like The Colossus of Rhodes [1961], passing for the '70 with Superman [1978] and its epic destruction of Krypton to the more recent Blockbuster movies like Independence Day [1996], Armageddon [1998], The Lord of the Ring: The Return of the King [2003], The Day After Tomorrow [2004] or Inception [2010].

2.2 Shattering and CGI

Since the dawn of CGI, the first half-artists computer technicians that started to create a new world in the computer screen, developed systems to produce realistic effects with the aim of reproducing and simulateing Earth happenings. One of this effects has been the simulation of shattering things. Many papers have been published and many work has been produced up to today [Norton et al. 1991; O'Brien and Hodgins 1999; O'Brien and Hodgins 2002; Martinet 2004].
The first big problem that the artists understood was the impossibility of creating "natural" effects through a manual design. It would be extremely difficult to create the movement and the behaviour of a glass that shatters on the floor or to simulate the destruction caused by the explosion of a bomb in a building, using traditional CG animation tools like key-framing and handmade modeling. Many of the solutions explored in the official papers have been useful to create tools to help artists in the creation of this effects, but often the research of realism has developed too complex solutions difficultly applicable in a real production environment. Norton et al. [1991] have been one of the first to produce a nice animation of fracture through pioneer CG techniques that have put the basis for many successive works like the two of O'Brien [1999][2002] where more sophisticated physics-driven simulations were produced through the use of more complex maths techniques like the Finite Element Discretization. These and many others papers have developed high realistic techniques but their biggest fault is the fact that in their original formulation they can be used difficultly in the production of films.
On the other hand other papers like Martinet [2004] have been thought with the aim of creating a pleasant visual effect, not physics driven, more usable and more efficient. More recently, a different trend has also developed; because of the more and more power computers make us available and because of the application of more efficient techniques, some nice real-time physics-driven effects have been used in some Blockbusters films, like the ones by Ron Fedkiw and his team [Bao et al. 2007] that were used in "Terminator3: Rise of the Machines", "StarWars: Episode III Revenge of the Sith" and "Poseidon".

2.3 Shattering in Real Production Environments

Usually, applying 100% realistic shattering techniques in a real Pipeline is an hard work and when it is possible artists try to avoid that and try to find some technical ploy that works visually nice. Besides, often a 100% simulation of the reality is simply not necessary and because of this, the art of finding new creative solutions has heavily been used in many of the last Blockbusters films. Accordingly to Zalzala [2004] - speaking about a scene from "The Day After Tomorrow" - : "Because of the scope of destruction and need for fast turnarounds it was decided not to use practical models or costly rigid and soft body deformations. Instead a versatile system based on procedural animation

techniques was developed to handle everything from geometry breakup to destruction animation."
Why to produce something 100% real when a clever technique would be unnoticeable to any human eye watching it at about 30fps?
"Cheating" the reality in this contest is just the need of producing something nice, fast and customizable in the way that the audience could feel it like real.

2.4 Artistic Solutions

Examples of this trend are [Zalzala 2004] and [Scheepers and Whittock 2006], where the use of creativity and cunning, more than Maths and Physics has produced realistic effects.
Zalzala [2004] created a nice shattering buildings effect transforming into a noise space a simple building facade, cutting along a certain height, and repositioning the original pieces back to their initial positions. In this way the facade appeared nicely cut, ready to be shattered.

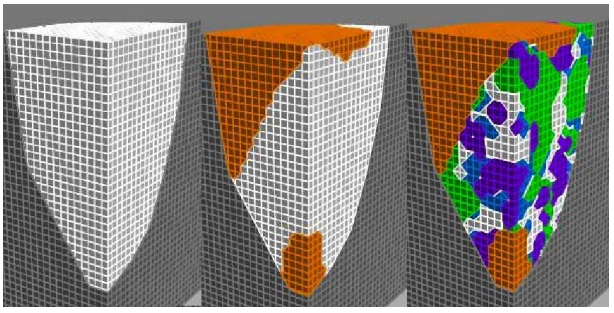

FIGURE 2.1: Geometry Cut with Frequency Noises (The Day After Tomorrow)

Scheepers and Whittock [2006] used a Voronoi technique to cut the pieces in a road destruction and then decided to animate these chunks bounding them to a light and a customizable particle system.

FIGURE 2.2: Wrecked Road in the film Cars. Disney / Pixar.

Significant is the part in [Scheepers and Whittock 2006] where the artists say : "Note that this approach does not handle collisions, an inconsequential limitation given the frenetic camera motion, low lighting, and concomitant chaos during the wrecking process. We provided a mechanism to reposition tiles as a post process, so gross intersections could be corrected". So no Rigid bodies simulation, no complex behaviours, no 100% realism, but at the end a pleasant visual effect.

2.5 The Tools Available

In many of the 3D packages like Houdini, Maya and 3D Studio Max are available some tools to shatter objects with various techniques, and when the out of box tools are not enough many plugins have been produced by artists and TDs like iShatter for Maya or Procutter for 3DSMax. Houdini offered, till the version 10, a shatter and a break tool, and in the version 11 (out on the market just a couple of weeks before the hand in of this dissertation) it has been implemented also a Voronoi shatterer. Often these shattering tools work quite fine, they are versatile and can produce nice shattering. But on the other hand, once the shatter has been produced the next step is the one to apply a Rigid Body Dynamics system in order to produce some kind of effects.

The main motivation behind my project is the lack of an easy tool to shatter and animate the shattering in a efficient and easy way especially in the simulation of Building Destruction, giving the artist a specific tool and not a generic one.

Chapter 3

Technical background

This chapter describes the technical aspects and the techniques used to implement the master's project.

3.1 Houdini

It was decided to create a tool as similar as possible to the ones created during the production of films and in particular to create a procedural tool. Among all the things considered, it was chosen to create a Houdini Digital Asset. Houdini was taught in the MSc Computer Animation and Visual Effects.
This 3D package is gaining every year more importance in the CG industry and it is more and more used in the big film productions because of its procedural nature and its work-flow, that simplifies the pipelines; like Framestore VFX Artist Guillaume Fradin says in the production of Avatar [2010] "Houdinis flexibility was a big help and most of the solutions we ended up were solved with out-of-box Houdini nodes without too much scripting,".
Also Digital Domain creates many of their own tools with Houdini, to achieve effects that will be repeated throughout a film, for his "100% procedural" nature and like Thomas Reppen, CG Effects Animation Lead, explains, Houdini gives the possibility to package up the tools in Houdini Digital Assets, allowing all the artist to use them easily [2009]. This procedural nature and the way in which Houdini has been developed have been the reasons for which it has been chosen for this final project.
To create a similar tool in Maya it would have been produced in MEL or more likely in C++ in form of a plugin. On the other hand the node system nature of Houdini makes itself a kind of visual programming language where the use of Hscript (Houdini's general purpose scripting language) and Python can extend its possibility.

3.2 Perlin Noise

Noise, and in particular Gradient Noise was firstly implemented by Perlin (1985) and Perlin and Hoffert (1989) [Ebert 2003]. Perlin noise usually and in particular in Houdini is implemented like a function that can receive like parameter a 4D or 3D vector, or a couple of floating numbers or a single floating number. The result can be a floating number or a vector of floating numbers that vary pseudo-randomly throughout a N-dimensional space. The algorithm starts from a set of precalculated gradient vectors to create the pseudo-random value. In Houdini the function is called *noise()* and the output value is in the range 0-1 and it is non-periodic, the computational cost is the lowest compared with the others noises available in the 3D package (Worley noise, Sparse Convolution noise and Alligator noise) [Documentation Houdini 9.5, Noise VEX function]. The Perlin Noise can be used in the creation of random pattern for the creation of "realistic stochastic natural texture" [Perlin 1985] or in general any time it has to be recreated a "natural randomness".

FIGURE 3.1: Perlin Noise Texture Marble

3.3 Particle System

The concept of Particle System appeared for the first time in CG for the production of "Star Trek II: The Wrath of Khan" in 1982. Reeves [1982] introduced the system like "a method for modeling fuzzy objects such as fire, clouds, and water".
Objects are modelled like a cloud of primitive particles (points) and are generated, moved and let die from the system.

Particle systems are used for simulating smoke, liquids, fire but also hair and fur or falling objects like leaves or petals.
In Houdini, Particle System is well developed and allows the artist to create many kinds of effects. The basic set up is formed by an emitter of particles that creates them, sets up their life length and their initial behaviour, and some forces that make them behave in some kind of specific way. This simulation is located in the POP Network.

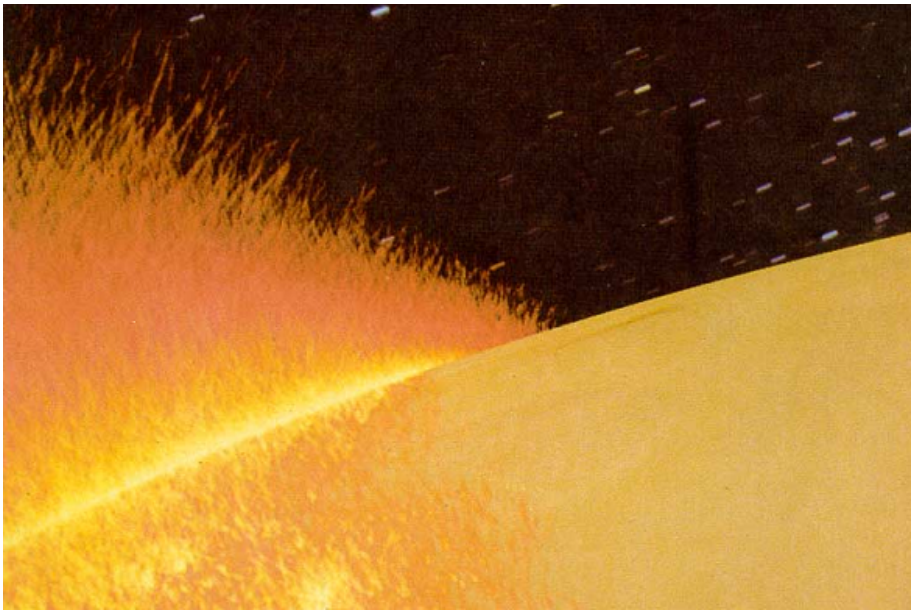

FIGURE 3.2: Particle System in Star Trek II: The Wrath of Khan

3.4 Rigid Body Dynamics

CGI simulation of rigid bodies uses Newtonian laws to simulate the behaviour that 3D objects would have in the real world, where objects interact, colliding, bouncing or sliding on each other. In CG the normal behaviour of objects is the one which allows the different geometries to penetrate and then pass through other geometries.
So, to simulate a quite realistic behaviour of some objects in a CGI simulation, two are the main prerequisites:

1) There has to be a way in which the object can detect the occurrence of any kind of collision with any other object in the scene

2) There has to be a way in which the object can react appropriately to the collision detected

This class of simulations is called Rigid Body Simulation where rigid bodies are non deformable objects that cannot inter-penetrate [Baraff 1989] and are subject to forces like gravity or external forces.

In Houdini this simulation is called DOP Network and allow to set up many effects and many options like forces, physics or collisions detections.

3.5 Development Environment

The Houdini Digital Asset was developed in Houdini 10.0.595 for Windows. It has been also tried on the same version for Linux. Apparently it seems working in the same way on both the systems.
The Digital Asset has been created in a way that each node has got a commenting name and so it is quite easy to follow the different operation just reading the nodes flow. Besides, the main group operations were put in colourized boxes to identify them easier. The Digital Asset is provided with a help that explains all the functions and parameters. Also some examples are provided to show the digital asset in operation.

Chapter 4

Implementation

The main idea for the Master Project was to create a tool for the simulation of procedural destruction and in particular for Procedural Building Destruction. The aim of the tool was the creation of visually nice effects of background or middle-ground where the destruction of a building, total or partial, was due to different causes, like for example an internal or external explosion. The main motivation behind the project was the lack of an easy and specific tool in the main 3D packages, to shatter and animate shattering in a efficient and easy way especially in the simulation of Building Destruction.
The system was initially thought to be divided in two main parts: the shatterer and the dynamics manager. The inspiration for the first part came from a paper called "Procedural Building Destruction for 'The Day After Tomorrow'" by Jens Zalzala [2004] for the Digital Domain company, where they created a modular system to destroy buildings hit by hurricanes. For the second part it was thought to use a mix of rigid body dynamics and geometries bounded to a particle system, to make it as light as possible; for this second technique it was taken some inspiration from the paper "The Wrecked Road in Cars" [2006] by Pixar. The particle system part was thought to drive the light structures, like broken glasses. The rigid body dynamics, on the other hand, to drive the motion of the heavy structures of the buildings. In the final design the Rigid Body was implemented but disabled for the final users for reasons that will be explained later.

4.1 Basic Idea

The basic idea behind the tool is this one: most of the buildings we are used to see everyday and that are built every year in our cities are essentially a set of walls, floors and roofs, which can be considered more easily like a set of extruded grids. Often buildings have got others things like external structures or doors, but when they are seen not so close, or maybe in the background, it is likely to ignore all these other things.

So the main idea was to find a way to shatter a simple 2D grid and then with an extrusion create a 3D shattered object. This idea was improved like explained next.
The other main idea is that often, seeing footages of real shattering buildings it is very difficult to see exactly what is happening. We can see pieces and chunks falling down, rotating and shattering, but then we see dust, maybe fire or even an explosion. It is not really possible to understand 100% all the interaction among the shattered chunks. So the second main idea was to drive the chunk with a "clever" and light particle system instead of a heavy and slow RBD system.

4.2 The Shatterer Tool

The HDA was called Shatterer Tool and internally it is divided into two main parts: the shatterer and the dynamics manager.

4.2.1 Shatterer: Main Idea

Internally the Shatterer implements a basic technique that could be summarized with this formula: noise and cut. Basically, the aim is to shatter a basic grid in a random, irregular and more natural possible way. It was Based on the Zalzala work [2004] presented to ACM SIGGRAPH 2004. Zalzala was also contacted by email for more details about his work.

So, the idea consists in taking a grid, that doesn't have to be necessarily a square, it can be any 2D shape and can also contain holes, and then try to have this primitive as clean as possible, in the sense that shouldn't have many internal points or edges.

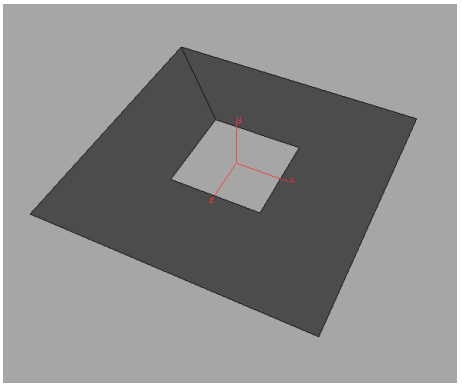

FIGURE 4.1: Initial Primitive

Once that, the 2D shape is cut in two parts: the one that has to be shattered and one that has to remain whole.

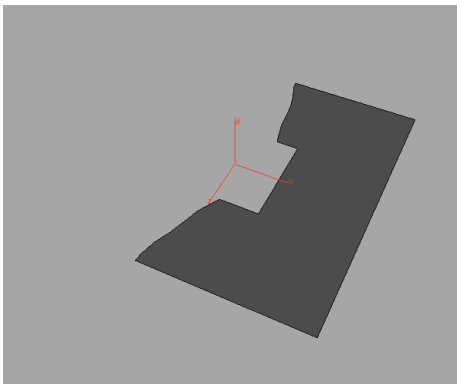

FIGURE 4.2: Initial Primitive Part to be Shattered

Then the shattering part is bricked, that means that some points are spread internally on the surface.

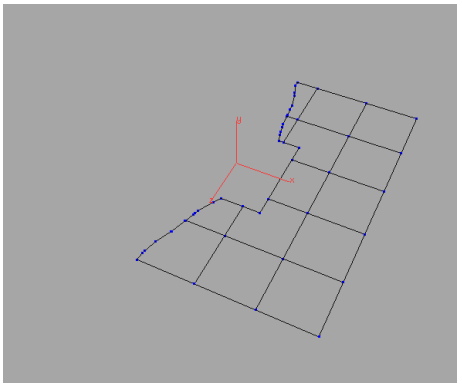

FIGURE 4.3: Initial Primitive Bricked

Then the position of each point is stored and saved; next a noise function is applied to the position of each point and their high is transformed corresponding the noise value.

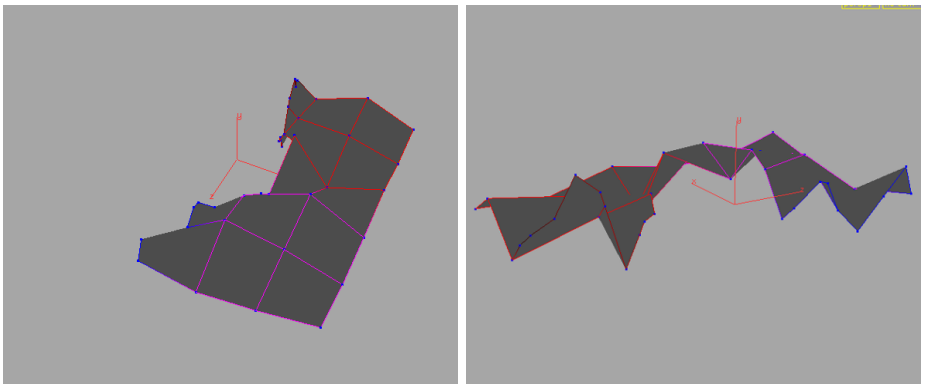

FIGURE 4.4: Initial Primitive with Noise Applied on its Points

Then, a cut is performed on the centre of this new noisy primitive and this produces two different new noisy irregular primitives. Then the position of each point is restored to its original value. This produces two different 2D plain irregular groups of primitives.

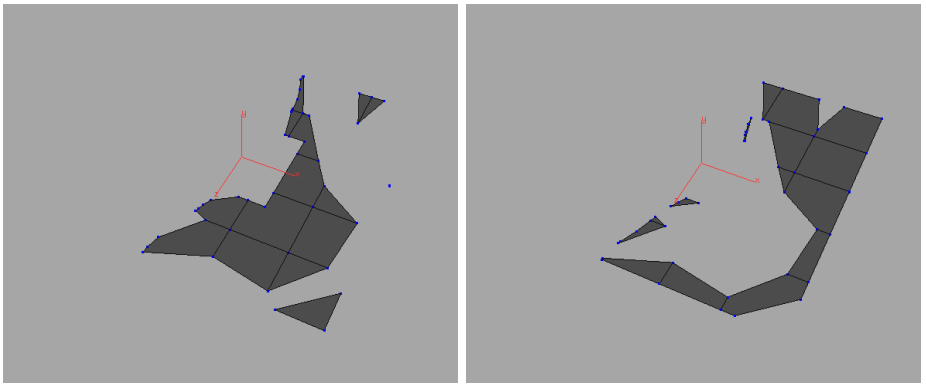

FIGURE 4.5: Two 2D Plain Irregular Groups of Primitives

Then one of the two groups of primitives is used like a cookie cutter for the original clean primitive, and so the result is the original primitive with some nice irregular cuts.

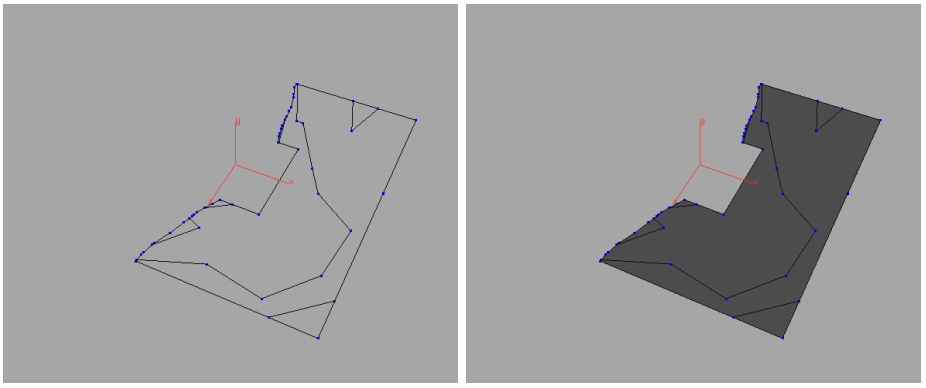

FIGURE 4.6: The Original Primitive with Some Nice Irregular Cuts

This operation can be performed many times producing some nice irregular patterns.

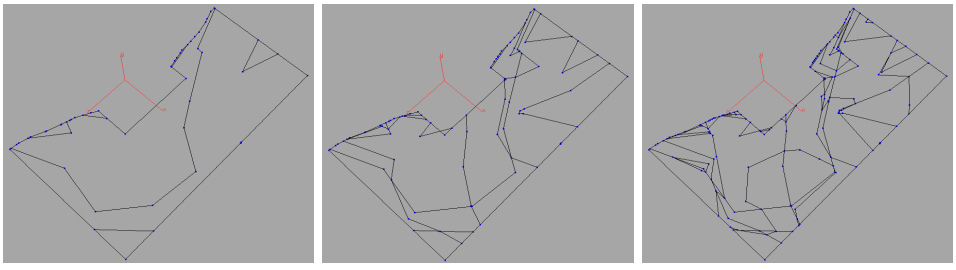

FIGURE 4.7: Operation Performed 1,2 and 3 times

4.2.1.1 Shatterer: Houdini Implementation

This technique has been implemented in Houdini in this specific way:

- The Shatterer takes as input one or more groups, where each group has to be a 2D primitive; it can be a grid or a holed grid or a set of grids.

- A 3D or 2D geometry is used to cut each of the initial groups to determinate the part that would be shattered and the one that would not. This cut is performed with a cookie sop.

- The shattering part of each group is treated with the technique explained before. The bricking operation is performed with the bricker option in the divide sop. Then the original point position is saved in the point colour channel. The point position is changed accordingly with the *noise()* function. For example the Y channel is shown below.

```
{
if($NY!=0,
(
if(ch("../../../SHATTERER_CONTROLLER/distribution_direction")==0,
 (noise
   (
      ($TX+0.1)*stamp("..","FORVALUEN",0),
      ($TY+0.1)*stamp("..","FORVALUEN",0),
      ($TZ+0.1)*stamp("..","FORVALUEN",0)
  )*
   stamp("..","FORVALUEN",0)*100000
 ),

 (
     if(ch("../../../SHATTERER_CONTROLLER/distribution_direction")==1,

       (noise
       (
          (($TX+0.1)*stamp("..","FORVALUEN",0))*ch("../../../SHATTERER_CONTROLLER/distribution_value"),
          ($TY+0.1)*stamp("..","FORVALUEN",0),
          (($TZ+0.1)*stamp("..","FORVALUEN",0))*ch("../../../SHATTERER_CONTROLLER/distribution_value")
      )*stamp("..","FORVALUEN",0)*100000),

       (noise
     (
          ($TX+0.1)*stamp("..","FORVALUEN",0),
          (($TY+0.1)*stamp("..","FORVALUEN",0))*ch("../../../SHATTERER_CONTROLLER/distribution_value"),
          ($TZ+0.1)*stamp("..","FORVALUEN",0)
     )*stamp("..","FORVALUEN",0)*100000)

     )
 )
 )
),
$TY)}
```

- The channels where the noise is applied are the ones with the normal not equal to zero. The noise function takes as parameters the point position. The noise function result is multiplied for a big value, because practical tests have shown better result. Then in the case the distribution of the noise has to follow an horizontal or vertical pattern a multiplier is used like in the example before.

- A clip operation is performed to cut the noisy primitive.

- The original point positions are restored recalling back the colour channel where were stored.

- One of the two resulting group is extruded and used as a cookie cutter on the original not bricked primitive.

- Once obtained the 2D shattered primitive, it is partitioned so that each piece is a group.

- Each group is extruded in the opposite direction of the original primitive normal and is colourized randomly in a different way to make it more visible during the setup. This extrusion was fine to simulate thin objects like glass, but once used to simulate walls it appeared quite unnatural. For this reason was added the option to perform an internal cut of each chunk made by a randomly rotating irregular plane, that divides the chunk and makes appear it more realistic.

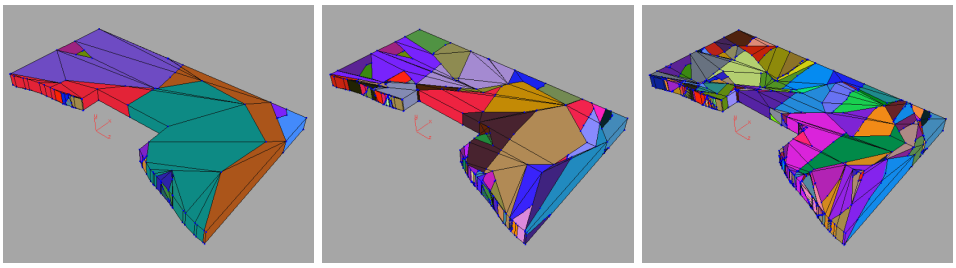

FIGURE 4.8: Operation Performed with Level of Detail 1,2 and 3

4.2.1.2 Shatterer: User Interface

The shatterer part of the digital asset is controllable by the UI that has been created with the standard Houdini digital asset parameters creation.

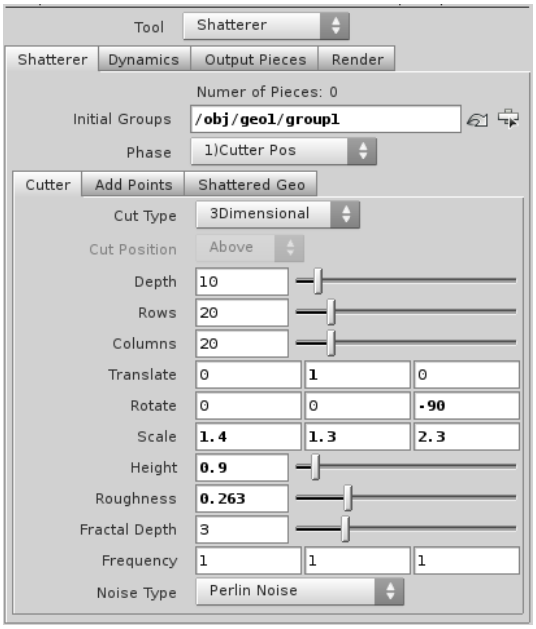

FIGURE 4.9: Shatterer User Interface: Cutter Position

Once selected on the very top the Shatterer option, the Shatterer folder is activated. It is divided in three parts: "Cutter Positioning", "Add Points" and "Shattered Geo". The Shatterer receive in input one or more groups in form of an imported sop.
In "Cutter Positioning", it is possible to choose between a 2D or 3D cutter, its position, its shape, its depth, the noise that affect it, its rotation and size.

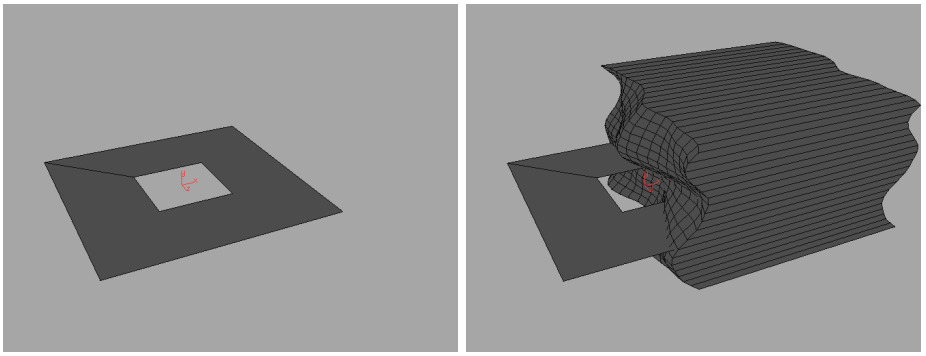

FIGURE 4.10: Cutter Positioning

Then when "Add Points" folder is activated, it is possible to choose the quantity of points to put on the surface. The number of points will affect the next step result. The offset applies a translation of the new points.

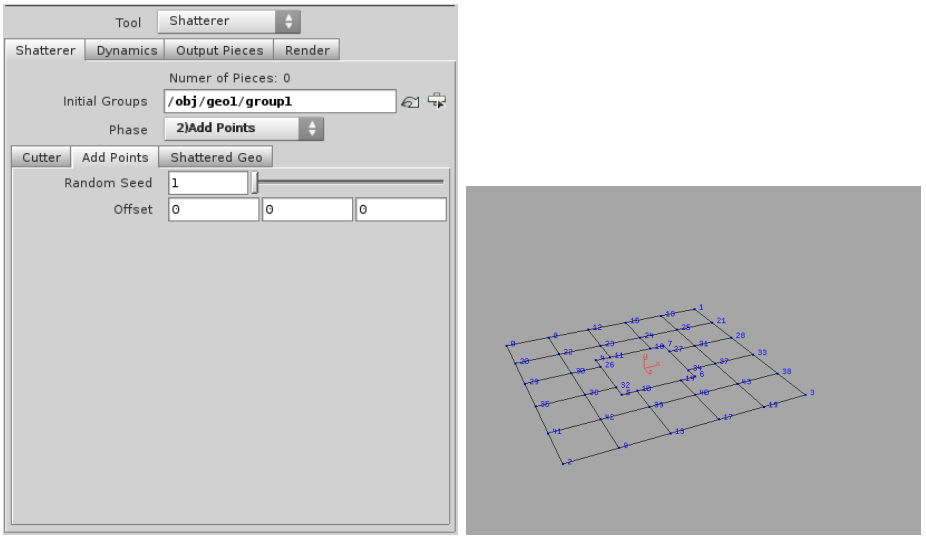

FIGURE 4.11: Shatterer User Interface: Add Points

The "Shattered Geo" folder allows to set up the level of details of the shattering, that means a bigger value corresponds to a bigger number of chunks created; then it is possible to choose the distribution of the cuts, if the chunks has to be colourized, the thickness of the chunks, if they have to be subdivided internally, and if the structure is like a glass or a bigger structure like a building (this option will be used later to perform a self collision detection).

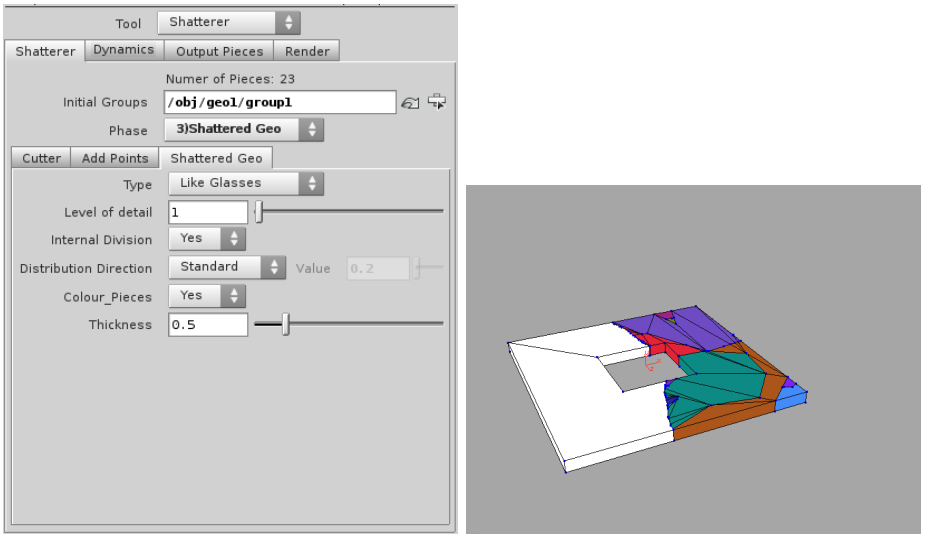

FIGURE 4.12: Shatterer User Interface: Colourized Shattered Geo

4.2.2 Shattering Dynamics Manager: Main Idea

Internally the Shattering Dinamics Manager implements a technique that could be summarized with this formula: chunks and particle system. Basically, the aim is to drive the motion of a shattering object, bounding each chunk to a particle of a particle system. It was based on the idea explained by Scheepers and Whittock [2006] presented to ACM SIGGRAPH 2006.
Thus, the idea consists in taking the shattered object, bound each chunk to a particle in a particle system, drive this particle system in a clever way, animating indirectly the chunks that will behave in a realistic way.

4.2.2.1 Shattering Dynamics Manager: Houdini Implementation

In Houdini it has been developed in this way:

- First of all, in each initial group that was shattered, the internal shattered pieces are reordered in a precise order.

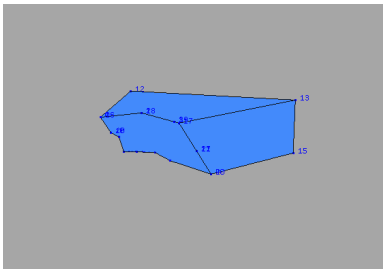

FIGURE 4.13: Chunk to be Animated

- Then with an "Add" sop for each shattered piece, is created a net of wires between all the points of each chunk.

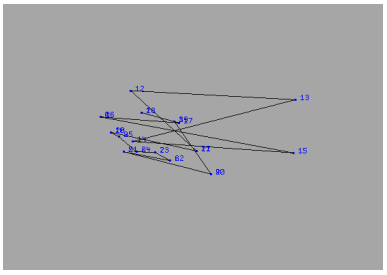

FIGURE 4.14: Wired Chunk

- Then for each shattered chunk is calculated the barycentre with the "fuse" sop

So now there are 3 main groups: one reordered group, one group of wired chunks and one group of barycentre points.

The barycentres group is the one from which the particle system is created through a POP Network. Then the particles are bound to the wired groups with a "Primitive" sop, where the source groups are the wired groups and the template group is the one coming from the POP Network. Then the wired groups are bounded as well to the original chunks with the "Point" sop, where the original pieces match the wired groups according to the points numbers.

The POP Network has as inputs two sources. The first one is formed by the barycentre of the shattered chunks and the second by the barycentre of the whole not shattering pieces of geometries. To these particles are added three attribute: a charge, a radius and a mass. Then two "Force" nodes are connected, one for the gravity force and one for the initial impulse of the motion. Then two "Collision" nodes are used to manage the collision with the ground and the self collision. A rotation is applied to the particles in the direction perpendicular to the motion vector. At the end all the particles are affected by an "Interact" node that makes them repel each other. The still particles don't have any particular motion but contribute the the repulsion. The repulsion option is the one that simulate the RBD system. In fact, the only way in which particles can simulate a RBD is having a charge and a radius of interaction. When two particles with enough charge have their zone of interaction in contact they tend to repel. In this way dosing radius and charge is possible to create a kind of zone of repulsion that simulate what happens in a RBD.
The POP Network can work in two modes: auto and manual.
The auto mode makes the particles move all together and all in the same direction. Each particle inherit an UP vector that is the normal vector of the original relative chunk. This UP vector is used to set the initial force that gives the initial acceleration to each particle. The manual mode can set up different chunks with different forces. It works on the concept explained by Zalzala [2004] where different groups of chunks are controlled by "bombs" which are possible to position and whose timing is possible to be adjusted. Boxes represent these "bombs" and through one or more of these, it is possible to select the chunks to be directed manually. A rotating Pyramid over the box indicates the direction of the motion of the particles. If not all the chunks are selected by the box, the one not selected won't move during the simulation,will be considered still and not affected by any force.

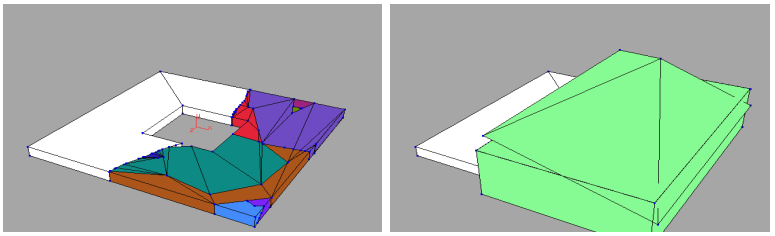

FIGURE 4.15: Positioning of the Bomb in Manual Mode

The boxes cannot overlap each other, otherwise problems could come out.
In both the modes it is possible to decide at what frame the shattering has to start.

4.2.2.2 Shattering Dynamics Manager: User Interface

Once selected "Dynamics" on the top of the UI, it is possible to setup many options. The first one allows the user to simulate or to preview the shattering. The preview is created with a line for each particle that is drawn on the screen, and shows the path that each chunk will do on the shattering.
The simulation mode is the actual simulation where the chunks move and rotate.

The first folder is the Physics one and allows to set up the gravity force value and the mass value for each chunk. The mass is just a multiplier of another value that depends on the size of each chunk.

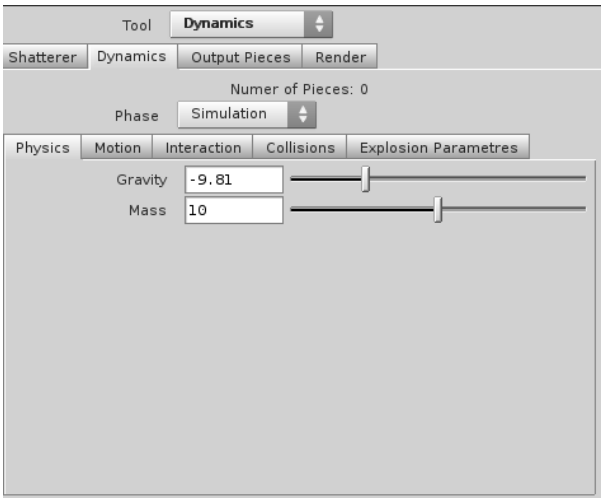

FIGURE 4.16: Shattering Dynamics UI: Physics

The second set of option is for the "Motion". Here is possible to set up the initial impulsive acceleration, if there is any rotation, how fast is that, at what frame it has to start and has to finish and after how many bounces on the collision ground has to stop. The start and end parameters depend on the actual start of the motion of the particles. If it has been set up the start of the motion at the frame x and the rotation at the frame 3, the rotation will start at the frame x+3. These options are cumulative, the first that is validated annuls the others.

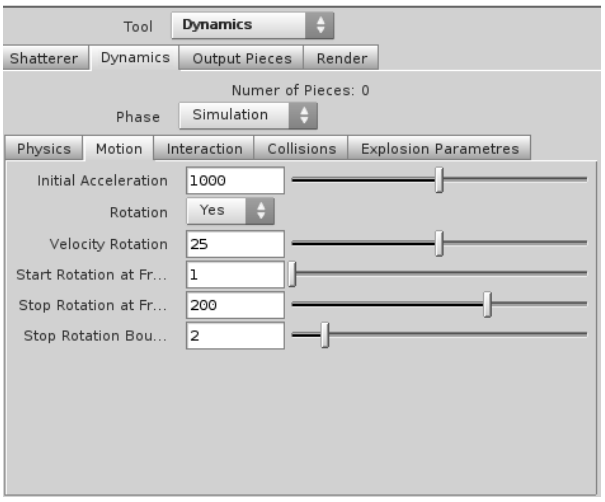

FIGURE 4.17: Shattering Dynamics UI: Motion

The "Interaction" options are "Drag Piece" that regulate how much the particles are dragged by their mass. Then the "Piece Interaction", the "Level of Interaction" and the "Acceleration Interaction" regulate the quantity of force the particles acquire in the repulsion phase with the other particles and what initial acceleration they have.

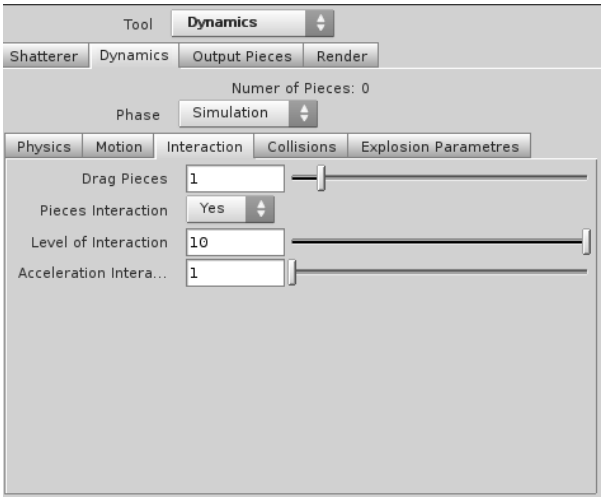

FIGURE 4.18: Shattering Dynamics UI: Interaction

The "Collisions" folder set up the ground collision, importing any sop. The tolerance of the ground is the height at which the particles react over the actual ground, resolving the problem of penetration in the ground. The self collision is activable but works only if in the shatterer folder has been chosen the type of object to shatter to be like a big structure . Then it is possible to set up the bounce gain normal and the bounce gain tangent that regulate how much the particles will bounce on the normal and tangent direction respectively. After the last bounce, the particle can stop or slide on the surface.

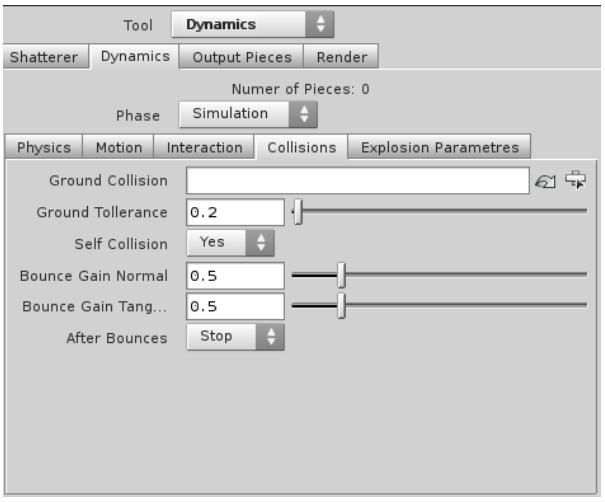

FIGURE 4.19: Shattering Dynamics UI: Collisions

The "Explosion Parameters" is the section where is possible to set up how the particles will move. In the auto mode they will follow the normal direction of the original chunks. In the manual mode there are two phases. The Positioning is the one in which one or more boxes are placed over the chunks; each box has got a rotating pyramid that determinates the direction of the explosion. The second one is the "Done" position where the setup is activated and the operations performed. Both the auto and the manual mode can set up the start frame of the "Explosion".

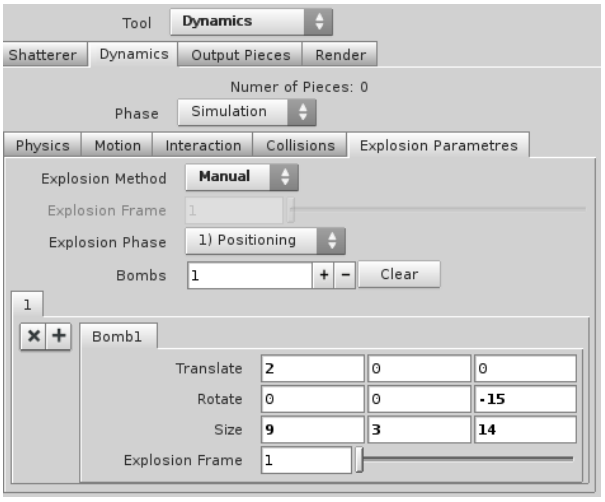

FIGURE 4.20: Shattering Dynamics UI: Explosions Parameters

4.2.3 Output Shattered Pieces and Render

Two more options are available in the top folders tab: The first one is the "Output Pieces". In the case the user wants to use just the shatterer tool and obtain just the

chunks with no dynamics, that is possible choosing this option . This operation bypasses the dynamics operations and allows the user to choose between 3 different outputs: shattered pieces, whole pieces or both.

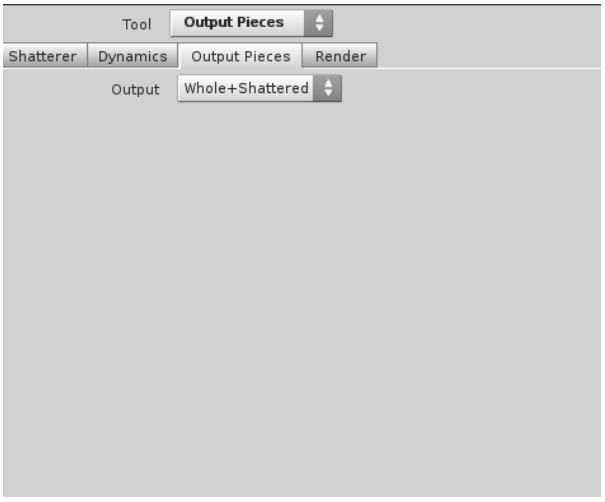

FIGURE 4.21: Output Pieces UI

The second option is "Render" and helps the user to render the shattered object. The "Render Mode" is a useful tool because it allows the user to have the original objects until the shattering starts. If the shattering has been set up at frame x, the shattered pieces won't appear until frame x, and that works also when different shattering timing is added to different groups. "Normals" set up the normals of the object. "Original" doesn't change the original normal of the initial grid. "No Normal" deletes all the normals and "Random" creates normals with random directions. With the last option "Main Crack" is possible to decide if visualizing or not the crack between the shattered part and the whole part.

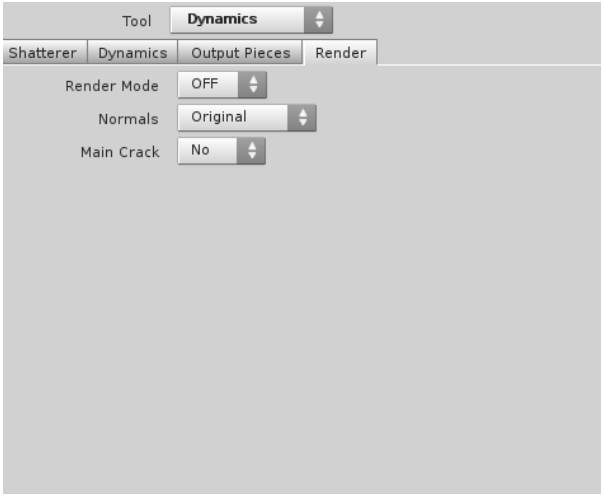

FIGURE 4.22: Render UI

4.3 Creation, Testing and Debugging

The creation of the tool has been an iterative process where details were added to an initial core of basic functions. Great importance have had the phase of testing and debugging.
Using the tool to create different scenes and examples has been useful for understanding what were the options necessary to develop an usable and working tool. Often when a lack of control was underlined by the use, that control was added and more and more controls and options were added iteratively, simulating the relation developer-artist where the former tries to grant the latter's wishes. The practical testing has also underlined many bugs not visible during the development. In particular because of the procedural nature of the tool, it has not been easy to create a tool that works with any piece of geometry and for this, continuous solutions have been developed.

4.4 Problems and Bugs

The Shatterer Tool, if used with knowledge of how it works, is a good and efficient tool. As explained before, many testing has cleaned the tool from the majority of the problems.
Unfortunately it was impossible to solve completely some of the problems. In particular the ones reletive to the "Cookie" sop in Houdini. This sop is a very complex and powerful node but sometimes it simply doesn't work as expected. It has been used in particular to divide the initial grid in two big pieces, one to be shattered, and one to remain whole. Especially when it tries to cut strange grids, usually with internal holes, it can produce some artefacts. In this case the user has to have the patience of tweaking the position of the main cutter until found a good and working position.
Another problem is due to the complexity of the internal operations performed by the Shatterer. The operation of shattering is recursive. For this reason, if a too high detail value is put on the options, the application can crash or can freeze Houdini. The solution to this problem is to use the options with moderation, starting always with small values, that most of the times give good results.
Another problem has been encountered during the shattering operation. Because each input group adds of one the times that the internal "Foreach" sop has to be calculated, if too many groups are added the tool could crash.
The majority of the bugs concerns the use of memory and CPU that can freeze in infinite cycles.
In the main idea had been considered the use of RBD besides the particle system to move the chunks. An RBD System has been developed internally but at the end it has been decided to deactivate it from the options of the final user. The reasons of this decisions, have been that some problems have been met during the creation of the HDA,

that seemed to find some incompatibility with the internal DOP Network; but more important has been the consideration that, allowing the user to have the shattered chunks like output, it can develop an ad hoc RBD system more accurately than any possible internal RBD system.
Some working tests have been produced applying an external RBD to the shattered objects. The only problem is that when the HDA is installed, the shelf tool to automatically create an RBD doesn't work. For this reason the RBD system has to be set up by hand.

To sum up, the main problem lies in the fact that the tool can make Houdini crash, especially during the shatterer phase. But once understood the reasons for which it crashes - like the use of a too high number of points during the "Add points" operation or the too high value of the details - it easy to make it work stable. Once passed on the dynamics phase the tool becomes almost 100% stable.

4.5 Cosiderations: Performance, User-friendliness and UI

Considering all above, the tool is easily usable and the tests handed in with this thesis show this. The efficiency was an important prerequisite and it has been achieved. The shattering operation of one or more grids in a good amount of pieces (500-700), takes from 1 to some tens of second on a medium-high performance laptop (Intel Core2 Duo 2.20Ghz). It has been tested successfully with until 30 different primitive groups as input, working fine. The shattering dynamics works fine too. It has never made the tool crash. To perform the initial operation, it takes a time proportional to the number of pieces to be moved. It is a particle system, so it is quite fast, but some other operations make it a little bit slower. In the worst of the test, once the tool had performed the initial operation in a time varying from few seconds to a couple of minutes, it can show the result in the viewport - in the worst of the cases - with the velocity of one frame per second. The worst case has considered a number of pieces equal to 1000-1500. An equivalent operation in RBD has taken tens of minutes or hours. The normal case of a good amount of pieces, in order to obtain a good visual effect, has been created in minutes, a part from the time to set up the scene.
The other prerequisite was the user-friendliness. Everything has been done to create an intuitive tool, easy to set up in minutes. The use of simple but effective techniques has also been important. In particular the operation of selections of the part to be cut, or what chunks to move in a particular way, are managed placing a box in the right position. Simple idea but easy to understand and to use.
As far as the User Interface is concerned, it is developed with the standard HDA parameters tool. Unfortunately at first sight it can seem a little bit confusing, but once understood the logic behind it, is not difficult to follow the work flow, top-down, left-right.

Chapter 5

Conclusions

The aim of the project was to create a tool to easily set up a procedural destruction, in particular a procedural building destruction to be used in middleground or background on big destruction scenes, in a real film or short film production. The project achieved this objectives. In particular the tool developed to shatter objects can be used no only in the context of the buildings destruction but can also be used for any kind of shattering, where the object to shatter has got a regular 3D "like a box" shape. The shattered object can be used with the internal dynamics manager, very useful in the case of creating procedural building destruction or similar effects, or with a separate RBD simulation to be set up by hand. Performances and efficiency are good and usability is high, like shown by the examples created.
Using the tool properly allows to create a nice effect in minutes.
The only thing not respected on the very first design was the inclusion of an internal RBD system for the reasons explained on "Problems and Bugs" chapter.
The tool is not problem-free but the majority of the problems concerns the use of some internal Houdini Sop that produce sometimes and in some situations strange behaviours.

The user-friendliness of the tool is only limited by a non optimum user interface, but once understood its logic, this little hurdle can be easily overcome.
This is a common problem in the creation of an HDA where there is the lack of a proper way to create a nice and highly customizable UI in Houdini. It was considered to create a python interface, but because of the lack of time it was decided to reserve it for the future work list, giving the priority to other aspects.

The first possible future work would start from this issue: the creation of an adequate User Interface in Python.
Another possible future work would be the integration of other external objects (e.s. ledges or doors) in the internal dynamics, so that it could drive also their motion. Then the self collisions option, not working really fine, should be improved.
Another feature that should be added is the creation of some more pre setted options

apart from the explosion one, like the one to simulate the effect of an earthquake or a tornado on buildings.

To conclude the Shatterer Tool is first of all a specific tool to shatter simple 3D objects or more complex objects sub-dividable in a set of simpler objects, like happens in buildings. The second feature is the possibility to decide if animate the shattered object with the internal light Particle System tool or an external RBD system.
It has been thought for building destruction but it can be used in many other destruction examples.
It can be considered a good tool to be used in big destruction scenes, accompanied and integrated with other effects.

Appendix A

User Guidelines

The user that wants to develop a scene with the Shatterer Tool has to follow some easy rules to make it work properly.

- Once the HDA has been imported one or more grid have to be placed with in mind how the final design of the object has to be. For example, if a simple building has to be shattered, four grids has to form the main external walls, then can be added more for the floors and the roof and the internal walls.

- Each of these grids has to be as much clean as possible (no internal points or edges).

- The grids can have internal holes, but this could produce some random problems.

- Each grid has to be grouped. If more then one grid, for example the grids that form the windows of a singular facade, lie on the same plane, they can be grouped all together, otherwise no.

- The design of the object has to consider that the extrusion is calculated on the opposite direction of the original normal of each grid.

- If some of the grids are not part of the possible destruction, for example a wall opposite to the shattering part, it is advised to merge it after the HDA node. Each group more imported in the HDA will increase the computational time of the shattering operation.

- The user is invited to use in moderation the options especially during the shattering, because many recursive operations could cause problems (see “Problems and Bugs” Chapter). In particular in the “Add Points” and “Level of Details” is better to start from small numbers and try to increase one number per time.

- If no good results come out even changing the “Add Points” and “Level of Details”, it is suggested to increase the size of the initial 2D group

- One general rule is to save quite frequently especially during the shattering operations, because Houdini could crash quite frequently too.

- During the dynamics phase, if the number of chunks is more than 700 it can take a couple of minutes to cook the operations, depending on the power of the machine.

- Once found the best motion or to find it, it is suggested to save the simulation like a series of bgeo file on disk.

- To create irregular cuts with half broken natural chunks, not select all the chunks during the "Positioning" phase in manual mode but just some.

- In the "Positioning" phase in manual mode do not overlap the various boxes.

Bibliography

Armageddon, 1998. Directed by Bay M. USA: Touchstone Pictures.

Bao, Z., Hong, J., Teran, J., Fedkiw, R., 2007. Fracturing Rigid Materials. *IEEE Transactions on Visualization and Computer Graphics*, 13 (2), 370-378.

Baraff, D., 1989. Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies. In: *International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 1989*, Ithaca, NY, USA. 23 (3), 223-232

Ebert, D.S., Musgrave, F. K., Peachey, D., Perlin, K., Worley, S., 2003. *Texturing and Modeling: A Procedural Approach.* 3rd ed. USA. Morgan Kaufmann

Inception, 2010. Directed by Nolan C. USA: Warner Bros. Pictures.

Independence Day, 1996. Directed by Roland Emmerich. USA: 20th Century Fox

King Kong, 1933. Directed by Cooper M.C., Schoedsack E.B. USA: RKO Radio Pictures.

Martinet A., Galin E., Desbenoit B., Akkouche S., 2004. Procedural Modeling of Cracks and Fractures. In: *Proceedings of the Shape Modeling International*, Genova, Italy. Washington DC: IEEE Computer Society, 346 - 349.

Norton, A., Turk, G., Bacon, B., Gerth, J., Sweeney, P., 1991. Animation of fracture by physical modeling. *The Visual Computer*, 7 (4), 210-219.

O'Brien, J. and Hodgins, J., 1999. Graphical Modeling and Animation of Brittle Fracture. In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, Los Angeles, USA. New York: ACM Press/Addison-Wesley Publishing Co.

O'Brien, J., Bargteil, A., and Hodgins, J., 2002. Graphical Modeling and Animation of Ductile Fracture. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, San Antonio, Texas, USA.New York: ACM.

Perlin, K, 1985. An Image Synthesizer. In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, New York, USA.

REEVES, W. T., 1983. Particle Systems A Technique for Modeling a Class of Fuzzy Objects. In: *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, Detroit, USA.

Samson and Delilah, 1949. Directed by DeMille C.B. USA: Paramount Pictures.

SideFX Houdini, 2010. *Framestore Avatar.*
http://www.sidefx.com/index.php?option=com$_{c}ontenttask = viewid = 1694Itemid = 68[Accessed10July2010]

SideFX Houdini, 2009. *G.I. Joe: The Rise of Cobra.*
http://www.sidefx.com/index.php?option=com$_{c}ontenttask = viewid = 1572Itemid = 68[Accessed10July2010]

SideFX Houdini. Documentation. *Noise VEX function.*
Houdini 9.5. http://www.sidefx.com/docs/houdini9.5/vex/functions/noise [Accessed 15 July 2010]

Sidefx. *Video tutorials*, 2009 - 2010.
http://www.sidefx.com/tutorials [Accessed 3 July 2010]

SCHEEPERS, F., WHITTOCK A., 2006. The Wrecked Road in Cars - or How to Damage Perfectly Good Geometry. In: *International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 2006 Sketches*, Boston, Massachusetts, USA. Article No.: 97.

Superman, 1978. Directed by Donner R. USA: Warner Bros.

The Colossus of Rhodes, 1961. Directed by Leone S. Italy: Procusa.

The Day After Tomorrow, 2004. Directed by Emmerich R. USA: 20th Century Fox

The Lord of the Ring: The Return of the King, 2003. Directed by Jackson P. USA: New Line Cinema

ZALZALA J., 2004. Procedural Building Destruction for The Day After Tomorrow In: *International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 2004 Sketches*, Los Angeles, California, USA. 144.