Bachelor Thesis

Simulating Building/Monuments Aging using Procedural Generation

Nearchos Nearchou

University of Cyprus

Computer Science Department

December 2021

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

APPROVAL PAGE

Bachelor of Science Thesis

Simulating Building/Monuments Aging using Procedural Generation

Presented by Nearchos Nearchou

Reasearch Supervisor _____

Andreas Aristidou

University of Cyprus

December, 2021

Acknowledgements

I would like to express my deep gratitude to my thesis supervisor, Assistant Professor Andreas Aristidou at the Department of Computer Science of the University of Cyprus for his expert advice and encouragement throughout this difficult project.

I would also like to thank Christos Othonos for sharing this inspiring project with me and provided me with all the necessary knowledge in order to get a complete image of the project. He was always accessible to answer my queries and clear up any facts I needed to evaluate.

I would also like to thank my friends who endured this long process with me, always offering advice and unconditional love.

Finally, I would like to thank my family for the support and encouragement they showed in this very intense academic year.

Abstract

Procedural Generation plays an important role in modern computer graphics and design. For that creation of buildings, procedural construction is key. Although there are several tools for creating good-looking structures, many of them lack a realistic and artistic touch. Structural degradation by weathering effects and real-world occurrences must be taken into consideration to make them appear more authentic. This project illustrates an implementation, which uses the CGA Shape Grammar to design a structure, the grid model to store several attributes of the model, and a variety of techniques and tools to illustrate how buildings and various materials age.

Table of Contents

Acknowledgements **3**

Abstract **4**

Table of Contents **5**

Chapter 1 Introduction **7**

1.1 Motivation 7

1.2 Contributions 8

1.3 Objective 9

Chapter 2 Literature Review **10**

2.1 Procedural Modeling of Buildings 10

2.2 L-system 11

2.3 CGA Shape Grammar 11

2.4 Building Aging 14

Chapter 3 Realistic Aging Effects & Materials **16**

3.1 How Different Materials age 16

3.2 Aging Effects 19

Chapter 4 Data Acquisition and Tools **23**

4.1 Houdini 23

4.2 CGA tools 27

4.2.1 CGA Pipeline 30

Chapter 5 Methodology **34**

5.1 Overview 34

5.2 Aging Simulation 36

5.3 Apply Aging 38

Chapter 6 Results 43

6.1 Introduction 43

6.2 Procedural Damage 44

6.3 Procedural Humidity 45

6.4 Procedural Wind Resistance 46

6.5 Procedural with Beams 47

6.6 Limitations 49

Chapter 7 Conclusions & Future Work 51

7.1 Conclusion 51

7.2 Future Work 52

Chapter 1

Introduction

Contents

1.1 Motivation

1.2 Contributions

1.3 Objective

1.1 Motivation

Computer graphics necessitates the use of technology. They are becoming a widespread feature in user interfaces and commercial motion films on television. Today's computer visuals are vastly different from those of the past. Artists and architects now have an interactive user interface that allows them to manipulate the structure of an item using a variety of input devices.

In films and video games, graphic designers seek to convey their creative vision and evoke feelings within the audience. As a result, the film and gaming industries aim to achieve distinctive outcomes by hiring a large number of artists and spending a significant amount of money and resources. Therefore, procedural generation emerges to assist businesses in avoiding costly outcomes.

Procedural generation is the process where the computer creates the most needed materials using only the necessary resources. As a result, a lot of time and money can be saved to increase the efficiency and the development of a big-scale project.

Furthermore, procedural generation in buildings has been increasingly popular in the last decade, and individuals are experimenting with new ways to portray their structures. However, many procedural construction approaches have significant flaws, such us their generators, which are quite sluggish, and the data intake is enormous. These limitations create an opportunity and encourage scientists to examine the use of building generation for video games, with a particular focus on inventing an algorithm that reduces the amount of mesh data created for each building and the generation time.

In terms of the results that computers generate, most projects lack a realistic look, since they generate “excellent” outputs according to the data set given. In contrast real life, where people live in houses that have flaws due to the passage of time and weather conditions, and this is what the game users and spectators want to see. As previously stated, individuals struggle with modelling rather than displaying the impact from ageing and weather conditions. They want to handle their large input data in order to provide larger outcomes, which are what the majority of consumers really want to view; results.

The goal of this solution is to develop outcomes that are fairly similar to those seen in the actual world, not flawless but still pleasing to the eye. The aim is to handle as much data as possible in order to get high-resolution, low-complexity outcomes.

1.2 Contributions

The following are the thesis' key contributions:

- Use the software SideFX Houdini, employ façade operations with the CGA Shape Grammar, simulate the model procedurally, and apply aging to it.
- Use of CGA Grammar Shape to split all of the building's facades and assign each façade

to a distinct group.

- Introduce a procedural method for assigning attributes to each point of the construction by reading files.
- Present weathering effects utilizing aging tools/techniques that are similar to how structures age in the actual world.
- Demonstrate alternative ways for how a structure may behave rather than target the simulation’s efficiency.
- Introduce various building materials and how they mature. Some materials are stiffer than others, and the effect that a material may have varies.

1.3 Objective

The project's desired result is to implement a system that generates age effects for buildings in a procedural manner. The purpose is to create a tool to aid graphic designers, as well as architects, to achieve more accuracy when creating historic structures or imagining how a building would age. Furthermore, the approach may be beneficial to the gaming industry. When a project requires the construction of a new or deterioration of an existing structure, a large number of graphic designers and programmers work tirelessly to get the desired outcome. With the aid of procedural generation, a lot of time and resources are saved, allowing graphic artists to focus all of their attention and creativity on the minor details, resulting in a considerably better product. Finally, the technology would allow the artists to use their talent to produce fascinating aging effects and demonstrate their abilities.

Chapter 2

Literature Review

Contents

2.1 Procedural Modeling of Buildings

2.2 L-System

2.3 CGA Shape Grammar

2.4 Building Aging

2.1 Procedural Modeling of Buildings

Parish and Muller (2001) employed procedural modeling to construct CityEngine, a software tool that was first developed at ETH Zurich and eventually bought by ESRI. CityEngine is a commercial multi-platform 3D modeling program that was created to create 3D urban landscapes. The procedural method to modeling efficiently is the central principle of CityEngine (Müller, Zeng, Wonka and Van Gool, 2007). The pipeline includes numerous procedural modeling tools for creating large-scale urban layouts as well as CGA rules for generating complex structures (Müller et al., 2006). In summary, this method uses a rule-based approach to provide a rapid and easy way to develop urban design scenarios. The issue with this strategy, as a result of the mix of

different types, would be to alter parameters to adapt to current city conditions.

2.2 L-system

Muller et al. (2006) used the Lindenmayer systems or L-systems which they were outlined by the Hungarian scientist and botanist Aristid Lindenmayer in 1968 who needed to portray plant development (Müller et al., 2006). The exhibit of astounding success based on the utilization of design grammars had fundamental impact on procedural methods. In spite of the fact that L-systems are not specifically pertinent for this goal, the notion of grammar and some of the rules (scale, translate, rotate) were embraced for Procedural Modeling Buildings (Müller et al., 2006).

L-systems follow a straightforward consecutive structure which can be characterized by three fundamental parts: “V” which is an alphabet of the system, an axiom “ω” as a string of symbols and a finite set of production rules “P” which characterize the way in which the factors modify themselves (Wonka, Wimmer, Sillion and Ribarsky, 2003).

2.3 CGA Shape Grammar

The scripting language that is used for the generation of entities in CityEngine is a Computer Generated Architecture (CGA) shape grammar (Müller et al., 2006). CGA is based on the L-system data structure which uses shapes rather symbols.

CGA shape grammar is characterized by:

- a set of geometries or shapes,
- a set of geometry/shape operations,
- a set of rules,
- the initial shape.

The following sections will provide a brief overview of each one of the CGA shape grammar characteristics.

Shape

A shape is the foremost imperative concept of CGA shape grammar.

The properties of a shape are:

- the pivot, which is defined by a root and direction point,
- the bounding box's scope, which is determined by translation, rotation, and size
- the geometry

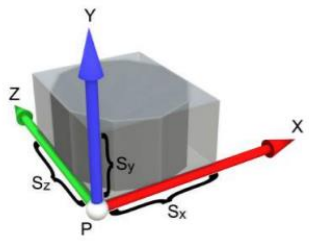


Figure 2.8: main attribute of a CGA shape. CGA shape grammar shape definition, from (Müller et al., 2006). The pivot is P, the scope is S, and a sample shape is presented within the scope.

Rules

The alteration of shapes is done using generic principles, similar to L-systems. All CGA rules have the format

Predecessor --> Successor.

As can be seen from the figure above, the predecessor (left side of arrow) must be replaced by the successor (right side of arrow). A set of shape operations are employed to define the model's new shape after Successor.

The examination of how CityEngine divides a block into four storeys using a 2D polygon will be explained in the following example (Klippel, n.d.):

```
Lot --> extrude (10) Envelope

Envelope --> split(y) {~2.5: Floor} *
```

The first rule mandates that the 2D form "Lot" must be substituted by a 3D shape "Envelope", which is created by extruding the 2D polygon by 10 units. The second rule states that each Envelope created by the first rule shall be divided (y axis) into four 2.5-height Floor forms, for a total of four floors (Klippel, n.d.).

Operations

The project employs general operations to alter forms in the same way as L-systems do. The following are the key operations:

- insertion: adds geometry and/or a texture to the scene. These are architectural components of structures, such as a windows, doors, or beams.
- basic graphics transformations: translate, scale, rotate.

- subdivision split: Splits the current form into numerous shapes. Splits are ubiquitous in architecture. The repeat splits, following specific rules.
- component split: breaks a form into smaller components, such as faces, lines, or points. This is helpful in design to isolate a facade from the building mass model, for example. A mass model is a simplified model that depicts the main structure of a structure while omitting specifics.

CGA example

Figure 2.9 displays a basic illustration of a facade's breakdown. A CGA Shape Grammar may readily capture this breakdown.

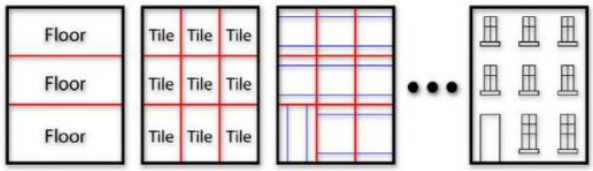


Figure 2.9: Deterioration of the facade explained in paper (Muller et al. 2007).

2.4 Building Aging

Gutierrez and Sheffer introduced a variety of tools and strategies for presenting building aging by

employing weathering processes and aging occurrences in a variety of materials, including mudbrick and rocks (Gutierrez and Sheffer, 2015). The models created and described in this paper are based on their method, and they have been designed to include their ageing representation. This is achieved by modifying the FEA mesh for a specified time frame to simulate aging (Gutierrez and Sheffer, 2015). Assuming aging values for each grid point and keeping track of the damages as time passes, when the value is 0, it implies it has been entirely destroyed, however when the value is 1 it suggests it has not been completely destroyed (Gutierrez and Sheffer, 2015). According to the aging values, the FEA mesh is extruded and molded. The purpose is to move the aged data into the procedural pipeline so that the building may be completed. Each surface is represented differently once the data has been transmitted. For example, the ceiling is being pushed inwards as the walls are being peeled. Another thing to keep in mind is that each material has a distinct stiffness. Mudbrick, on the other hand, is more susceptible to erosion and slow weathering, which results to incremental and, in many cases, smooth destruction (Gutierrez and Sheffer, 2015).

After reviewing the main theories and research that has been conducted Procedural Modeling of Buildings, CGA shape grammar, L-Systems and Building aging, the theoretical framework behind this project is established. Following this, more information about how the theories were implemented in the project will be provided.

Chapter 3

Realistic Aging Effects & Materials

Contents

3.1 How Different Materials age

3.2 Aging Effects

3.1 How Different Materials age

This section of the document will address the aging of various construction materials. The goal is to see how all of these materials evolve over time so that data is included in the solution.

Mudbrick

A mudbrick, also known as a mud-brick is the oldest and most often used in construction material. It's an air-dried brick constructed of loam, mud, sand, and water, with a binding substance like rice husks or straw (The Oxford Companion to Archaeology, 1997). Mudbricks have been around since 9000 BCE, but bricks have been burned since 4000 BC to strengthen their strength and longevity (The Oxford Companion to Archaeology, 1997). In figure 3.1 can be seen an example of how a mudbrick ages.

Figure 3.1: Left image is example of mudbrick in houses and the right image is how it ages.

Wood

Wood is mostly employed in the floor construction, although it may also be found in the walls. Wood has played a vital part in the history of civilization. Humans have utilized it for a variety of purposes, including fuel, building materials, furniture, paper, tools, and weaponry (D'Costa, 2015). Figure 3.2 shows the effect of an aged wood.

Figure 3.2: Left image represent a healthy wood and the right side an aged wood

Concrete

Concrete is a long-lasting and adaptable construction material that may be found all over the world (6 Popular Uses for Concrete - Ozinga, 2013). It's used on some of the most unusual concrete structures in the world, and it can be seen everywhere in nowadays (6 Popular Uses for Concrete - Ozinga, 2013). Figure 3.3 shows the crack effect that is being created from an old concrete.

Figure 3.3: Stone aging representation

Stone

Stone is another significant construction element. Stones can be used for flooring, roofing, brickwork, paving roads, and as aggregates in concrete, depending on the kind (Types of Building Stones and their uses, 2019). Natural stones are used to construct the majority of prehistoric structures because they are more solid throughout time (Types of Building Stones and their uses, 2019). An example of how a stone age can be seen in figure 3.4.

Figure 3.4: Stone aging effect

3.2 Aging Effects

Cracks

Cracks are usually visible in walls rather than roofs, especially on mudbrick and concrete materials. It's possible that it will show up on other materials, but it's not very common, especially on stones that don't have a binding substance. Figure 3.5 depicts a few additional examples.

Figure 3.5: Crack effect on houses

Tiles Collapsing

When the tile absorbs too much water then the outer layer isn't adequately protecting it. As a result, if moisture is absorbed, the tile swells, expands, and creates pressure. As a consequence, the tile stands out, and in many circumstances, since tiles act as a support system for other tiles, they all follow suit. Figure 3.6 further shows how the tile collapsing action frequently results in holes in the structure.

Figure 3.6: Representation of tiles collapsing

Peeling

Many houses have an outside layer that conceals the construction material (wood, concrete, mudbrick, etc.). However, if proper care is not performed, that layer will peel away, revealing the substance beneath.

Figure 3.7: Peeling effect on houses

Chapter 4

Data Acquisition and Tools

Contents

4.1 Houdini

4.2 CGA tools

4.2.1 CGA Pipeline

4.1 Houdini

SideFX Houdini it’s a 3D tool that procedurally allows its artists use the software for the film and game industry. Houdini's procedural character comes from the ability to store nodes and transfer the data quickly and effectively to other attributes, allowing its users to use them in different forms without starting over; and that’s what makes Houdini extremely flexible. Houdini is also famous for its Visual Effects. Having a strong toolset allows its users to explore as much as possible the animation and procedural modeling (FX Features | SideFX, n.d.).

Bellow follows a list of the essential tools and features that Houdini will provide for this solution.

Geometry

Geometry in Houdini is built up of primitives and the most frequent are the polygon faces. A vertex is a point on a polygonal face. The difference between a vertex and a point is that a point can be shared by associated primitives however vertices are unique to each primitive (Geometry functions, n.d.).

Attributes

Houdini uses attributes as values. Point attributes include things like color, location, UV coordinates, weight and normal (Geometry attributes, n.d.). Attributes are being stored as primitives, vertices, points, or as an entire geometry (referred to as the "detail" level) (Figure 4.1) (Geometry attributes, n.d.).

All these attributes can be utilized for a number of things such as changing the shape of the geometry.

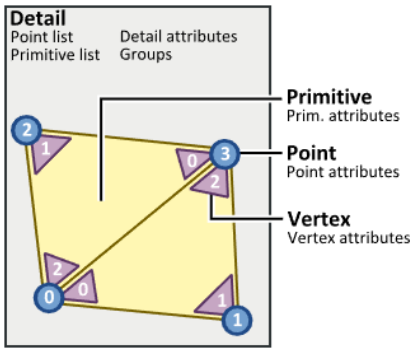


Figure 4.1: A deeper understanding of an attribute in Houdini

Operators

The procedural essence of Houdini can be found in its operators. There are several operators in Houdini but only the relevant ones for this application will be discussed. The most important are the Surface Operators (SOPs) which these are used to construct and modify geometry usually for procedural modeling (Houdini (software) - Wikipedia, 2006). Dynamic Operators (DOPs) are solvers that are used to construct simulations, Vector Operators are for altering geometry, Shading Operators represent a shader that will be used on geometry and finally Composite Operators are utilized to perform compositing on recordings (Houdini (software) - Wikipedia, 2006).

Script

For its scripts Houdini uses Python or Hscript which does not manipulate geometry. The use of the scripts is mainly for doing difficult computations and it is also some sort of way to collaborate with Houdini IDE (Python Script, n.d.). Figure 4.2 illustrates an example, where python is used to read some files and pass values to the primitives depends on the file content.

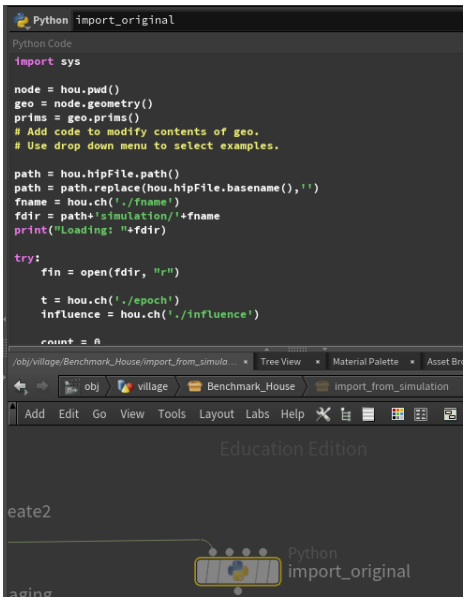


Figure 4.2: Python script

Groups

Groups refers to a grouping of points or faces. To create a group, the user must give the name, a type (primitives, points, edges, vertices) and finally a function (replacing, union etc.) (Figure 4.3). Something fascinating about Houdini is that when a point is eliminated then Houdini removes it from all the groups that it was being used in. There are two types of point groups: ordered and unordered. Ordered groups stock their points in selection order and unordered groups stock their points in creation order (Groups, n.d.).

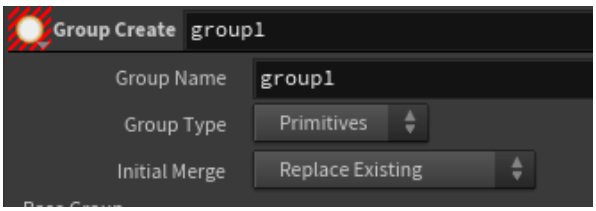


Figure 4.3: Creating a group

Delete

The delete node is capable of deleting data as well as complete simulation objects (Delete, n.d.). Just like the Group node the user selects the type they want to delete (primitives, points etc.) and they can also choose to keep them instead of deleting them. This comes in handy when creating interesting objects. Figure 4.4 show the Delete Node in Houdini

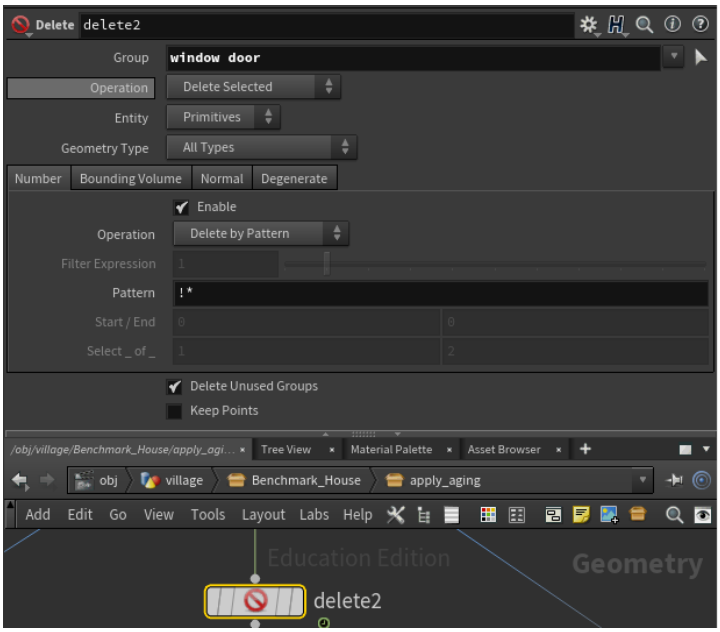


Figure 4.4: Delete Node

4.2 CGA tools

The following section will cover different CGA Shape Grammar tools that were used in SideFx Houdini. Those tools that were implemented do not exactly follow the CGA Shape Grammar but they use some basic operations like façade split.

CGA Shape Subdiv Façade

Subdiv Façade is a digital object that simulates the CGA Shape facade splitting rule. Its major goal is to separate the facades into numerous groups and give each one a name. This adds additional complexity to the model and will aid the user in determining where certain things should be put in the future (in which facade).

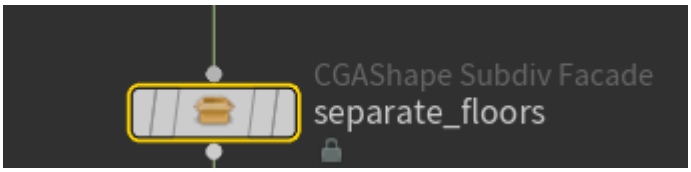


Figure 4.5: Subdiv Façade Node

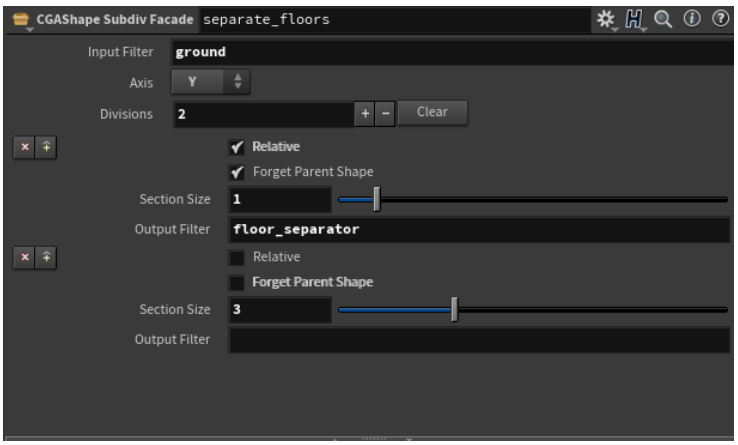


Figure 4.6: Interface of Subdiv Façade Node

CGA Shape Repeat

The CGA Repeat Node follows the Repeat rule of CGA. In Figure 4.7 the Repeat Node is

represented. The user has to decide how many subdivisions wants and then input it in the correct field (Figure 4.8).

The Repeat function is very similar to the Subdiv Façade function with the major difference to be the manner in which the computations are carried out.

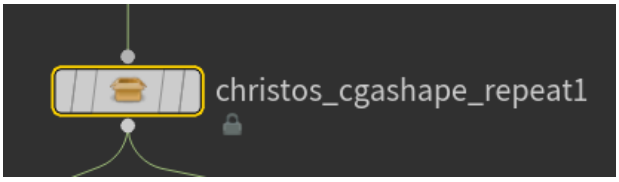


Figure 4.7: Repeat Node

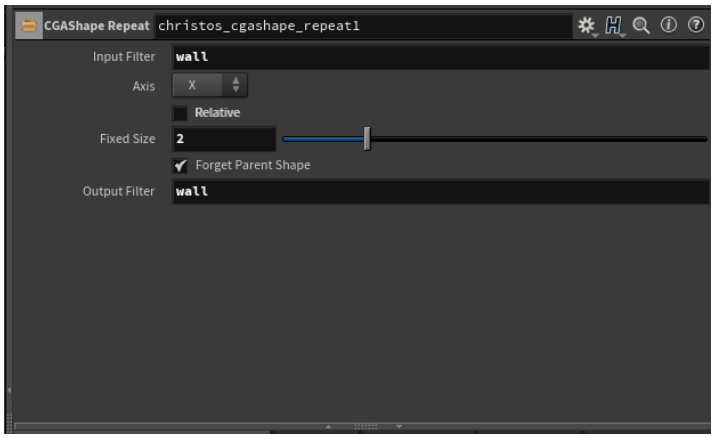


Figure 4.8: Interface of Repeat Node

CGA Shape Insert

Just like the Repeat Node which follows the repeat rule the Insert Node follows the Insertion rule of CGA Shape Grammar. Figure 4.9 shows that the node need 2 inputs which are the geometry and the model. Applying insert function allows us to add in the example (figure 4.10) the windows, however the window will be added in the group windows where subdiv façade took place. Basic graphics transformations (translate, scale, rotate) are used in order to place the windows in their exact places.

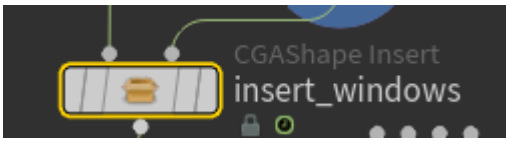


Figure 4.9: Insert Node

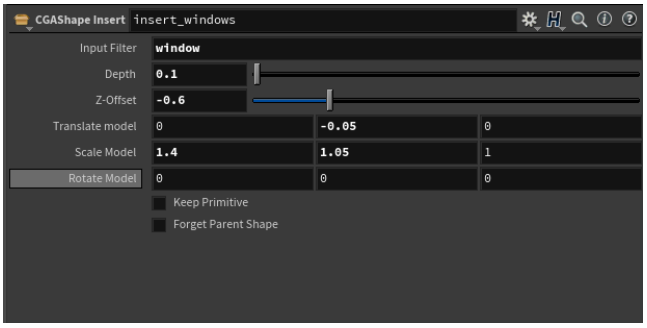


Figure 4.10: Interface of Insert Node

4.2.1 CGA Pipeline

Now the CGA Pipeline will be established. The stages of the Pipeline can be seen in Figure 4.11 and each one separately will be briefly explained.

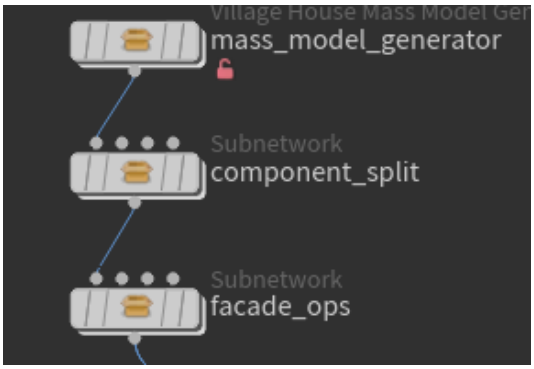


Figure 4.11: CGA Pipeline

Mass Model Generator

The process begins with rules that construct the general shape of the building, in the example is a Box (Figure 4.12). There are 2 methods to achieve this, the more suitable was chosen which starts with a floor plan and extrude on to get a roof (Figure 4.13).

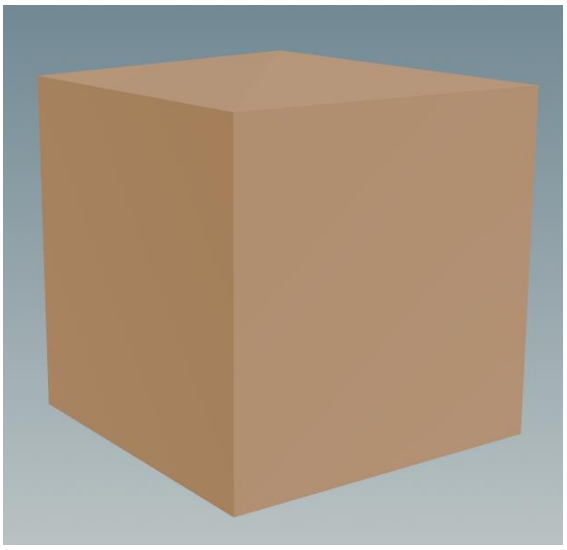


Figure 4.12: Starting Shape

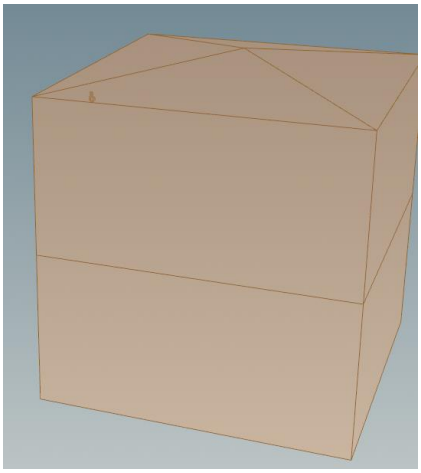


Figure 4.13: Extrude the roof

Component Split

Each Façade is individually named and put in a separate group according to what the user wants to show there. In the Figure 4.12 can be seen the group façade entrance to help the user understand where the entrance of the building is going to be and the user has to place the door in the group entrance. There are also more group facades like roof, door, windows and more.

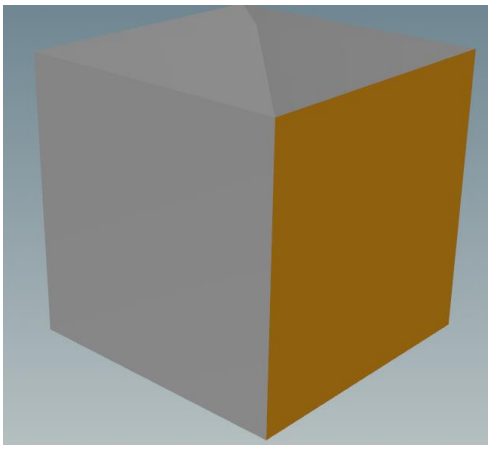


Figure 4.14: Separate Facades (grouping) – group entrance

Façade Operations

Now that the facades are prepared they can be used to end up with the shapes of windows, doors and other features. Figure 4.15 shows the facade of walls and door. Façade door will tell the user where exactly the door needs to be placed. Another example can be seen in the figure 4.16 of the window strips and then a more detailed façade for the windows.

It can be seen that it begins with a less detailed subdivision façade and finish up with more detailed ones.

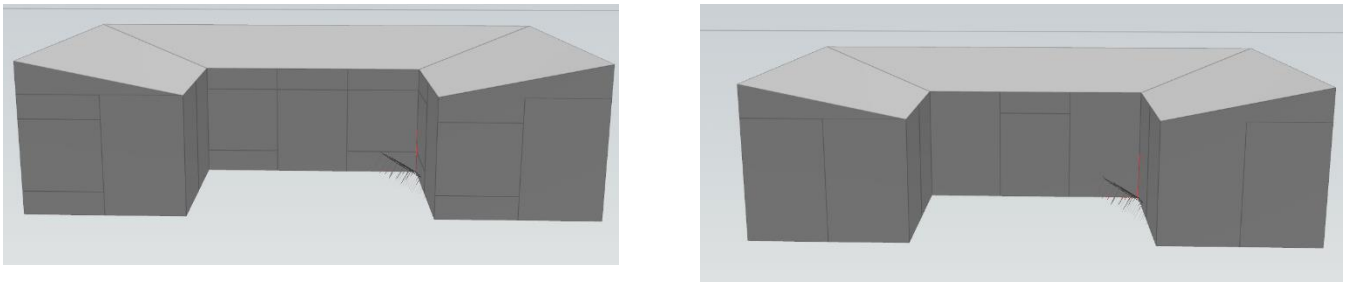


Figure 4.15: Façade Walls and Door

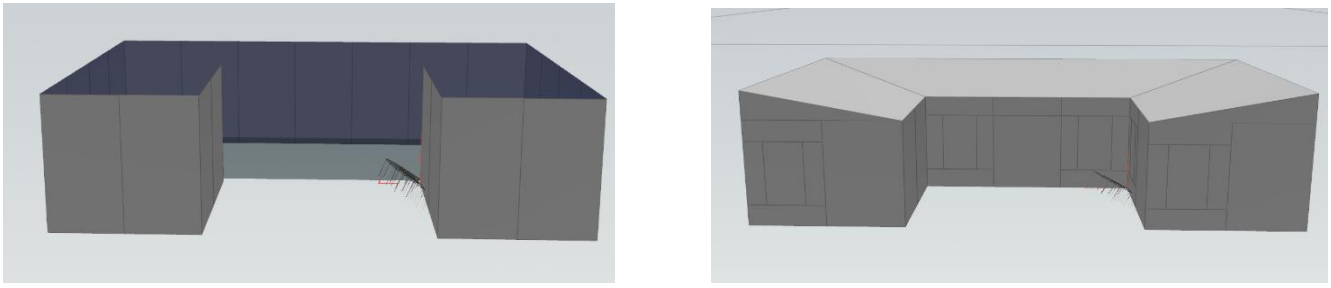


Figure 4.16: Façade Window Strips and Windows

Chapter 5

Methodology

Contents

5.1 Overview

5.2 Aging Simulation

5.3 Apply Aging

5.1 Overview

The process of how the simulation works will now be explained in detail. Figure 5.1 will be used as an example and each component will be individually introduced to gain a better understanding of how it operates.

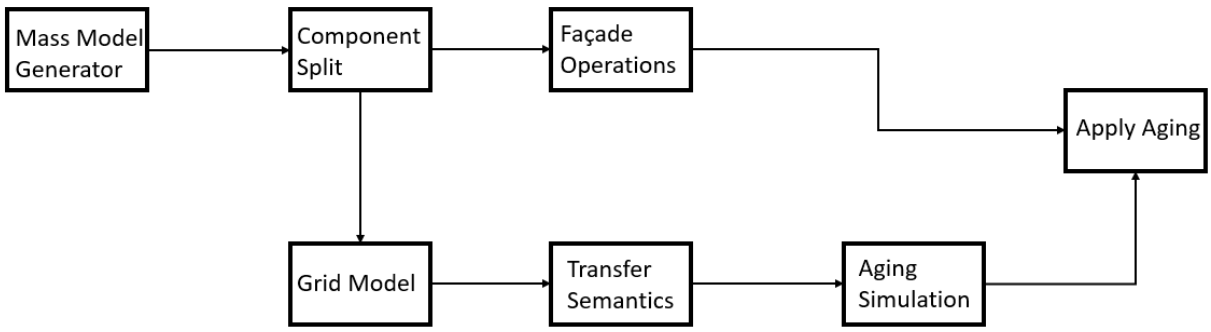


Figure 5.1: Diagram how the whole simulation works

Mass Model Generator

The process begins with the rules that construct the general shape of the building. There are two methods that do this: Either the basic shapes are combined (cubes, cylinders, cones, etc.), or the floor plan is placed first and serves as the extrusion point; this is the method chosen for this example.

Component Split

Once the general shape is being provided, the addition of the roof is following (when needed), and then split the surfaces (facades) to end up with the shapes of windows, doors, beams and other features. For the simulation to work properly each façade is being placed into an exclusive group.

Grid Model

The next step is to create a grid in the model, which will be used to hold the properties. The simulation will be better if the grid is more detailed but the complexity will also be higher.

Transfer Semantics

Transfer Semantics is a way to communicate and transfer knowledge to the grid model about the various objects that were created. The simulation's efficiency is aided by the transfer semantics.

Aging Simulation

Here the calculation of different primitives is taking place in the grid model in order to decide which primitives are going to have decline effects. The various approaches that took place here will be explained in subsection 5.2.

Façade Operations

Similarly, to the CGA technique it divides each façade and depending on the findings, the grid may need to be adjusted. The grid model's distinct cells are ignored when the facades are separated.

5.2 Aging Simulation

In this section the process of how the aging simulation works will be explained in more detail. The simulation depends on two files; a data file and a disp file. The data file is responsible to tell the model which primitives are damaged and which are still standing, and the disp file is responsible for all the displacements that need to be done in the model.

Import aging

The simulation starts with import aging. First, the data file is generated from a matlab script. The model is imported in matlab, then the matlab simulation runs and then data file is being generated. The file has values 0 or 1. The data contains as many columns as the times the user ran the simulation and the rows are equal to the primitives of the grid model. In other words, if the grid

mesh that ran the simulation has 3000 triangles, then the file will have 3000 lines. The first column is the row number. The remaining columns are whether this primitive stands (= 1) or is damaged (= 0) in each period of the simulation. In figure 5.2 can be seen the python code which reads the data file and set each primitive its value depends on the file.

Figure 5.2: Import aging code

Import displacements

The disp file contains the displacements of each edge at the end of the simulation. Unlike the data file, this file contains 3 float values in each row. Each value corresponds to the axis values (x, y,

z) of each edge. In figure 5.3 can be seen the python code which reads the disp file and does all the necessary displacements.

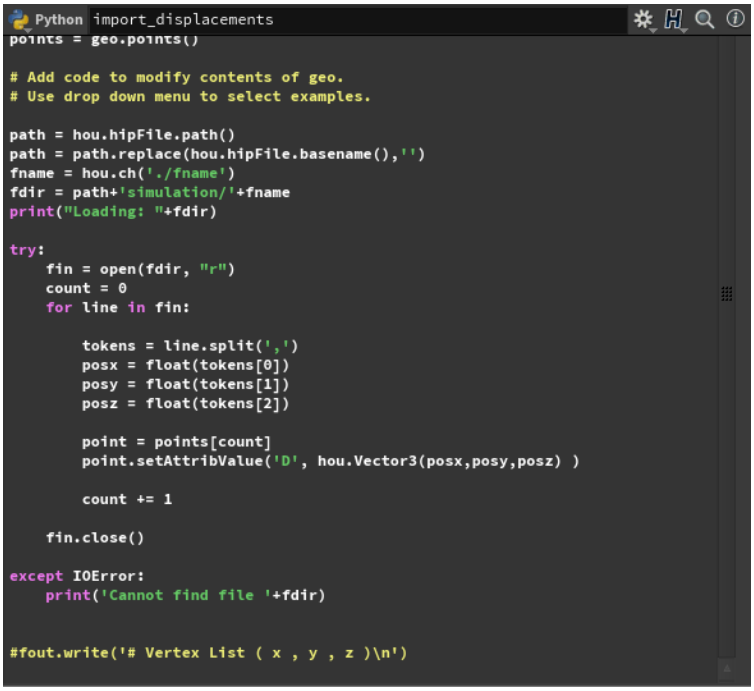


Figure 5.3: Import displacements code

5.3 Apply Aging

Figure 5.4 provides more insights which help in the understanding of the application of aging. The utilization of the three separate aging techniques/tools, and the combination of them gives fascinating weathering effects.

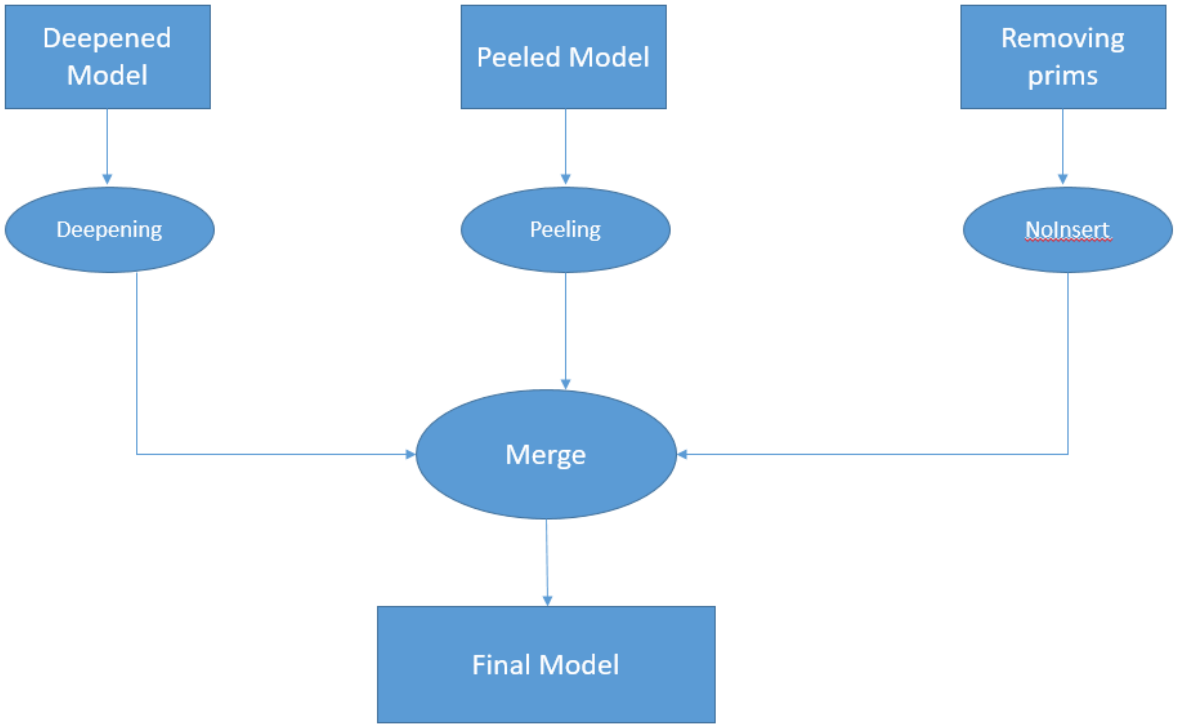


Figure 5.4: Diagram of how aging is being applied

Deepening

Deepening is the first tool that is being used. If all of the primitives' points are damaged, the quad is destroyed, resulting in holes in the region. This method could be used on walls, but it would require more subdividing and the complexity would be very high, therefore roof areas are better suited for it.

This tool generates a fascinating effect in figure 5.5, with tiles tilting against the hole that was created, making the eye to understand that it is about to fall.

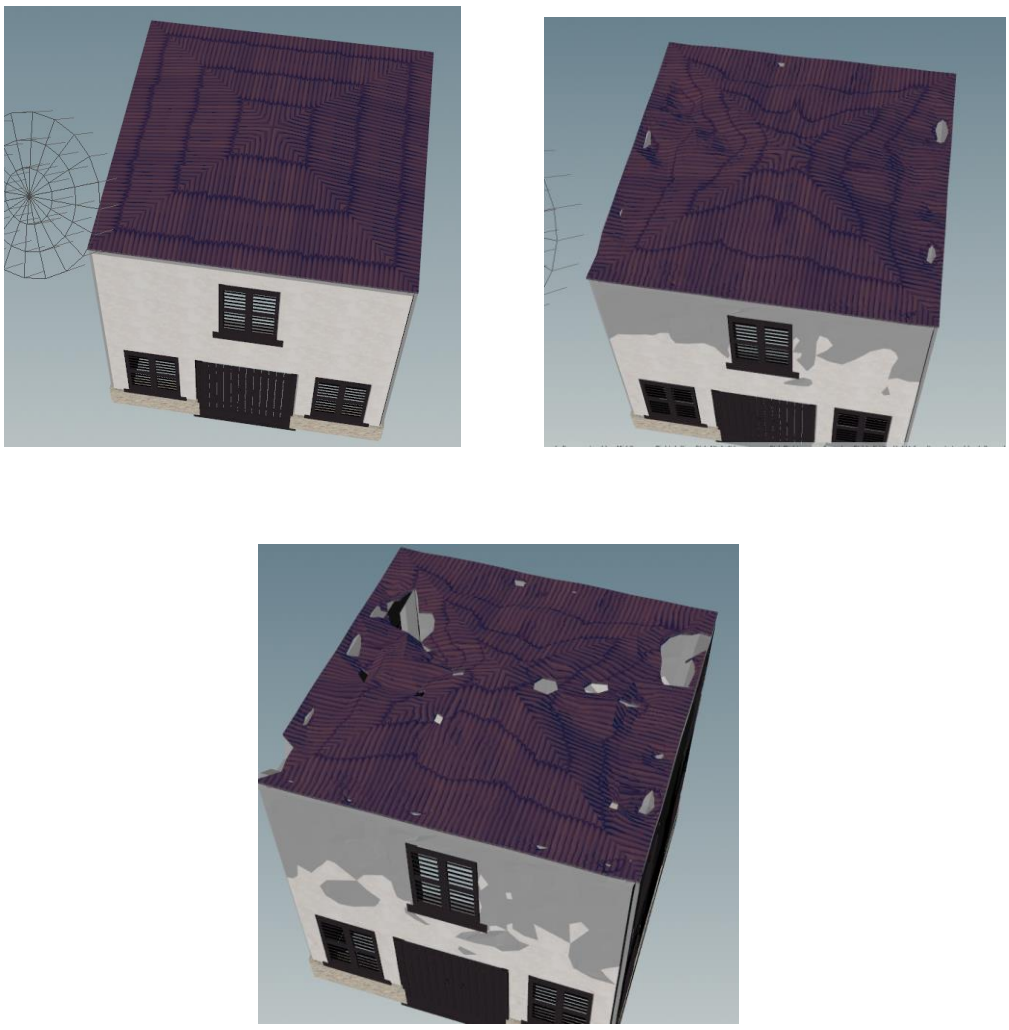


Figure 5.5: Effect of Deepening

Peeling

Peeling is the next aging procedure that is employed. Peeling considers the model to have many layers. Depending on how much a region has been damaged, the layers will begin to fade away, revealing the other levels underneath. For example, suppose the structure is made up of three layers: wood on the outside, brick on the center, and metal on the inside. Then, with a little damage, the exterior layer will begin to peel away, revealing the middle layer (brick), and after further

damage, the interior layer (metal), and ultimately, when the damage is sufficient, everything will vanish. An example can be seen in figure 5.6.

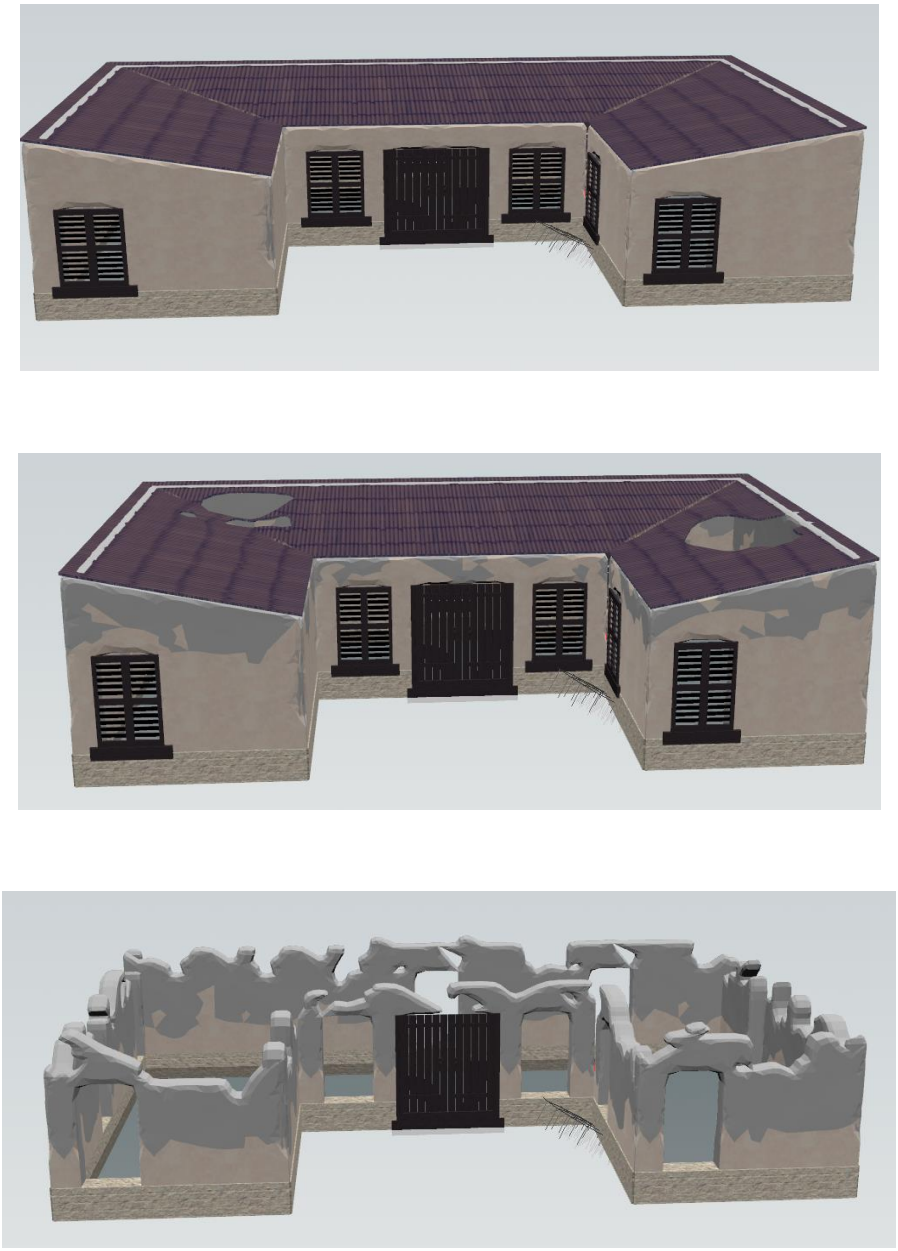


Figure 5.6: Effect of Peeling

NoInsert

Finally, NoInsert is used to prohibit things from being inserted into destroyed regions. NoInsert eliminates primitives to mimic the collapse of wall elements like windows, doors and beams similar to how Deepening works for the roof. However, unlike Deepening, displacements are not taking place here. In Figure 5.7 it can be seen how the door and windows gradually disappearing.

Figure 5.7: Effect of NoInsert

Chapter 6

Results

Contents

6.1 Introduction

6.2 Procedural Damage

6.3 Procedural Humidity

6.4 Procedural Wind Resistance

6.5 Procedural with Beams

6.6 Limitations

6.1 Introduction

To test the results of the procedural ageing solution, a home with tiles, doors, and windows is constructed using the CGA method. Figure 6.1 depicts many renderings of the house.

Figure 6.1: Some renderings of the house

6.2 Procedural Damage

Below follows an illustration of how the presented home ages as a result of damage contact. As shown in Figure 6.2, there are two sites where the house would be harmed. As the simulation progresses, certain changes are noticed in the house. As the frames (seasons) pass, the holes extend and the deterioration spreads to neighboring places. The three strategies described in section 5.3 can also be viewed.

Roof has Deepening, walls use Peeling for their layers and finally NoInsert to delete objects (door, window, beam).

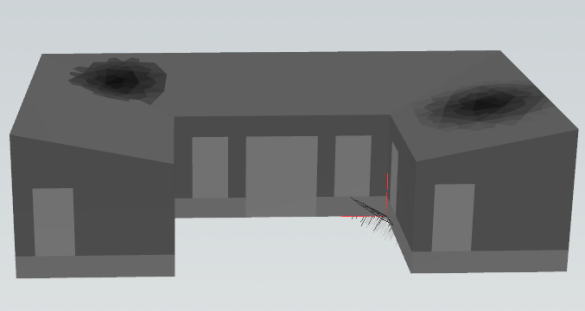


Figure 6.2: Procedural aging with damage

6.3 Procedural Humidity

Similarly, the same model is being used to show a different aging effect. The humidity effect also uses the three techniques like the damage effect but the peeling now is far stronger. It can be seen in Figure 6.3 the house's walls are peeling faster and it ages differently. The roof is also aging in slower pace because humidity has less power than damage.

Figure 6.3: Procedural aging with humidity

6.4 Procedural Wind Resistance

Another different and interesting aging effect is the Wind Interaction. The Wind Interaction is very similar to the damaged one but the damage is starting from right and goes on to the left side.

Certainly the three techniques (Deepening, Peeling, NoInsert) also apply in this effect but this time not in the whole house, they start from starting point (right side) and they keep going on until they reach the end point (left side). The results on this one are very clear and realistic.

Figure 6.4: Procedural aging with wind resistance

6.5 Procedural with Beams

The last illustration depicts how a roof might collapse with and without the use of wooden beams for support. To be able to make modifications and test the idea further, a new and more basic house is employed. It can be seen how the house would age without the support of the beams in Figure 6.5. This result indicates that the home aging isn't very realistic, and it needs to be improved.

As a result, to give the house a more authentic and convincing aging aspect the developer decided to employ the supporting beams. As expected, the use of the beams is particularly crucial as can be seen in Figure 6.6, and the aging effects are significantly more pronounced. Finally, the addition of the Deepening to the top, Peeling to the walls, and NoInsert to the beams it taking place.

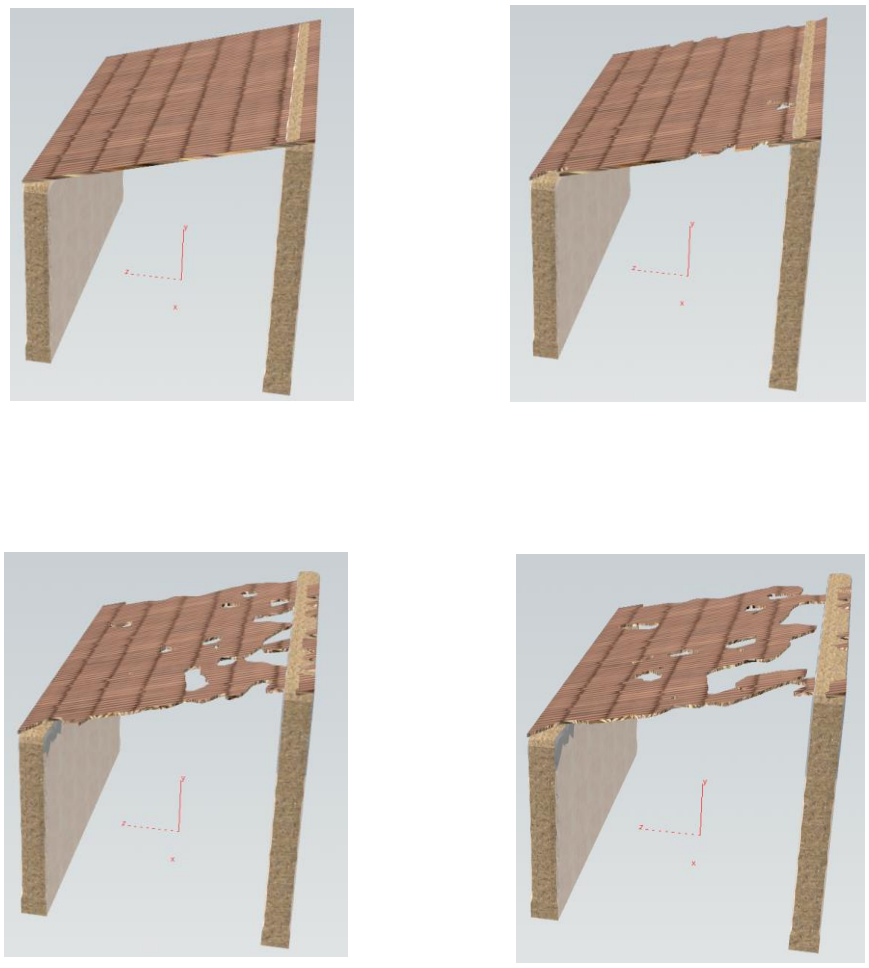
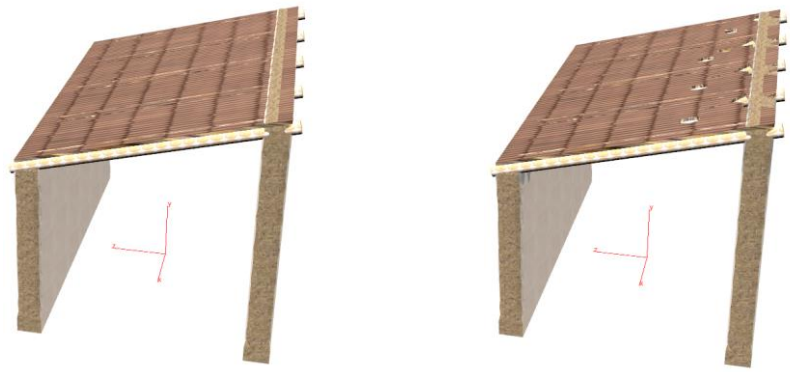


Figure 6.5: Procedural aging without beams



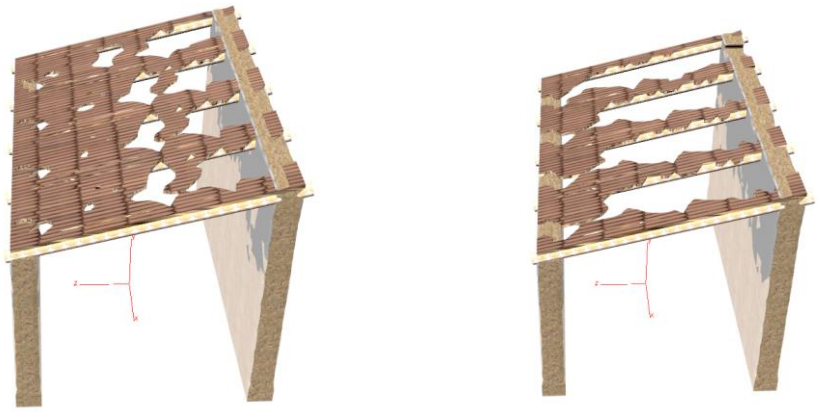


Figure 6.6: Procedural aging with beams

6.6 Limitations

After witnessing such beyond expectations results, some unexpected outcomes occurred that the system provided, similar to the ones in Section 6.5 without the beams.

Figure 6.7 shows an output of the same model what was discussed at the start of section 6, which appears to be peculiar. It can be seen that the wall ages unexpectedly after running the simulation for a few more frames. It is evident that the implementation is very good, but it isn't flawless and causes some minor anomalies, such as this one, which might be resolved with a little more information.

It appears to be a limit to the simulation's ability to perform flawlessly, but if there was a way to add more richness to the grid model while sacrificing time complexity, the simulation would be close to perfect.

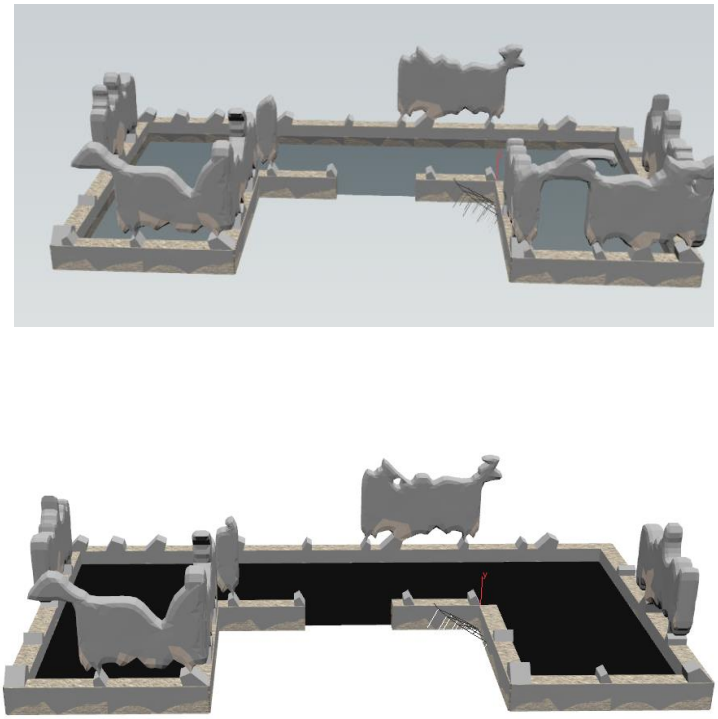


Figure 6.7: Unrealistic results

Chapter 7

Conclusions & Future Work

Contents

7.1 Conclusion

7.2 Future Work

7.1 Conclusion

Without a doubt there is a growing interest in procedural building, and the demonstration illustrated though this paper opens up a lot of possibilities for anyone who wishes to explore it further. The meaning of the CGA Shape Grammar has proven to be quite useful as it divides the models and allows graphic designers to create as many groups as they want in order to have as many features in their models as possible. Additionally, the Grid model as is used as a point database, recording various properties for each point that may be used by a user to allow one point to communicate with its neighbors.

In terms of the simulation, the files were used to create a procedural method. As the frames run the characteristics of each point change and the required displacement occurs while reading the files.

Finally, the results of the three aging procedures that were employed were also observed. The outcomes were really intriguing, and any artist may utilize their imagination and talent to adjust

how suitable an aging portrayal might be in whatever way they desire.

7.2 Future Work

Through this implementation some really great results were shown very similar to what to be expected. The model delivers good results in general, but there is still space for improvement. Taking into account the constraints, a goal could be established to overcome as many of them as possible.

The first issue that the simulation model lacks is sufficient detail, as doing so was not part of this research’s scope. That's because more details equals to more primitives, which means higher time complexity and the simulation will be much slower. So the first step is to figure out how to add more depth to the simulation model without sacrificing a lot of time in order to achieve better and more accurate results.

The tools focused on weathering processes, which mostly affects roof and wall surfaces. It would be great if a technique/tool could be found that exploits devastation from the ground up, such as earthquakes. As a result, the users can observe how a building may be suffering from below.

Another area where the implementation could use some improvement is the performance. Parallel programming is the most popular solution for these types of challenges. Parallel programming may assist by running code more efficiently, saving time and money by sifting through "huge data" more quickly than ever before.

Finally, the models could be utilized in other programs besides Houdini, such as Blender or Unity. Those two, notably Blender, are incredibly popular in modeling, rendering, and animation software. The simulation may be tough to transfer, but the models should not be an issue, and a

new or comparable technique could be used there.

Bibliography

Buildersmart.in. 2019. Types of Building Stones and their uses. [online] Available at: <https://www.buildersmart.in/blogs/types-of-building-stones> [Accessed 5 December 2021].

Choice Reviews Online, 1997. The Oxford companion to archaeology. 34(11), pp.34-6037-34-6037.

D'Costa, K., 2015. A Story of Wood. [online] Scientific American Blog Network. Available at: <https://blogs.scientificamerican.com/anthropology-in-practice/a-story-of-wood/> [Accessed 5 December 2021].

En.wikipedia.org. 2006. Houdini (software) - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Houdini_(software)> [Accessed 4 December 2021].

Gutierrez, D. and Sheffer, A., 2015. Procedural Modeling of Aged Buildings, 37(2).

Klippel, A., n.d. Understanding CGA Shape Grammar | GEOG 497: 3D Modeling and Virtual Reality. [online] E-education.psu.edu. Available at: <https://www.e-education.psu.edu/geogvr/node/891> [Accessed 9 December 2021].

M¨uller, P. and Parish, H., 2001. Procedural modeling of cities | Proceedings of the 28th annual conference on Computer graphics and interactive techniques. [online] Dl.acm.org. Available at: <https://dl.acm.org/doi/10.1145/383259.383292> [Accessed 9 December 2021].

Müller, P., Wonka, P., Haegler, S., Ulmer, A. and Van Gool, L., 2006. Procedural modeling of

buildings. ACM Transactions on Graphics, 25(3), pp.614-623.

Müller, P., Zeng, G., Wonka, P. and Van Gool, L., 2007. Image-based procedural modeling of facades. ACM Transactions on Graphics, 26(3), p.85.

Ozinga. 2013. 6 Popular Uses for Concrete - Ozinga. [online] Available at: <https://ozinga.com/blog/6-popular-uses-for-concrete/> [Accessed 5 December 2021].

Sidefx.com. n.d. Delete. [online] Available at: <https://www.sidefx.com/docs/houdini/nodes/sop/delete.html> [Accessed 4 December 2021].

Sidefx.com. n.d. FX Features | SideFX. [online] Available at: <https://www.sidefx.com/products/houdini/fx-features/> [Accessed 4 December 2021].

Sidefx.com. n.d. Geometry attributes. [online] Available at: <https://www.sidefx.com/docs/houdini/model/attributes> [Accessed 4 December 2021].

Sidefx. n.d. Geometry functions. [online] Available at: <https://www.sidefx.com/docs/houdini/vex/geometry.html> [Accessed 4 December 2021].

Sidefx.com. n.d. Groups. [online] Available at: <https://www.sidefx.com/docs/houdini/model/groups.html> [Accessed 4 December 2021].

Sidefx.com. n.d. Python Script. [online] Available at: <https://www.sidefx.com/docs/houdini/nodes/top/pythonscript.html> [Accessed 4 December 2021].

Wonka, P., Wimmer, M., Sillion, F. and Ribarsky, W., 2003. Instant architecture. ACM

Transactions on Graphics, 22(3), pp.669-677.