Visualization in Autodesk Maya 2012 Representing a Cell DEVS Simulation of a Hydraulic Fracturing Model

Meagan Leflar
Carleton University
meagan_leflar@carleton.ca

Keywords: Discrete event simulation, DEVS, environmental science, hydraulic fracturing, Cell DEVS, simulation, model, Autodesk Maya 2012, visualization, animated materials, polygons, polygonal generation, mesh generation.

Abstract

Induced hydraulic fracturing is a system employed by the energy industry to extract natural gas or petroleum from reservoir rocks. This process involves the insertion of a wellbore into the ground which injects high pressure fluid into the ground, causing fractures in the rock to create a conduit for natural gas and petroleum. Numerical simulations of induced hydraulic fracturing have existed for some time. This research explores the implementation of a 3D visualization of simulation results from a cell DEVS model of induced Hydraulic fracturing. This visualization uses Autodesk Maya 2011 and MEL script to accomplish this task. Further visualization designs and possibilities are also explored.

1. INTRODUCTION

This project has been undertaken as part of a collaborative effort with Sergio Zlotnik, a fractures expert from the Polytechnic University of Catalonia in Barcelona. Under the tutelage of Prof Gabriel A. Wainer, Professor of Systems and Computer Engineering at Carleton University, Christopher Burt and Meagan Leflar worked together to implement and visualize Zlotnik's theoretical cell DEVS (Discrete Event System Simulation) model representing the behaviour occurring when one practices hydraulic fracturing. Christopher Burt was responsible for the implementation in CD++ (software which enables the development of DEVS simulations with C++). The output of Mr. Burt's work became the input to the visualization using Autodesk Maya 2011 which is described in this paper.

This visualization in Autodesk Maya uses MEL script to read the simulation output from CD++ (a log file), and uses this information to generate appropriately coloured polygons to represent the cell DEVS simulation in 3D space.

However, before we discuss the exact nature of this simulation further, let us explore the background of hydraulic fracturing, the simulation model that Sergio Zlotnik designed and that Christopher Burt developed.

2. BACKGROUND

2.1. Hydraulic Fracturing

Simply put, hydraulic fracturing is the production of fractures in rock layers propagated by the movement of a pressured fluid. This is a phenomenon that can occur naturally in the environment, and has been observed to at possibly result in the congregation of gas or petroleum to reservoir rocks by using the newly created fractures as a conduit [1]. Induced hydraulic fracturing (also known as hydrofracturing or less-formally as simply fracking) is a system that was developed to create hydraulic fracturing manually in order to collect petroleum or natural gasses such as shale gas.

Induced hydraulic fracturing is accomplished by drilling a wellbore into reservoir rock formation, and pumping fluids out of the wellbore at high enough pressures to create fractures. The simulation model that will be explored in this paper uses the scenario of induced hydraulic fracturing.

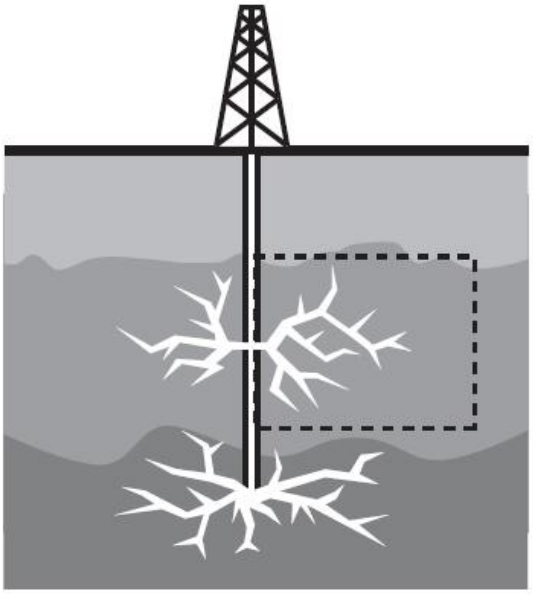


Figure 1: Illustration of the induced hydraulic fracturing process, provided by Sergio Zlotnik. The dotted line here is to represent the domain of the model.

Induced Hydraulic fracturing is widely used, and has been used for many years now [2], and thus it is important to understand the hydraulic fracturing process in order to better judge where and how fractures will occur.

2.2. Modeling Hydraulic Fracturing

As recently as 2010, it has been discussed that the petroleum industry's lack of understanding of physical properties and the physics controlling production from many important resources plays limits its ability to model and forecast with confidence production and reserves from these resources in many cases [3]. While measurement technology and modeling accuracy is improving, the industry is often forced to resort to empirical methods that lack the usual validation required for high confidence in results.

Dr. Zlotnik also understands that many numerical simulations of hydraulic fracturing have been done pertaining to plane stress [5], plane strain [6], or radial crack [7]. However, the wealth of complex models does not appear to be filling field-work and industry needs. "Petroleum engineers, at a loss to determine which model might best describe their application, typically pick the simplest they can find to give them some kind of number." [4]

The model proposed by Sergio Zlotnik is a simple framework for simulating induced hydraulic fracturing using cell DEVS which could be extended to include more complex calculations and more advanced consideration could be integrated at a later time.

2.3. Cell DEVS

The approach that was taken to the development of this hydraulic fracturing simulation was that of a simple cell DEVS model. DEVS stands for Discrete Event Systems Specifications. DEVS models work as a hierarchy of links connecting atomic models (the smallest component of a DEVS simulation) via inputs and outputs.

Cell DEVS are slightly more specialized in that the simulation exists as a grid of cells each with a state denoting an identifying float value that the cell has which is relevant to the simulation. In these simulations, the cell's state is often expressed as a color using a pallet file which the developer writes to dictate which float values pertain to which colours.

Each cell has a specific set of relevant neighbouring cells. The cell's state is then determined at specific times using a set of rules that govern the cell's behaviour. These rules are often written based off of the states of the cell's neighbouring cells.

Other relevant information about cell DEVS models include the dimensions, described always as rows by columns, and that cell DEVS systems can be layered as well as connected with standard DEVS models. In the case of this induced hydraulic fracturing simulation, however, the cell DEVS model is only one layer.

3. THE MODEL

Both Dr. Zlotnik's conceptual model and Christopher Burt's implemented model will be discussed in this section.

3.1. Dr. Zlotnik's Model

In his exploration of the fracturing process, Dr. Zlotnik suggests that permeability and porosity of the rock play an important role in how the fluids will

move within the rock layers, and how the rock layers will fracture. This would also, therefore, be relevant to the extraction of natural resources using hydraulic fracturing.

The permeability of a rock describes how easily a fluid can flow through it. The porosity of the rock represents how much fluid a particular section of rock can contain, meaning there would be holes in the rock to be filled by the fluid. However, if these holes or openings are not interconnected within the rock, though it may be porous, it would not be permeable. Hydraulic fracturing can then be used to create conduits between pores and can therefore increase permeability and lead to better results.

Dr. Zlotnik's model aims to remain simple in its design and implementation, and thus will assume certain idealized parameters:

- The rock is uniform in all orientations.
- The rock is homogenous.
- The rock obeys Hooke's law of linear elastic behaviour.
- There is an impermeable borehole wall that is unaffected by the wellbore pressure on the left side of the simulation grid.

In this model, each cell represents a portion of space that can be rock, fracture, or fluid. This simulation will begin with the injection of fluid through a cell midway up the left side of the cell DEVS grid. This cell will represent the wellbore in induced hydraulic fracturing, and the spread of pressure will begin there.

The first part of this model describes the movement of fluid along the fractures that are created or that exist in the rock. To calculate this, only the distribution of pressure will be used. The nature of each cell is calculated based on the averaging of the pressure of neighbouring cells. This model enables the increase of pressure injection (fluid from the wellbore) to spread along neighbours that are fractures.

The second part of this model deals with when the fractures are created. Fractures occur when the fluid pressure within the rock exceeds the smallest principal stress plus the strength of the rock. This is expressed in the following equation, supplied by Dr. Zlotnik:

$$p_{max} = 2\theta - p_f + T_0$$

In this equation, θ is the smallest principle stress, p_f is the fluid pressure, and T_0 is the tensile strength of the rock. If the pressure of one of the rock cell's neighbours reaches the maximum pressure, the rock will become a crack. The new crack is then filled by the fluid from its neighbour.

Rock cells should have their own pore fluid pressure and tensile strength properties. These properties will be distributed according to the table below. Please note, pore fluid pressure and stress are dependent on the depth of the cell in question.

Table 1: Table supplied by Sergio Zlotnik describing the properties of each cell.

simbol	description	units	value
ρ	rock density	kg m^{-3}	2200
T_0	rock tensile strength	Pa	$15 \pm 1 \times 10^6$
g	gravity acceleration	m s^{-2}	9.8
gP	pore fluid pressure gradient	Pa m^{-1}	10500
p_f	pore fluid pressure	Pa	depth(m)$\times gP$
σ	stress	Pa	depth(m)$\times g \times \rho$

3.2. Mr. Burt's Implementation

Using the description supplied by Sergio Zlotnik, Christopher Burt developed a cell DEVS simulation implementation using CD++, an Eclipse plugin that uses C++ to implement DEVS models.

Though the design of a 3D simulation was briefly discussed, it was decided that simply refining a more effective 2D simulation would be more beneficial at this time. However, by developing a pipeline for visualizing the simulation in 3D opens the door for future development and visualization in 3D.

Screen captures of the simulation that Christopher Burt implemented can be seen in figure 2.

His final simulation results were then handed off to us as .log file which contains a record of everything that occurred during the simulation.

4. METHOD

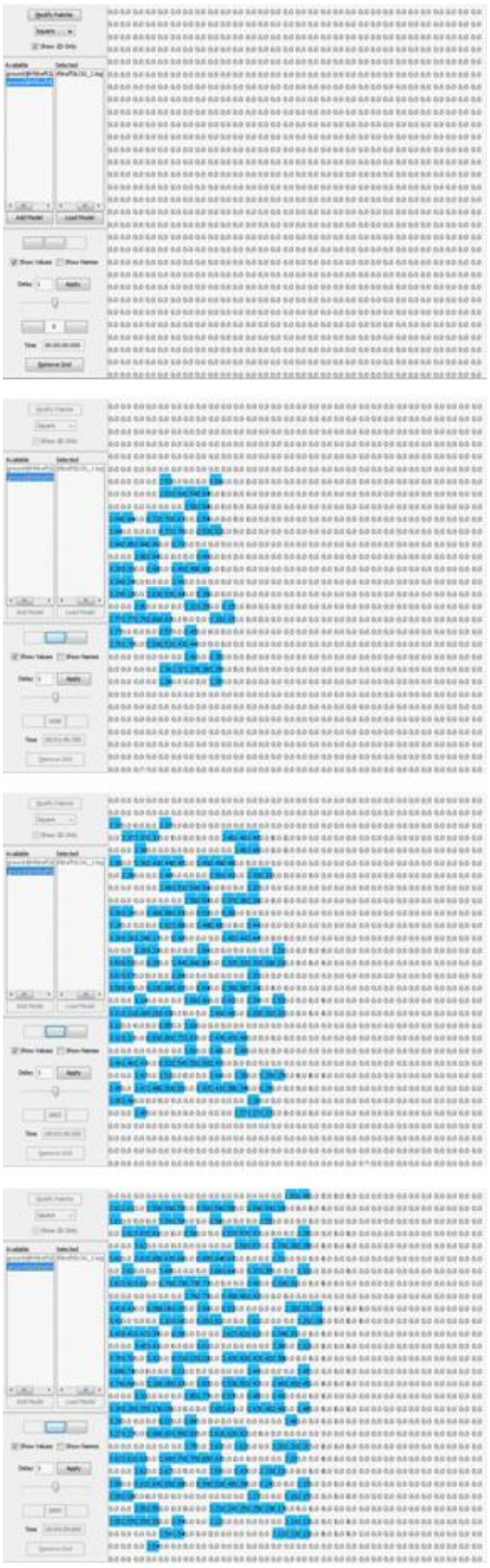
4.1. Methodology

The approach that we decided to take to visualize this simulation involves the use of polygonal cubes and the manipulation of their materials. This was decided because the cubic nature of the polygonal meshes implemented most directly reflected the nature of the simulation data, which is a grid using four edges to determine the neighbouring cells. Polygons were selected because they are the most intuitive and visual representation in 3D space to symbolize solid rocks.

Different approaches were considered initially, however. The first option that was explored was fluid dynamics. Fluids in Maya came to mind because of the fluid that is injected into the rocks as part of hydraulic fracturing. Unfortunately, simulating the desired fluid behaviour is already covered as part of the cell DEVS model, and we felt it was important to retain the cellular nature of the simulation to be visualized. Particles were considered, and dismissed for similar reasons.

Locators (point information) were also considered as an option for an ever simpler visualization, however they are visually uninteresting and discovering a way to communicate states using simply point information was unintuitive.

Figure 2: 2D simulation results from Chris Burt.

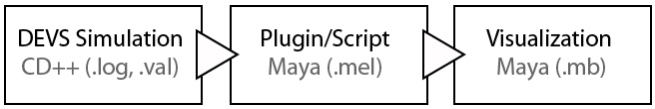


The focus of this visualization was to acquire the information from the CD++ simulation and translate it into Autodesk Maya 2012. Once this information has been properly interpreted, changing details of the visualization is fairly simple.

4.2. Visualization Pipeline

This visualization takes the output from the 2D simulation created in CD++, extracts the relevant information, and uses it to create a visualization in Autodesk Maya 2010 using MEL script. The figure below illustrates this simple pipeline.

Figure 3: Visualization Pipeline



The output from the CD++ simulation that Christopher Burt ran is parsed into a log file, with the extension .log. This file contains all of the events that occur during the DEVS simulation. This file will be read and analyzed using MEL Script in a .mel file. The pseudo code for this process will be discussed in more depth.

The visualization, though generated with the MEL Script file, will be outputted into a Maya Binary file (.mb). This Maya Binary file can then be saved by the user, or rendered manually. This simulation includes generation of a Maya scene (including animation and polygonal mesh generation) but it does not include rendering the animated scene. The animation can however be viewed by the user immediately after the MEL Script completes running. This can be done in the Maya window using the timeline to toggle the frames.

4.3. Script Overview

The pseudo-code for this visualization can be summarized nicely using the flow chart in figure 4.

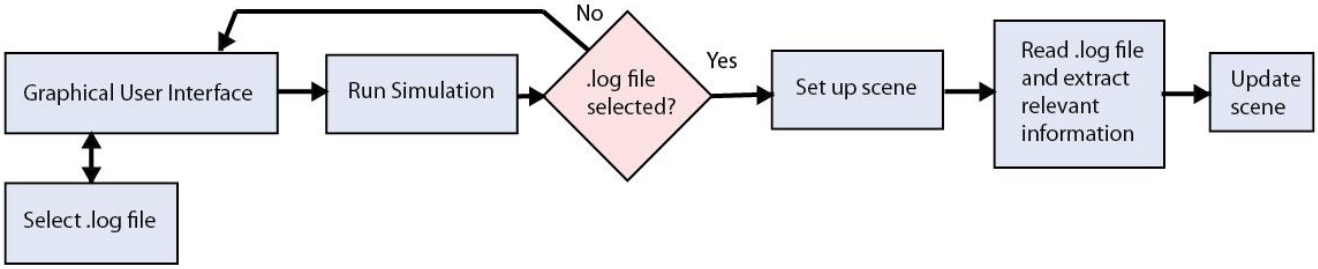
The first step is to create the graphical user interface (GUI), which will give the user the ability to control the dimensions of the simulation and to select the log file of the simulation that they wish to visualize.

Selecting the "Run Simulation" button the graphical user interface will then prompt the script to check if the user has selected a log file. If no log file has been selected, then the user is prompted to enter a log file and the simulation is aborted. If the user has in fact selected a log file, then the simulation will continue.

The next step is to set up the scene based on the dimension parameters that the user has inputted, in this case drawing cubes to the screen.

Once the scene has been set up, the information for each step of the simulation is then extracted from the user-selected log file. This information is then used to update the scene. For each update, a key-frame is added to create the animation with the appropriate timing. Key-frames will be discussed in more depth in section 4.7.

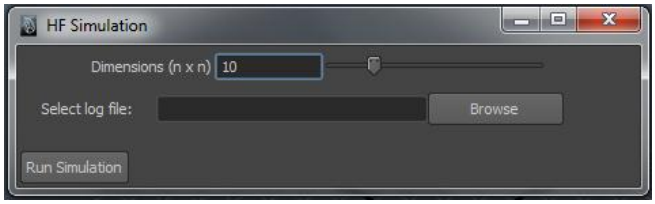
Figure 4: Pseudo code flow chart



4.4. Graphical User Interface

When the user first runs the visualization, they will be greeted with this graphical user interface window.

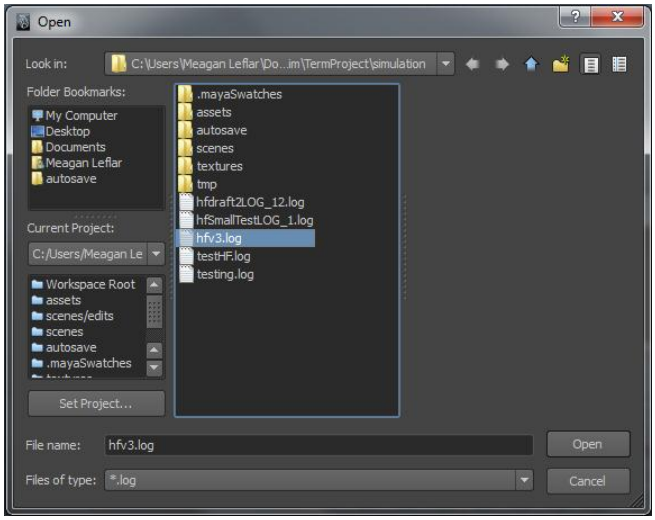
Figure 5: Graphical User Interface



This window was fairly simple to implement using MEL Scripts GUI commands. The data from the input parameters (dimension and log file) are acquired in the simulation function, which is called when the user presses the "Run Simulation" button.

The user selects the "Browse" button to select a file. The pseudo code for this is that a browse file function is called, which opens a separate window (see Figure 6) which enables users to browse for files. The code also filters the available files to .log files only.

Figure 6: File browser window



The style of window that was implemented using the MEL script is Maya specific and independent of different Operating Systems (OS) such as Mac, Windows or Linux.

4.5. Simulation (Main) Function

When the user selects the "Run Simulation" button the simulation function is called. This is essentially the main function of the simulation. This function creates a new scene, to ensure a clean slate for the rest of the simulation. It then extracts the file path to the user-selected log file (as a string).

The other essential function of this main simulation function is to check if the user has in fact selected a valid log file. If no valid log file has been selected, then a confirmation dialogue box is created, prompting the user to select a log file before continuing. On the other hand, if a valid log file has in fact been selected, then the set up function is then called, followed by the function which reads the relevant file and uses the information extracted from this file to implement the simulation visualization. These functions will be described in more depth in the following sections.

4.6. Set Up

The set up function uses the user-specified dimensions of the cell space to create a 3D grid of polygonal cubes.

The two important factors of the implementation of this function are the materials applied to each cube and the naming convention used.

4.6.1. The Materials

In Autodesk Maya 2011, materials are nodes that are applied to Maya meshes (such as the cubes used in this simulation) to make them appear to be made out of a certain material. For example, materials can be different colours, and have different specular levels.

In this visualization, each cube represents a cell in the induced hydraulic fracturing simulation model. Cells in the simulation model (and therefore cubes in the visualization) will not share the same state at the same time. Some cells will have states pertaining to being a rock, others will have states pertaining to being a fracture, and others even will have a state value pertaining to being filled with the fracturing fluid.

Because of the way Maya materials are designed, a different material must be applied to each cube in

order to be manipulated independently. Otherwise, as soon as one instance of the material on one cube is changed, that same material on every other cell would also change.

4.6.1. The Naming Convention

In order to be able to access the material of each cube in a logical manner, a specific naming convention was designed and implemented. This naming convention mirrors the row by column layout of the cell DEVS model. This naming convention is illustrated in a small five by five matrix in figure 6. Aligning the names of the meshes and the materials with the naming convention in the cell DEVS model allows for a smoother transition between simulation and visualization.

Materials are therefore called "mat5_0" where "mat" designates that this object is a material, 5 designates that it has been applied to a cell in row 5 (the sixth row) and column 0 (the first column).

The associated polygonal mesh of the cube is therefore called "c5_0". The "c" designates that this objects is a cube, and the "5" and "0" tell the script that it is in row 5 and column 0.

This naming convention is essential to the pseudo code described in section 4.8.

Figure 6: Illustration of naming convention used for the induced hydraulic fracturing visualization in Maya, based on the row/column naming convention in CD++.

(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)
(4, 0)	(4, 1)	(4, 2)	(4, 3)	(4, 4)

Figure 7: Illustration of the grid of cubes that are generated programmatically using MEL script in the set up function.

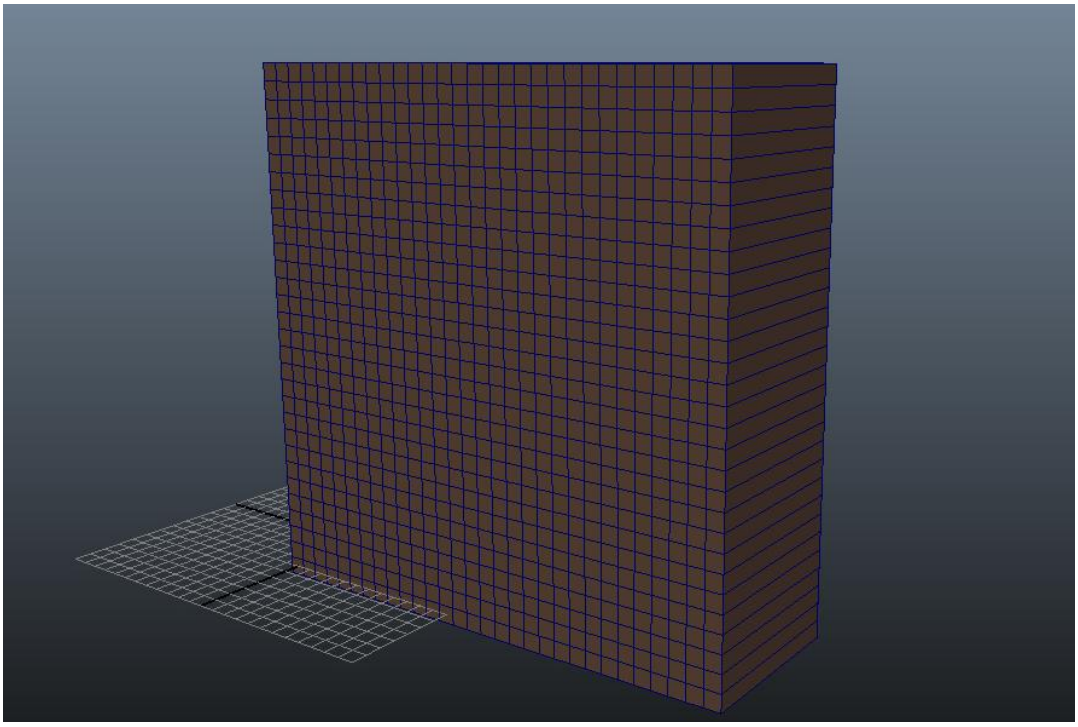
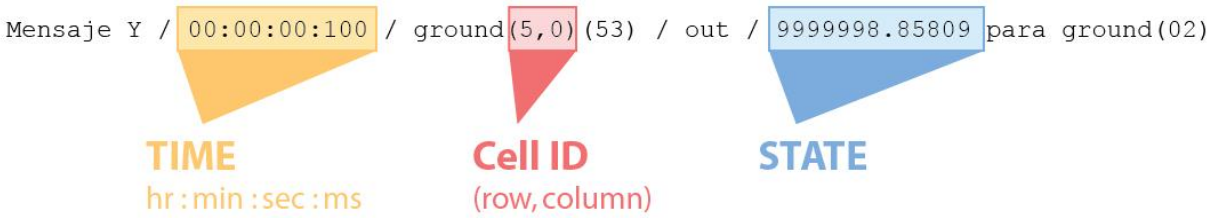


Figure 8: Visual break down of information extracted from log file.



4.7. Extracting Relevant Information

In this function, the log file will be opened and read. The first step to understanding this section is to understand the anatomy of the log file produced by CD++.

Although there is much more information present in the log file than simply the out messages, the out messages are the only messages relevant to visualizing a simulation in this case. These output messages are sent when the cell changes states and they are preceded with "Mensaje Y". Therefore, the first step is to filter out all of the messages that do not start with "Mensaje Y".

The pseudo code for this is as follows:

- Acquire the first line from the file
- Use regular expressions (a means of matching strings of text) to check if the message type is "Y".
- If so strip relevant information from this line.
- Move to next line, until the end of the file has been reached.

An example of a standard output message, and the relevant material that it contains, is illustrated above in figure 8. The relevant information here is time, cell ID, and state.

4.7.1. Time

Time is parsed from the output message string using tokenization (dividing a string by a certain character, in this case slash and colon characters). Hour, minute, second, and millisecond values where acquired and converted to floats.

However, the time in hours, minutes, seconds, or milliseconds are not relevant. The relevant value here is the frame number at which the change occurs.

Despite recent theaters upgrading to 48 frames per second for a larger-than-life immersive animation experience (due to the theatrical release of *The Hobbit: The Unexpected Journey* in December of 2012) the standard video frame rate is 24 frames per second [8]. Autodesk Maya's default frame rate is also 24 frames per second, but is customizable. For the purpose of this visualization, the assumption was made that this visualization would run at 24 frames per second, and care was taken during the exporting process that this remained accurate.

Taking this into consideration, the hour, minute, second, and millisecond values were then converted into frames by first converting hours, minutes and milliseconds into seconds, and then multiplying the total number of seconds by the frame rate (24 fps).

$$seconds = 24 / frame$$

This frame number will later be used to set key-frames for when the state of a cell changed.

4.7.2. Cell ID

The identification of each cell is formatted row by column, as described previously in section 4.6.1. This information is extracted from the log file output lines using a combination of tokenization techniques and regular expressions. This script returns two values, the row identification and the column identification. These

parameters will be used to locate the correct cube or material to edit based on the change of state.

4.7.3. The State

Similar to the time and cell ID parameters, the cell state was acquired from the output messages in the log file by using a tokenization method.

When it comes down to it, the state of the cell is the most important parameter of this entire visualization. The value that the state represents here is the pressure of the cell in question. This value was used in the initial simulation to calculate whether or not neighbouring cells would cause the current cell to "fail" and become a fracture, or become filled with fluid.

In the case of this visualization, the cell state is a value which will determine in what way that cube (pertaining to the same cell ID) will be visualized. This will be discussed in more depth in the following section.

4.8. Update Scene

This section is likely the aspect with the most options. This is when the real visualization occurs. In what way should we visualize the information that has been extracted from the simulation log file? In this case, we are manipulating the materials places upon the polygonal cubes that were generated programmatically in the set up function.

4.8.1. Selecting the colour

Although it might have been interesting to also play with the opacity of each cell based on the pressure (and this would not be very hard to implement in the future) to create a tree-like structure, this visualization functions predominantly by changing the colour of the material applied to the relevant meshes. Figure 9 below illustrates the colours of different states.

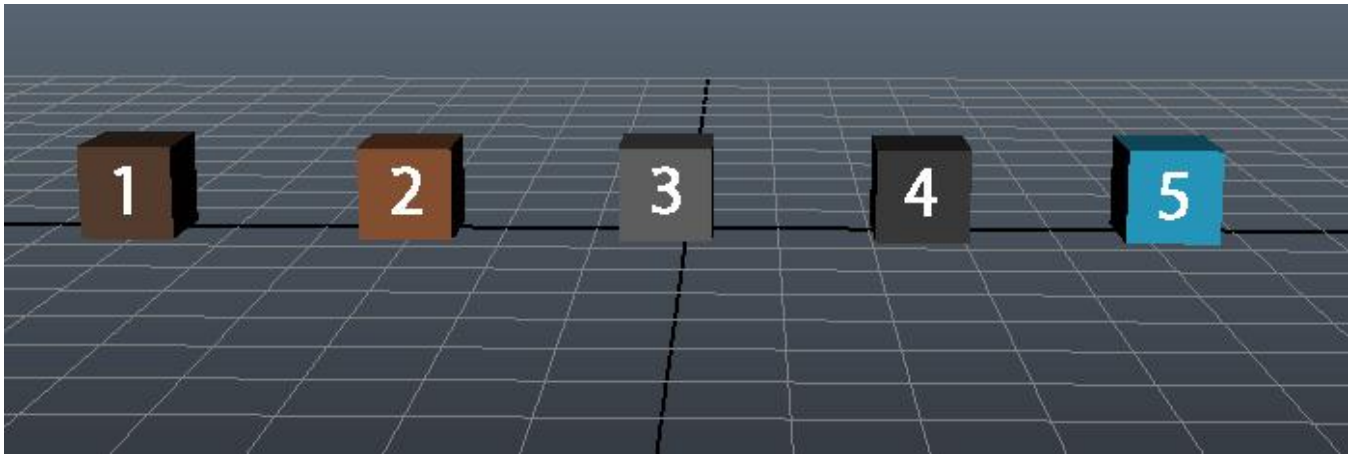
The colour of cube 1 represents state values between zero and one. Cubes with low fluid pressure values such as these symbolize rocks, and are therefore the dark, solid brown colour illustrated in figure 9.

The slightly lighter brown of cube 2 in figure 9 represents fluid pressures between one and ten thousand Pascals. This symbolizes rock that is under a little bit of pressure.

The grey cube illustrated as cube 3 in figure 9 is the colour that cells take when their fluid pressure rises between ten thousand and one million Pascals. This symbolizes a fracture, but is not strong enough to break neighbouring rock.

The dark grey cube, cube 4, in figure 9 symbolizes a full-fledged fracture. The pressure in this cell is between one million and one hundred million Pascals.

Figure 9: Illustration of different states in the induced hydraulic fracturing visualization.



Finally, the fifth cube that is blue in figure 9 represents cells with a pressure of one hundred million and higher. This is illustrated as a fracture filled with water, and at a base depth of three hundred meters that is nearly strong enough to break neighbouring rock.

The colours here were changed slightly from the original colours in Christopher Burt's simulation, because the earthy colours appeared more appropriate to represent the ground in the context of this visualization.

The colour to use was determined programmatically using an if statement and separating the states into relevant intervals. The colours themselves are hardcoded in to the visualization at this time. A possible future solution to this would be to create a similar reader for pallet files, which could then be manipulated to create a wider variety of results. However, though Maya materials do use RGB (red, blue green) values as their colour input, they are out of 1 instead of 255, so this conversion would also have to occur. This, and other possibilities, will be discussed in section 6.

4.8.2. Key framing

In order to create an animation in practically any animation software, including Autodesk Maya 2011, one must use key-frames. Key-frames are frames the define movement or change in an animation. Usually they mark the beginning or end of a smooth transition. For example, to animate a circle moving from point A to point B in 10 seconds, one would set a key-frame at 0 seconds with the circle at point A and a key-frame at 10 seconds with the circle at point B.

It then follows that anything that one might use to visualize a simulation in Autodesk Maya 2011 would have to possible to key-frame. Fortunately, most attributes in Maya can be key-framed. Unfortunately, the opacity of the textures in Maya cannot be directly key-framed, they must instead use Set Driven Keys. Essentially, this means that the value can be controlled by another object's parameter, but it cannot be keyed (meaning a key frame cannot be set on it) directly. Because creating a Set Driven Key for each of many cubes would be unnecessarily messy and complex, we decided to work directly with the colours of materials instead.

Controlling the opacity of textures could have been appealing as an option because, if one used a layered texture in Maya (a material with several textures applied upon it), one could use customized textures of water and ground as part of the visualization instead of a flat colour to represent the state. This option could still be explored further at a later date, but would also involve programmatic UV mapping of the polygonal meshes to which the textures will be applied. In simple terms, UV mapping determines the orientation and scale of the texture being applied to each face of the polygon.

Materials and textures sound like similar objects, but there are slight differences. Textures are applied to materials (specifically linking to and overriding the colour of materials) and materials are applied directly to the meshes in the scene. The attribute that we control in the induced hydraulic fracturing visualization MEL script is the colour attribute of a carefully labeled material.

The pseudo code for each state update is as follows:

- Go to frame (calculated in section 4.7.1)
- Set colour attribute of the material with the appropriate row and column identification to the colour specified by the state value.
- Set a key-frame on that material's colour attribute.

Once all the key-frames are in place, the visualization has completed and the user can view the visualization by clicking and dragging their cursor along the timeline at the bottom of the Autodesk Maya 2011 window. To control the length of the timeline displayed, simply change the first value below and to the right of the timeline from 24.00 to a higher number (five thousand for example).

5. VISUALIZATION RESULTS

This visualization was implemented in MEL script, and for that reason is perhaps not the most efficient implementation method with respect to time. The log files that were produced during the induced hydraulic fracturing simulation were over two million

Figure 10: Comparison between the CD++ results and the 3D visualization in Autodesk Maya 2011. The results on the left are the CD++ results, and the results on the right are screen-captures (not full renders) of the 3D visualization. This small resolution 10 x 10 grid was used to test the visualization script.

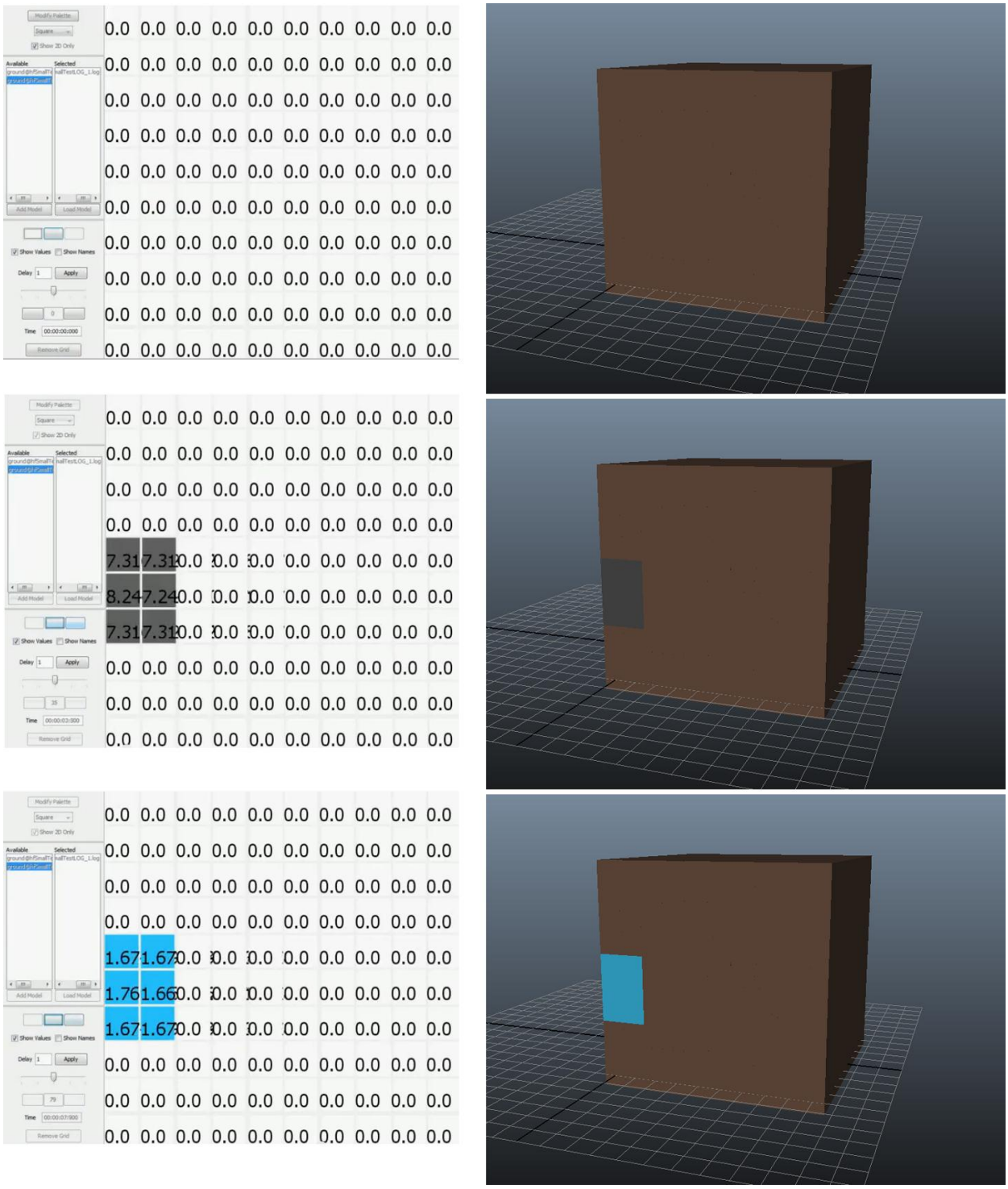
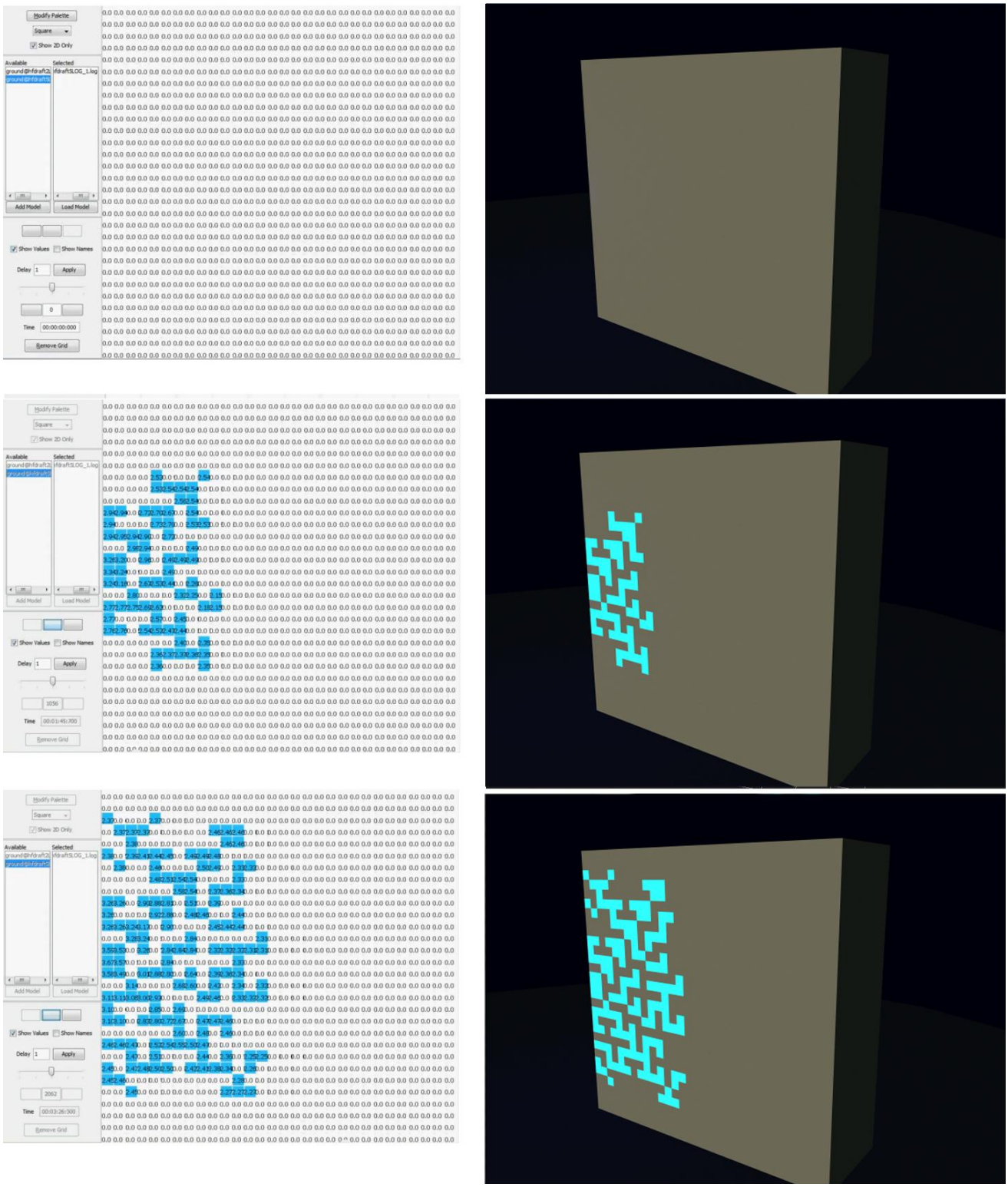


Figure 11: Comparison between the CD++ results (left) and the 3D visualization in Autodesk Maya 2011 (right) of a higher resolution 30 x 30 model. In this case, the visualization images were rendered.



lines long for a simulation space that is thirty cells by 30 cells. Unsurprisingly, it took quite some time for the visualization script to go through each line of the log file and extract the relevant information. The final visualization takes approximately 20 minutes to run with that resolution of simulation.

For this reason, initial testing, debugging and development were done using a smaller 10 by 10 grid of cells and a shorter simulation time of only 10 seconds. This simulation can be run in seconds.

The results of both visualizations are included in this paper. Figure 10 and 11 compare the 2D output of the CD++ plugin to the 3D output of the Autodesk Maya 2012 visualization. The higher resolution visualization looks slightly different because it was fully rendered. The lower resolution visualization was not rendered because it is simply being use to illustrate a smaller resolution test, and to demonstrate that different log files (from this hydraulic fracturing simulation) can be used to produce an equally accurate visualization in Autodesk Maya 2011.

As figures 10 and 11 demonstrate, the MEL script effectively and accurately takes the values from the log file and visualizes this data in a new way. Second by second, our visualization matches the CD++ results.

For the final renders of the higher dimension simulation visualization, additional lights were added to the scene for a more pleasing visual effect. The full video is available for download here: http://ug.csit.carleton.ca/~mleflar/fullSim_1.mp4

6. CONCLUSIONS

All considered, this visualization of an induced hydraulic fracturing cell DEVS simulation works accurately. To conclude this paper we will touch on the issue of efficiency, as well as approaches that could be taken in the future to improve this visualization. Such approaches include: reading from a pallet file, controlling opacity of cells, implementing a GUI which enables users to customize the simulation visuals, animating textures, output media for the simulation, and taking full advantage of the three dimensional nature of Autodesk Maya 2011. Additionally, we will explore the controversy behind induced hydraulic fracturing and reflect on the importance of simulation technologies in environmental awareness.

6.1. Efficiency

The most significant issue that this visualization has encountered is that reading the log file with MEL script can be slow for very long log files. The files that were received from Christopher Burt were over two million lines of code; text files too big to be sent via email. While the log files for the 10 by 10 cell grids took no time at all, the visualizations using 30 by 20 cell grids took up to twenty minutes to implement.

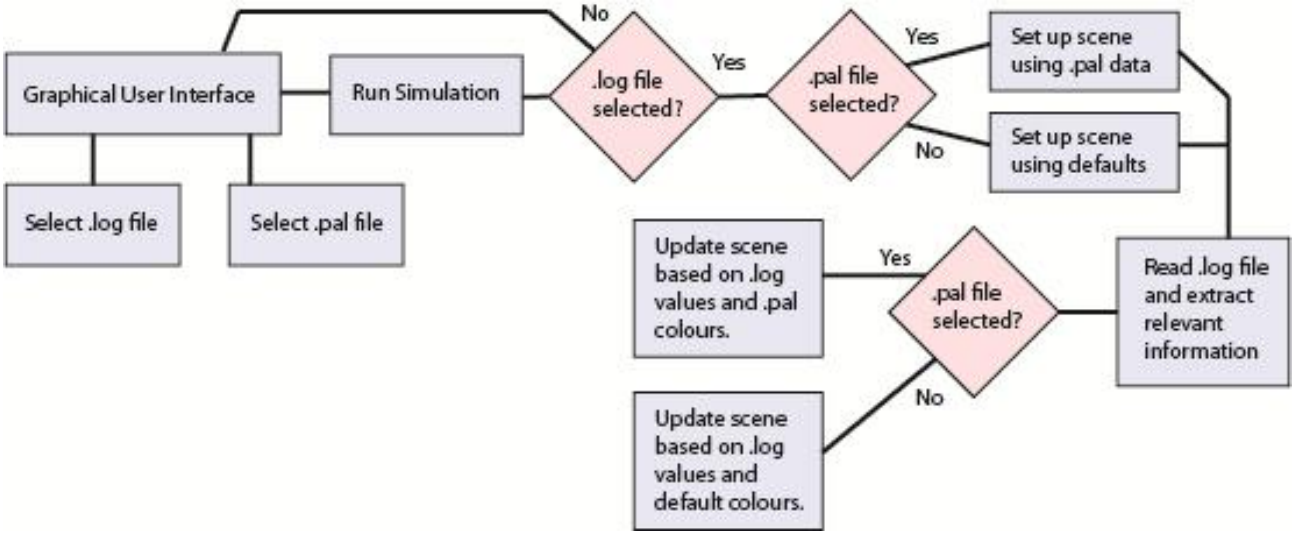
One has to wonder at the length of some of these log files. Could the CD++ simulation be designed in such a way to influence the length of the log file? Is every change in the pressure relevant to the state of the cell, or could the state be changed solely when the pressure reaches specific thresholds? This approach might serve to minimize the length of the log files because the state would only change when significant change in pressure levels have occurred. However, this approach would then influence the calculations of the average fluid pressure of the neighbouring cells, and may be less realistic in the end result.

A second approach that could be taken to minimize this efficiency dilemma would be to read and parse the file using C++ which could then port over the simplified file into MEL to implement the 3D visualization based on this information. Or, one could simply create the entire visualization in C++ using Maya API instead of MEL script. While MEL script is higher level, better documented, and much faster to develop, Maya API would likely read the file more quickly. Unfortunately, documentation for Maya API is fairly limited, which is why a combination of both C++ and MEL script might be ideal for future visualization projects involving mammoth log files.

6.2. Pal File

One way that this visualization might be improved in the future would be to allow users to also select the pallet (.pal) file from their CD++ simulation and use it to specify the colour scheme for their 3D visualization as well.

Figure 12: Updated pseudo code to include implementation of a pallet file as well as a log file.



Pallet files are files that contain information about the colour assigned to different ranges of state values. These files can be read and parsed in similar ways to the log files, except that the colour values in the pallet file are in RGB (red blue green) 255 format, while colours for Maya materials are out of 1, so this conversion would have to be taken into consideration when implementing this feature. The new MEL script pipeline would look something like the flow chart illustrated in figure 12.

6.3. Opacity

Another interesting way that the induced hydraulic cell DEVS simulation could be visualized would be to link the opacity of the polygonal cube to the pressure value of the cell. This would create an interesting tree like structure that symbolizes the areas of high and low pressure instead of trying to use colours to represent the shift from rock to fracture.

6.4. Customizability GUI

An alternative to using a pallet file to give users the ability to customize the visualization would be to extend the graphical user interface to enable users to control the colours, opacity, textures, etc., directly in the visualization interface.

6.5. Animating textures

During the development of this visualization, the possibility of using layered textures instead of simply changing the colour of the material was explored. A layered texture is a material that can have more than one texture applied to it, and the opacity of each texture could then be toggled to control which one was most visible at a specific instant. However, the issue that was encountered was that opacity of each texture could not be key-framed in the typical way, which made it difficult to control quickly using MEL script.

With layered textures (and layered Shaders, which are a similar concept but slightly different implementation) one cannot key-frame the opacity of individual textures that make up the layers, however one can connect the opacity of that texture layer to the attribute of another object using Set Driven Keys. One can then key-frame the attribute that they opacity of the texture layer has been connected to. This was not pursued initially because the process became very roundabout, but in hindsight could perhaps be refined with time if an appropriate attribute is selected as the driven key.

Another issue that becomes present using textures instead of simply using materials is that the polygonal mesh would have to be UV mapped in order for the textures not to look stretched and distorted. UV mapping essentially defines the relationship between the 3D object and the 2D texture, usually by “unwrapping” the 3D object. It is a fairly simple process to UV map a shape using automatic mapping,

but automatic mapping is not always flawless, so other approaches to programmatically UV mapping the meshes would also have to be explored.

6.6. Output Media

This section of the conclusion is more abstract and contemplative, but with the growing popularity of 3D output devices (such as 3D TVs and the technology behind 3DSs which does not involve using 3D glasses), it might be very interesting pursue taking advantage of 3D output medial as well as 3D visualization software when it comes to moving simulations from 2D to 3D.

6.7. Three Dimensions

It was briefly discussed with Christopher Burt and Sergio Zlotnik that the simulation be designed in 3D instead of simply in 2D. However, it was decided that the simulation rules should be refined and tested in more depth in 2D before a more extensive 3D simulation could be developed. Although the Autodesk Maya 2001 visualization is in 3D space, it is simply a 3D representation of a 2D model. It would better harness Maya's visualization capabilities if the simulation was in three dimensions instead of two as well.

In light of this, if a 3D model is developed in the future by Professor Zlotnik and his team, this 3D visualization could be modified to support a 3D simulation, and take the visualization of induced hydraulic fracturing to the next level.

6.8. Controversy

It is impossible to discuss induced hydraulic fracturing at length without briefly touching upon the controversy that "fracking" inspires. Although proponents of induced hydraulic fracturing argue for the benefits of natural gas over more conventional energy sources like coal and oil, and the economic advantages of this method of extraction, there are a few areas (as large as France) that have banned hydraulic fracturing [9].

The controversy with fracking revolves primarily around four main issues: water contamination, leaked methane, a market influence that could have a negative effect on climate change, and earthquakes.

When hydraulic fracturing is induced, not all of the fluid that is injected into the ground is always recovered. Sometimes the hydraulic fracturing fluid, which can contain unfriendly chemicals, leaches into the water table which can contaminate the local environment [10]. This would likely be the reason that hydraulic fracturing was banned from New York State in the Hudson watershed, on the logic that New York City's drinking water supply is more important than natural gas they can import from elsewhere.

Another concern expressed about induced hydraulic fracturing is that poor controls on operations can lead to the leaking of methane, a notorious greenhouse gas [11, 12]. Protections are being put into place to monitor this occurrence, but legislation only takes effect in the United States in 2015.

Advocated against induced hydraulic fracturing are also concerned that a more cost-effective and accessible energy source (natural gas) may lead to a higher demand and consumption, minimizing the benefits of cleaner burning fuel over coal in the eyes of the economy. These advocates believe that induced hydraulic fracturing has the possibility of crowding out renewable energy sources on the market, leading to an overall negative influence on climate change [13].

In addition to this, there is also some evidence that induced hydraulic fracturing might cause small earthquakes, which can potentially trigger larger earthquakes [10, 14].

Despite these arguments against induced hydraulic fracturing, there is still a great deal of optimism in the industry. The IEA (International Energy Agency) has offered a report on how to make induced hydraulic fracturing more environmentally friendly and socially acceptable endeavor (available for download here: http://www.worldenergyoutlook.org/goldenrules/#d.en.27023). In addition to this, many of the issues touched upon here could be mitigated by better monitoring and better understanding the process of induced hydraulic fracturing. If technicians could predict more accurately where the fluid will congregate, contamination and loss of fluid could be better controlled. This

emphasizes the importance of simulating induced hydraulic fracturing for a more environmentally conscious and more effective hydraulic fracturing process.

Reference List

1. Spence, D.A., Turcotte, D.L., Magma-driven propagation crack. *Journal of Geophysical Research 90* (1985), 575-580.
2. Fjaer, E., Holt, R.M., Horsrud, P., Raaen, A.M., Risnes, R. Petroleum related rock mechanics. *Elsevier series Development in Petroleum Science, vol 53, 2nd edition* (2008).
3. Duong, A.N., Phillips, C. Rate-Decline Analysis for Fracture-Dominated Shale Reservoirs. *SPE Reservoir Evaluation and Engineering 14, 3* (2011), 377-387.
4. Cleary, M.P., Barr, D.T., Willis, R.M. Enhancement of Real-Time Hydraulic Fracturing Models With Full 3-D Simulation. *SPE Gas Technology Symposium,* (1988).
5. Nordren, R.P., propagation of a Vertical Hydraulic Fracture. *Society of Petroleum Engineers* (1972), 306-314.
6. Khristianovic, S.A., Zheltov, Y.P. Formation of Vertical Fractures by Means of Highly Viscous Liquid. *Proceedings of the fourth world petroleum congress* (1955), 579-586.
7. Sneddon, I.N. The Distribution of Stress in the Neighbourhood of a Crack in an Elastic Solid. *Proceedings of the Royal Society A* (1946), 229-260.
8. Sangani, K. The Final Reel. *Engineering & Technology 7, 8* (2012), 32-35.
9. Wikipedia. "Hydraulic Fracturing by Country," last modified December 4, 2012, http://en.wikipedia.org/wiki/hydraulic_fracturing_by_country.
10. Garthwaite, J. The Science and Technology Behind the Natural Gas Boom. *Grist* (2012).
11. Lehner, P., October 15, 2012, " Fracking's Dark Side Gets Darker: The Problem of Methane Waste," *Switchboard: Natural Resources Defense Council Staff Blog,* http://switchboard.nrdc.org/blogs/plehner/frackings_darker_side_gets_darke.html.
12. Gavin, April 16, 2011, "Fracking Methane," *Real Climate: Climate science from climate scientists,* http:///www.realclimate.org/index.php/archives/2011/04/fracking-methane/.
13. Drum, K., September 7, 2012, "Is Fracking Good for the Environment?" *Mother Jones,* http://www.motherjones.com/kevin-drum/2012/09/fracking-good-environment.
14. Fischetti, M. Ohio Earthquake Likely Caused by Fracking Wastewater. *Scientific American* (2012).