Predictive Analysis of Cryptocurrency

Project Synopsis

of Major Project

Bachelor of Technology
(Information Technology)

Submitted By
DAMANPREET SINGH (1411247)
GURNOOR SINGH (1411254)

Guru Nanak Dev Engineering College
Ludhiana 141006

Abstract

OpenSCAD is an open source organization that serves a free software to create solid 3d CAD objects. OpenSCAD has in a way redefined how easy 3D modeling can be. But the Wikipedia article on OpenSCAD says that it is a non-interactive modeler, but rather a 3D compiler based on a textual description language.

In OpenSCAD when the object is described using the textual description language, the description is first compiled (following intermediate forms) into a node tree. This node tree consists, in geometric terms, the details of every component of the final design. Then follows the rendering process. This process is very CPU intensive and eats up a lot of cycles. This involves traverssing the node tree from the very bottom and evaluating nodes as they are encountered. Each evaluated node is then rendered on the screen.

In its present form, the compile and render processes are done in one thread under the one process. This is of course not the most optimal ways of going about this problem.

Adding multiple processor threading support to the compile and render feature will ensure that the system is run on 100 percentage on all available cores/ht rather than just the one.

This issue may be clear in intent but involves many complex considerations. One cannot just start implementing multithreading on these processes. The libraries which are being used for the rendering process are not thread safe (CGAL). Hence it becomes very difficult to implement this directly. Our project is as much about implementing a solution to this as it is about looking for one. Finding the best possible approach for achieving this is one big milestone that needs to accomplished before begining to realising that approach.

One of the original ideas was to create a dependency graph (as in a makefile) for the tree, and then traverse the graph, to keep the original tree const and to avoid multi-threaded access to the node tree. This seems a promising approach but remains to explored thoroughly.

Other issues pertaining to this project are adding features to halt an ongoing render process, fixing some low hanging grammer issues and fixing the mechanism that allows various nodes to be tagged as pseudo root nodes.

Acknowledgement

We, students of Guru Nanak Dev Engineering College, Ludhiana, have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

The authors are highly grateful to Dr. M.S. Saini Director, Guru Nanak Dev Engineering College, Ludhiana for providing them with the opportunity to carry out their Major project.

The author would like to wholeheartedly thank Amanpreet Singh Brar, Associate Professor, at Guru Nanak Dev Engineering College, Ludhiana who is a vast sea of knowledge and without whose constant and never ending support and motivation, it would never have been possible to complete the project and other assignments so efficiently and effectively.

Finally, we would like to thank our mentors at OpenSCAD organization Marius Kintel and Torsten Paul. Without their encouragement and Guidence, it would not have been possible to complete this project in such an efficient manner.

Amarjeet Singh Kapoor & Govind Sharma

CHAPTER 1

INTRODUCTION

1.1 Overview

Figure 1.1: OpenSCAD's logo

Multi-threaded compile and render for OpenSCAD is our major project. It is under the umbrella organization of BRL-CAD. OpenSCAD is a free and Open-source software application for creating solid 3D CAD objects. It is a script only based modeler, with a specific description language. Parts cannot be selected or modified by mouse in the 3D view. An OpenSCAD script specifies geometric primitives and defines how they are modified and manipulated to render a 3D model. OpenSCAD is available for Windows, Linux and OS X. It does constructive solid geometry (CSG).

OpenSCAD has in a way redefined how easy 3D modeling can be. But the Wikipedia article on OpenSCAD says that it is a non-interactive modeler, but rather a 3D compiler based on a textual description language. Pay attention to the above line, it is primarily what I will be talking about.

Solid 3D modeling. That sounds like some serious business. But it's just an awesome tool

for making models pertaining to many uses (mostly 3D printing). And 3D printing as we can all agree upon is cool. 3D models can be created by anyone using OpenSCAD. OpenSCAD is as much for designers as it is for you and me. What else can most people agree upon apart from the fact that solid 3D modeling is cool? A graphical interface is simpler and more intuitive to use. There is a general aversion for typing commands in order to get things done. Simply put, more people have an inclination towards GUI. Another concern is speed at which the results are presented before the user. It would be a waste of CPU capacity if not all of it is being used in the compilation and rendering of the models created in OpenSCAD. This can be achieved by spliting the process into multiple threads.

Since the rendering of various models internally done using the CGAL library, it becomes very important to check how the library behaves on a multi-threaded approach. This is something that will be considered before actually starting the implementation.

1.2 Project Category

OpenSCAD is a free and Open-source software application for creating solid 3D CAD objects. It is under the umbrella organisation of BRL-CAD. Since our project comes under OpenSCAD, it can be classified as being an application development project. But as much as it is an application development project, it is also a research project. It took a fair share of observation and study in order to find the right approach the problem. The majority of the time was spent in figuring out how the internals of this huge piece of software work. The study involved learning about various mathematical constructs used in the geometry of the models. Researching the computer graphics part of the sotware was also very essential in order to form a stronger understanding of how the problem in hand is to be tackled. As mentioned before, the CGAL library (which is used to render the models) is not entirely thread safe. So it bacame very crucial to analyse what parts of it are being used in the render process and how these parts actually work. Upon researching the above mentioned things, the right way was to be decided upon. The project essentially changes the design approach of an important segament of OpenSCAD i.e. rendering. So of course it required great consideration before implementing it.

1.3 Objectives of the Project

Objective of this project are following:

- Explore different OS independent ways of parallizing the evaluation.
- Support for multi-threaded evaluation, possibly with some limitations to handle non-thread-safe library calls
- Thread safe cache infrastructure

- Support for safe halting of the render process.
- Fixing some grammer related oversights in the modelling language.
- Installing warning mechanism in case of there being multiple pseudo node tags in the model.

1.4 Problem Formulation

The following considerations lead to the formation of issues that then transpired into this project:

- The rendering process was a bottleneck in performance.
- The time it took to render big models really made it cumbersome for the modelers to make small changes and watch their effect as part of their regular testing of the model.
- The software was not fully utilizing the cpu cores of the system on which it is run hence the efficeincy as well as time consumption was compromised.
- Since the rendering process involves traverssing, processing and then actuating on screen; nodes of an already formed tree; parallel processing the nodes was a clear idea moving forward.
- Having separate threads processing separate nodes in the tree was not a straightforward choice because the underlying library as well as the code structure of the software was not thread safe to satisfaction. By this it means that while processing various nodes, the methods used access the same variables in a non safe manner that can be used only in sequential access but will not work right in concurrent access. The same issue is with the CGAL library used in rendering. It also is not ideal for parallel processing.
- When the rendering command is given unintentionally then there must be a fault safe method of revoking this command without wipping out the already processed nodes from cache.
- The progress bar is not generalised for all kind of structural primitives. This restricts its potency.
- When nodes are traverssed in parallel, there is no control over the order in which they are traverssed. As such the progress report code in its current form will not be able to report the progress correctly.
- In assigning some nodes as pseudo root for certain renders, the user is not given any warning in case of there being multiple such tags in the description of the model.

- There are some oversights in the modelling language that allow for some incomplete syntax to go undetected. This needs to be fixed by improving the grammer.

All of the above things lead to the formation of issues that are attempted to be solved in this project.

1.5 The Existing System

OpenSCAD in its present form applies the compile and render process sequentially in one thread of a process. The whole process begins when the user write (or completes) the textual description of their model and chooses to either just compile it or complie and render it one go (F6). The textual description of the language is taken through the following stages sequentially.

1. The descriptional language is scanned, tokenized and converted into an Abstract Syntax Tree (AST). This is much like what any compiler does initially.

2. The AST is converted into an Abstract Node Tree. This consists of the nodes that are to become the part of the final model.

3. The node tree is fed to the CSGEvaluator which generates a preview of the model abstracting finer details.

4. The final rendering of the model is done by the GeometryEvaluator.

5. There are some cases where incorrect use of certain syntax is not caught. This may lead to unexpected behaviour.

6. When tagging certain nodes as root for the next render, the user is allowed to tag more than one node. In such cases, no warning is given to the user on how this may lead to unexpected behaviour.

7. The cancel button on the progress report widget does not work for CGAL rendering.

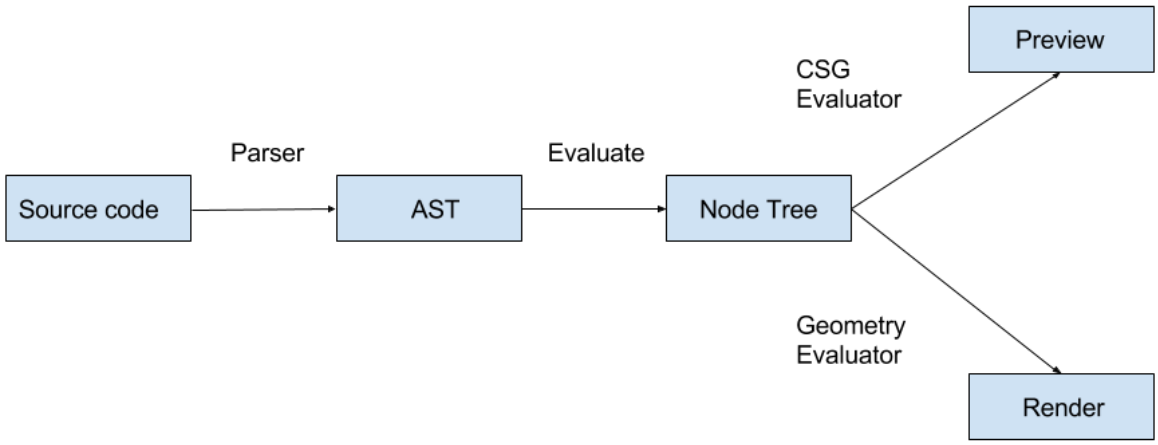

Figure 1.2: Present FlowChart

The present system is a bit slow in terms of its usage of the available CPU power. In being a single threaded process, everything has to happen in a sequence. This is not something which is required all the time. For example, during the evaluation of the node tree, various nodes can be evaluated in parallel in separate threads. Similarly at the time of rendering, this parallelism can be achieved.

Limitations of previous system

- Not utilizes the full potential of the CPU.
- The rendering process is very slow.
- When the user accidentely presses F6, there is no way to cancel that process and the user has to wait for the rendering to finish in order to continue their work.
- The system hangs frequently for more complex models on normal computers.
- It uses large amount of system resources. Their can be a significant improvement in this.

CHAPTER 2

FEASIBILITYSTUDY

2.1 Feasibility

Feasibility study aims to uncover the strengths and weaknesses of a project. These are some feasibility factors by which we can used to determine that the project is feasible or not:

- **Technical feasibility**: Technological feasibility is carried out to determine whether the project has the capability, in terms of software, hardware, personnel to handle and fulfill the user requirements. This whole project is based on Open Source Environment and is part of an open source software which would be deployed on any OS.
- **Economic feasibility**: OpenSCAD's multi-threaded compile and render is also Economically feasible with as It could be developed and maintain with zero cost as It is supported by Open source community. Plus This project is started with no intention of having any economic gain but still there is an option for donations. Economic Feasibility which can be categorized as follows:
    1. Development costs.
    2. Operating costs.

2.2 Significance of Project

Speed, concurrency, efficiency and better management of resources go a long way in improving any products' performance, usability and popularity. The same can certainly be said about an open source project like OpenSCAD. Our project will explore above aspects for OpenSCAD.

2.3 Objectives of the Project

Objective of this project are following:

- Explore different OS independent ways of parallizing the evaluation.
- Support for multi-threaded evaluation, possibly with some limitations to handle non-thread-safe library calls
- Thread safe cache infrastructure
- Support for safe halting of the render process.
- Fixing some grammer related oversights in the modelling language.
- Installing warning mechanism in case of there being multiple pseudo node tags in the model.

CHAPTER 3

PLANNING OF WORK

The basic implementation of this project will be done using prototype model and their will also be need to modify the structure of the project. Final flow of project will be fig. 3.1

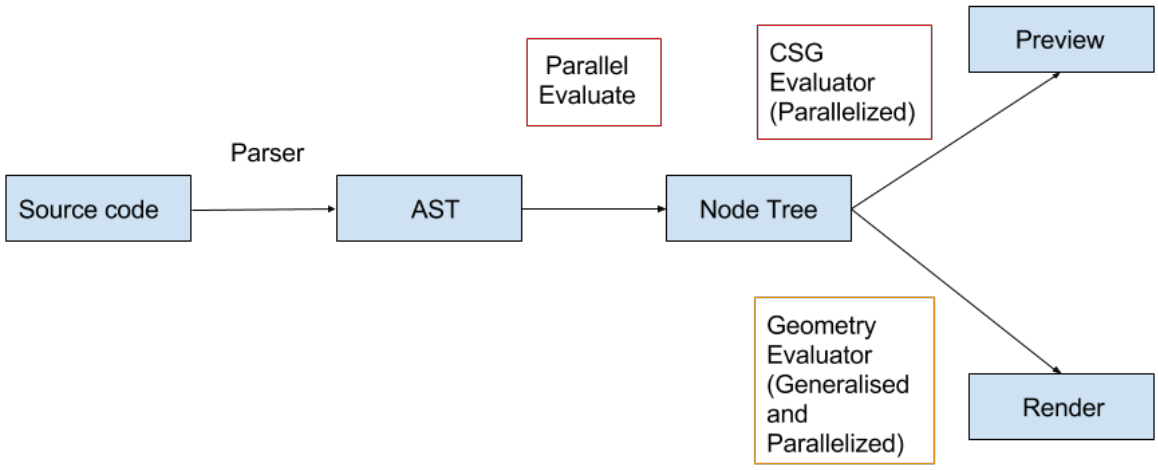

Figure 3.1:

3.1 Steps of Implementation

- The first thing that needs to be done is understanding the flow of the software and figuring out what sections of code needs to be parallelized.

- After forming a firmer understanding of the problem, it will be required to check all the relevant data structures involved in evaluating the tree nodes. It will also be checked if any data structures used are needed to modified or not.
- After fixing the appropriate data structures, all the feasible and plausible approaches will be explored. Each approach will be evaluated on its merits and feasibility.
- Among all the options the one which is works well across all platform will be chosen. It goes without saying that the chosen solution must also meet all the requirement and implement the solution efficiently.
- After going through all of this the implementation will begin.

CHAPTER 4

FACILITIES REQUIRED FOR PROPOSED WORK

4.1 Software Requirements

The following softwares may be used while developing and testing the software:

1. A C++ compiler supporting C++11
2. Qt4.4 $\rightarrow$ 5.x (http://qt.io/)
3. QScintilla2 (2.7 $\rightarrow$)(http://www.riverbankcomputing.co.uk/software/qscintilla/)
4. CGAL (3.6 $\rightarrow$) (http://www.cgal.org/)
5. GMP (5.x) (http://www.gmplib.org/)
6. MPFR (3.x)(http://www.mpfr.org/)
7. cmake (2.8 $\rightarrow$, required by CGAL and the test framework))(http://www.cmake.org/)
8. boost (1.35 $\rightarrow$) (http://www.boost.org/)
9. OpenCSG (1.3.2 $\rightarrow$)(http://www.opencsg.org/)
10. GLEW (1.5.4 $\rightarrow$)(http://glew.sourceforge.net/)
11. Eigen (3.x)(http://eigen.tuxfamily.org/)
12. glib2 (2.x)(https://developer.gnome.org/glib/)
13. fontconfig (2.10 $\rightarrow$)(http://fontconfig.org/)
14. freetype2 (2.4 $\rightarrow$)(http://freetype.org/)

15. harfbuzz (0.9.19 →)(http://harfbuzz.org/)
16. Bison (2.4 →)(http://www.gnu.org/software/bison/)
17. Flex (2.5.35 →)(http://flex.sourceforge.net/)
18. pkg-config (0.26 →)(http://www.freedesktop.org/wiki/Software/pkg-config/)

BIBLIOGRAPHY

[1] "OpenSCAD - News", Openscad.org, 2016. [Online]. Available: http://www.openscad.org/news.html#20160714. [Accessed: 27- Jan- 2016].

[2] "OpenSCAD - The Programmers Solid 3D CAD Modeller", Openscad.org, 2016. [Online]. Available: http://www.openscad.org. [Accessed: 27- Jan- 2016].

[3] "openscad/openscad", GitHub, 2017. [Online]. Available: https://github.com/openscad/openscad/wiki/Project%3A-Multi-threaded-geometry-rendering. [Accessed: 30- Jan- 2017].

[4] "OpenSCAD User Manual/WIP - Wikibooks, open books for an open world", En.wikibooks.org, 2016. [Online]. Available: https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/WIP#Customizer. [Accessed: 27- Jan- 2016].

[5] "openscad/openscad", GitHub, 2016. [Online]. Available: https://github.com/openscad/openscad. [Accessed: 27- Jan- 2016].

[6] "Doxygen: Main Page", Doxygen.org, 2016. [Online]. Available: http://www.doxygen.org. [Accessed: 27- Nov- 2016].

[7] "OpenSCAD", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/OpenSCAD. [Accessed: 27- Jan- 2016].

[8] "OpenSCAD", Openscad.org, 2017. [Online]. Available: http://www.openscad.org/. [Accessed: 16- Apr- 2017].

[9] "User Interface for Customizing Models", amarjeetkapoor1, 2016. [Online]. Available: https://amarjeetkapoor1.wordpress.com/2016/07/04/user-interface-for-customizing-models/. [Accessed: 27- Jan- 2016].

[10] "User:Amarjeet Singh Kapoor/GSoC2016/Project - BRL-CAD", Brlcad.org, 2016. [Online]. Available: http://brlcad.org/wiki/User:Amarjeet_Singh_Kapoor/GSoC2016/Project. [Accessed: 27- Nov- 2016].

[11] "User Interface for Customizing Models (Part 3)", amarjeetkapoor1, 2016. [Online]. Available: https://amarjeetkapoor1.wordpress.com/2016/08/17/user-interface-for-customizing-models-part-3/. [Accessed: 27- Nov- 2016].

[12] "openscad/openscad", GitHub, 2016. [Online]. Available: https://github.com/openscad/openscad/wiki/Project%3A-Form-based-script-parameterization. [Accessed: 27- Nov- 2016].

[13] "User Interface for Customizing Models (Part 2)", amarjeetkapoor1, 2016. [Online]. Available: https://amarjeetkapoor1.wordpress.com/2016/07/18/user-interface-for-customizing-models-part-2/. [Accessed: 27- Nov- 2016].

[14] "Qt (software)", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Qt_(software)#/media/File:Qt_logo_2015.svg. [Accessed: 27- Nov- 2016].

[15] "",Upload.wikimedia.org, 2016. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/c/ce/Doxygen.png. [Accessed: 27- Nov- 2016].

[16] "Git", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Git#/media/File:Git-logo.svg. [Accessed: 27- Nov- 2016].

[17] "Qt (software)", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Qt_(software). [Accessed: 27- Nov- 2016].

[18] "C++", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/C%2B%2B. [Accessed: 27- Nov- 2016].

[19] "Travis CI", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Travis_CI. [Accessed: 27- Nov- 2016].

[20] "CMake/Testing With CTest - KitwarePublic", Cmake.org, 2016. [Online]. Available: https://cmake.org/Wiki/CMake/Testing_With_CTest. [Accessed: 27- Nov- 2016].

[21] "", I-msdn.sec.s-msft.com, 2017. [Online]. Available: https://i-msdn.sec.s-msft.com/dynimg/IC251606.jpeg. [Accessed: 16- Apr- 2017].

[22] "Lambda Expressions in C++", Msdn.microsoft.com, 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/dd293608.aspx. [Accessed: 16- Apr- 2017].

[23] "Using Lambda Expressions for Shorter, More Readable C++ Code – Visual Studio Magazine", Visual Studio Magazine, 2017. [Online]. Available: https://visualstudiomagazine.com/articles/2012/06/18/cplus-lambda-expressions.aspx. [Accessed: 15- Apr- 2017].

[24] "Visitor pattern", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Visitor_pattern. [Accessed: 17- Apr- 2017].

[25] "Visitor pattern", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Visitor_pattern#/media/File:Visitor_design_pattern.svg. [Accessed: 15- Apr- 2017].

[26] "Boost (C++ libraries)", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries). [Accessed: 17- Apr- 2017].

[27] "GNU", En.wikipedia.org, 2016. [Online]. Available:

https://en.wikipedia.org/wiki/GNU#/media/File:Heckert_GNU_white.svg. [Accessed: 27- Nov- 2016].

[28] "GitHub", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/GitHub. [Accessed: 15- Apr- 2017].

[29] "Layer 1", Upload.wikimedia.org, 2017. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/2/24/GitHub_logo_2013_padded.svg. [Accessed: 16- Apr- 2017].

[30] "", Upload.wikimedia.org, 2017. [Online]. Available: https://upload.wikimedia.org/wikipedia/comm [Accessed: 17- Apr- 2017].

[31] "Flex (lexical analyser generator)", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Flex_(lexical_analyser_generator). [Accessed: 22- Apr- 2017].

[32] "GNU bison", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/GNU_bison. [Accessed: 17- Apr- 2017].

[33] "JSON logo", Upload.wikimedia.org, 2017. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/c/c9/JSON_vector_logo.svg. [Accessed: 18- Apr- 2017].

[34] "JSON", Json.org, 2017. [Online]. Available: http://www.json.org/. [Accessed: 22- Apr- 2017].

[35] "STL Function Objects (Functors)", Cs.stmarys.ca, 2017. [Online]. Available: http://cs.stmarys.ca/ porter/csc/ref/stl/function_objects.html. [Accessed: 15- Apr- 2017].