Formal Proposal:
Application of Transmission Line Matrix Theory on Modern Day Graphics Processing Unit Hardware

BSc Software Engineering

Author Robert Jones

October 30th 2006

CONTENTS

Contact Details

Surname	Jones	First Name(s)	Robert Ian
Home Telephone Number	----------	Work Telephone Number	N/A
E-mail address	rob@phantom-web.co.uk		

Project Proposal:
Application of Transmission Line Matrix Theory on Modern Day Graphics Processing Unit Hardware

Introduction

Transmission Line Matrix (TLM) Theory allows for the simulation and modelling of wave propagation, primarily electromagnetic in nature. To date this simulation has been performed on the CPU of a computer system and due to their heavy computational nature are slow to complete, indeed for large matrices a real time solution might not even be possible.

The traditional solution to speeding up this kind of operation is to either install a faster CPU or employ a system of linking numerous computers together to form a cluster to complete the operation in parallel. However CPU speed increases are slowing down and the purchasing of clusters is an expensive task.

At the same time there has been significant advancement in graphics processor technology, from the beginnings as fast rasterizer to the massively parallel and programmable devices we have today.

Games take advantage of this programmability to offload tasks from the CPU to the GPU and the aim of this project is to investigate the application of this method of programming with the aim of speeding up the TLM simulation so that large matrices can be evaluated in real time, if not close to.

The project builds upon work carried out by Steve Harris and Kristian Dor and will be using some of their work as a foundation and, in the case of Kristian's final produce, a jumping off point for the rest of the project.

My main reason for choosing this project to work on is that it allows me to pursue my own interests in computer programming, GPU programming and visualisation which have been a hobby of mine for many years now. This project also lets me extend my own knowledge and learn some new and cutting edge programming techniques.

Project Aims

The purpose of this project is to investigate the potential gains in off loading the TLM simulations from the CPU to the GPU. Various techniques will be used with the over all aim to show which method has potentially the best gain and thus the best route to pursue when working with modern day hardware.

Project Deliverable

The final deliverable will be a piece of software which will be used to both benchmark and visualise the various techniques used to perform the TLM simulation.

The software will have a GUI interface allowing for real time changes in the method of simulation as well as details of the simulation being run, including but not limited to the size of the simulation.

The applications target audience are both academics and game programmers, as the application of TLM theory can be considered of interest to both parties; for academics the simulation and visualisation of the TML simulation will be of use to people studying in this particular field, while games programmers can make use of the dynamic surface generation for real time effects in games, such as shockwaves and other such phenomenon which might exist in a game world.

The deliverable will be programmed mostly in C++ as this is the language I have the most experience in, however a 'shading language' of some sort will be required to communicate with the GPU and I also have not ruled out the usage of a scripting language, such as Lua, for configuration purposes to cut down on the need to recompile during debug cycles. I have used Lua during a commercial project, as such while it is not my most favoured language I do have enough knowledge to work with it for this project.

Project Objectives

To carry out this investigation the project has been broken down into a number of objectives, each one building upon the one preceding it.

Objective: Investigate TLM Theory
Time Limit: 2 weeks

While a full investigation of TLM Theory might not be possible in the time span of the project to successfully translate the algorithm from the CPU to the GPU a reasonably in depth understanding of it will be required.

As there is no real measurement which can be taken to indicate when 'a reasonably in depth understanding' has been achieved a hard time limit, which should be longer than required, has been assigned to this objective. It will be down to judgement as to if I can move on from this objective before the time limit has been achieved.

As this is central to the whole project it is important that the TLM theory is understood to a decent level within the time frame as failure to do so will impact the timing of the rest of the project which follows.

Objective: Select and become familiar with the appropriate APIs
Time Limit: 3 weeks

While I do have a lot of familiarity with a 3D API, namely OpenGL, this might not suit this particular task. This is due to the majority of the reference material I have found which talk about performing a particular set of operations, which will be required for this project , refer to Direct3D (D3D) as the API of choice. Due to this it might be required to learn D3D to a significant level beyond my current passing familiarity with it, which in the event of using OpenGL will be enough for any translation from Kristian's D3D version to OpenGL for this code.

An in depth knowledge of the API will not be required; however the means to perform the various operations to allow for the GPU to produce the required data will be the main focus of the learning.

Support API's refer to any input APIs required; such as keyboard and mouse, as well as any other API which might be required during the development of this project which I do not already have some familiarity with.

Objective: Produce a basic framework to work within
Time Limit: 1 week

While each technique could have a separate project designed and developed for it the idea of developing a core framework which all the code can use would seem to be the best idea with regard to work load and code reuse.

This framework could exist as a library of code which the techniques can link to and thus produce the final application or the library might well be the application and the various techniques instead are loaded by it from some kind of dynamic system, such as a DLL.

This framework however should not over reach it's self, as while it would be desirable to be able to reuse this code at a later date for the purposes of testing and development the minimum of functionality is required;

Required functionality:

- Open a window to render into
- Run the TLM simulation
- Be able to collect information with regards to the performance of the simulation (as detailed after the objectives section)
- Be able to clean up and shut down cleanly.

This section will be considered complete one this basic functionality is completed.

Objective: Adapt Kristian's application code to framework
Time limit: 2 weeks

As previously noted this project will be building upon previous work carried out. As a baseline for the following algorithms Kristian Dor's code will be adapted into the framework to allow us to carry out a comparison.

This will require the code to be translated from C# and Managed D3D to C++ and which ever graphics API is selected.

Once this code has been translated, depending on the time left for this step, some attempt to improve the performance with Single Instruction Multiple Data (SIMD) programming techniques and threaded programming via the OpenMP API to take advantage of multi-core CPUs might be carried out. This will allow for comparisons to be made between all the CPU versions of the code with the later GPU versions.

Once the code is working in the new framework this section will be considered completed.

Objective: Produce a CPU/GPU hybrid implementation of the algorithm
Time Limit: 2 weeks

As a logical progression from the pervious object the first step of conversion to a fully GPU dependant algorithm will be the development of a hybrid one; the TLM is generated on the CPU and the GPU is used to generate the vertex data required to visualise the simulation.

This will allow the reuse of the code from the previous object to carry out the simulation while at the same time allowing the computational power of the GPU to be used to generate the mesh data to be drawn afterwards.

The reasoning behind this is that it allows for the minimal amount of changes to be made between versions and thus reduce the complexity of the change and lessen any bugs or other problems which might have occurred from going straight from a CPU simulation to an all out GPU simulation.

Once the simulation is running correctly within the framework this section will be considered completed.

Objective: Produce a GPU only implementation of the algorithm
Time Limit: 2 weeks

Having successfully completed a hybrid application in the previous objective the next logical step is to move the simulation completely to the GPU so that all the steps are carried out with the CPU only being involved to setup the GPU for each stage of the operation.

This stage represents the completion of the primary objectives of this project as the TLM simulation is now running completely on the GPU.

Objective: Produce a D3D10 implementation of the algorithm
Time Limit: 2 weeks

This objective is considered optional and depends on the time remaining for the project to be completed.

D3D10 is the new programming API for 3D graphics coming with Windows Vista and as part of this is a new section being introduced to the GPU; a geometry unit.

Unlike the vertex units on current GPU, which take on vertex in and push one vertex out the geometry unit can take one vertex in and output many vertices depending on the program which is being run on it. It is envisioned that this geometry unit, along with other D3D10 enhancements to the pipeline, might allow for better performance when it comes to running the TLM simulation on the GPU as it might allow for the reduction of the algorithm from three passes (TLM construct, geometry construct and

final render) to a single all encompassing pass which performs the whole TLM simulation and produces the final output.

However, due to D3D10 and Windows Vista still being in the beta stage and the lack of D3D10 capable hardware on the market this objective might be left out if time and resources do not allow for it. (While the simulation could be run via software emulation using the D3D10 and Windows Vista betas this could be considered too slow for practical work)

Objective: Produce a 'Close to the metal' implementation of the algorithm
Time Limit: Project long + 2 weeks implementation time

In this object "Close to the metal" (CTM) refers to an ATI developed API which, unlike the API used in pervious objectives, allows for the application to make use of the GPU without requiring to go via a graphics API.

This system is in beta, although currently being used by the Folding@Home project to great effect, and is a completely new system to me. For this reason learning how to use this API will be a project long objective with the final 2 weeks being allowed for implementing the code into framework.

This method is being investigated as it could potentially be the best performing version of the simulation as it by passes the graphic API layer and talks directly to the hardware (via a small abstraction layer) allowing for more application control over the system. At the same time this is likely to be the hardest to develop.

Much like the D3D10 objective this objective relies on there being sufficient time for it to be implemented and it takes a secondary position to other work during the project due to the bleeding edge nature of this area of programming and research. Unlike the D3D10 objective however this will work on current hardware available to me.

This objective might also lack a visual aspect to it, however this is more aimed at academics requiring raw numbers than those who are interested in the visualisation side of things.

Objective: Produce a final application
Time Limit: 2 weeks

Building on the earlier framework developed the final application, complete with GUI, will be the last thing designed and developed so that real time changes can be made to the running application.

This is the final step as the readings can be taken with the earlier framework and as such isn't critical to benchmarking the system, however the final application will allow for easier changes to be made to the simulation without restarting it after changing the configuration settings.

Data Collection and Analysis

To ensure the data which is collected can be compared all the simulation will have the same data collected for them. The performance of each algorithm will then be evaluated based on this collected data.

The following metrics will be used to compare the algorithm implementations;

- Frames per second (or over all frame time)
- Time to generate TLM
- Time to generate mesh data
- Image Quality
- Number of vertices rendered/generated
- Memory usage
- CPU usage

In the case of the CTM version if a final visualisation is not produced then only the TLM and mesh data generation times will be used and compared with the other implementations, not the over all frame time. The rest of the metrics will be looked at for all simulations.

Image quality will be both a subjective visual comparison as well as the difference between each image being calculated to show up any differences.

Indicative Bibliography

In preparation for this project I have looked at and for a number of sources which will be of use during the project's lifetime.

My main initial source was the previous work done by Kristian Dor [Dor, 2006], this was the basis for my thoughts about being able to extend the TLM Theory from a CPU based algorithm to a GPU based one.

The bibliography contained within the aforementioned work has given me two further sources for TLM Theory related works;

The first of these was The Transmission Line Modelling Method [Christopoulos, 1995] which I believe will be the perfect source to improve my understanding of TLM Theory and modelling.

The second was 'TLM and Shading Algorithms – Some Thoughts' [Harris], which I am hoping will provide some insight into visualisation techniques already employed for TLM visualisation.

From a technical stand point my main source of information has come from a series of ATI PDF documents.

The first of these, "R2VB Programming" [Persson E, 2006], is an introduction to render-to-vertex buffer programming for the D3D API. This is going to be needed as it forms the basis of the GPU method of performing the algorithm.

The second PDF, ATI CTM Guide Beta [ATI, 2006] of interest details the 'Close To The Metal' method of programming the GPU (as mentioned in the above objectives). This is a technical manual giving all the details on the way the GPU works at this level of abstraction so that the reader can produce their own CTM code.

As mentioned in the objectives above it might be required for me to improve my knowledge of D3D, as such the DirectX Software Development Kit (SDK) [MS, 2006] is a must have resource containing both examples and documentation so that I can learn how the API works.

When it comes to visualising the TLM simulation some kind of programmable shader is going to be required, with this in mind the book 'Advanced Lighting and Materials with Shaders' [Dempski & Viale, 2005] looks to be a good source of information for implementing the various lighting schemes which could be applied.

The final source of general information with regards to 3D rendering and the 3D pipeline is a text which is considered a 'must have' for any 3D programmer; Real-Time Rendering [Akenine-Möller & Haines, 2002]. This tome covers every aspect of 3D rendering which will be required in this project and then some and is an excellent source of general information.

Bibliography

Akenine-Möller & Haines, 2002 **Real-time Rendering**, second edition, A K Peters, Ltd, Natick, MA. ISBN: 156881-128-9

ATI, 2006 **ATI CTM Guide: Technical Reference Manual**, available from http://ati.amd.com/companyinfo/researcher/documents/ati_ctm_guide_beta.pdf

Christopoulos, 1995 **The Transmission-Line Modelling Method**, IEEE Press/Oxford Press, 1995

Dempski & Viale, 2005 **Advanced Lighting and Materials with Shaders**, Wordware Publishing, Inc. Plano, Texas. ISBN: 1-55622-292-0

Dor, 2006 **Dynamic Surfaces using Transmission Line Matrix Theory**, Suffolk College, Ipswich, Suffolk

Harris **TLM and Shading Algorithms – Some Thoughts**, Suffolk College, Ipswich, Suffolk

MS, 2006 **DirectX SDK**, available from http://msdn.microsoft.com/directx/

Persson E, 2006 **R2VB Programming**, available as part of the March 2006 SDK from ATI at http://ati.amd.com/developer/radeonSDK.html