

ORIGIN-CENTRIC TECHNIQUES FOR OPTIMISING  
SCALABILITY AND THE FIDELITY OF MOTION,  
INTERACTION AND RENDERING

By  
Chris Thorne  
BSc(Hons), MSc

SUBMITTED IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY  
AT  
THE UNIVERSITY OF WESTERN AUSTRALIA  
CRAWLEY, WESTERN AUSTRALIA  
DECEMBER 2007

© Copyright by Chris Thorne  
BSc(Hons), MSc, 2007

THE UNIVERSITY OF WESTERN AUSTRALIA  
SCHOOL OF  
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Engineering, Computing and Mathematics for acceptance a thesis entitled “**Origin-Centric Techniques for Optimising Scalability and the Fidelity of Motion, Interaction and Rendering**” by **Chris Thorne BSc(Hons), MSc** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Dated: December 2007

Research Supervisors: \_\_\_\_\_  
Associate Professor Amitava Datta

\_\_\_\_\_  
Dr Nick Spadaccini

THE UNIVERSITY OF WESTERN AUSTRALIA

Date: **December 2007**

Author: **Chris Thorne**  
**BSc(Hons), MSc**

Title: **Origin-Centric Techniques for Optimising**  
**Scalability and the Fidelity of Motion, Interaction**  
**and Rendering**

School: **Computer Science and Software Engineering**

Degree: **Ph.D.**      Graduation: **September**      Year: **2008**

Permission is herewith granted to The University of Western Australia to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

---

Signature of Author

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION.

THE AUTHOR ATTESTS THAT THIS THESIS IS THE AUTHOR'S OWN COMPOSITION, THAT PERMISSION HAS BEEN OBTAINED FOR THE USE OF ANY COPYRIGHTED MATERIAL APPEARING IN THIS THESIS (OTHER THAN BRIEF EXCERPTS REQUIRING ONLY PROPER ACKNOWLEDGEMENT IN SCHOLARLY WRITING) AND THAT ALL SUCH USE IS CLEARLY ACKNOWLEDGED.



# Abstract

This research addresses endemic problems in the fields of computer graphics and simulation such as jittery motion, spatial scalability, rendering problems such as z-buffer tearing, the repeatability of physics dynamics and numerical error in positional systems. Designers of simulation and computer graphics software tend to map real world navigation rules onto the virtual world, expecting to see equivalent virtual behaviour. After all, if computers are programmed to simulate the real world, it is reasonable to expect the virtual behaviour to correspond. However, in computer simulation many behaviours and other computations show measurable problems inconsistent with real-world experience, particularly at large distances from the virtual world origin.

Many of these problems, particularly in rendering, can be imperceptible, so users may be oblivious to them, but they are measurable using experimental methods. These effects, generically termed spatial jitter in this thesis, are found in this study to stem from floating point error in positional parameters such as spatial coordinates. This simulation error increases with distance from the coordinate origin and as the simulation progresses through the pipeline. The most common form of simulation error relevant to this study is *spatial error* which is found by this thesis to not be calculated, as may be expected, using numerical relative error propagation rules but using the rules of geometry.

The endemic nature of the problem to simulation, evidence of poor understanding of both the problem and its cause were the motivation for this study. The aim of this research was to find a general and optimal solution that would improve many aspects of simulation while accommodating the constraints of the software and hardware

environment.

The approach taken was to study the fundamental nature of spatial systems, derive a general and optimal approach and perform experiments to build a complete understanding of all issues arising from spatial error. The experiments included a case study applying origin centric techniques to ascertain their effectiveness. A new optimising simulation architecture was defined to show how and where origin centric techniques can be applied in a full system context. Knowledge of relative error propagation and spatial error were then combined to show how error in a spatial simulation can be calculated.

The results of this thesis show how a better understanding of the nature of simulation space, new origin centric techniques and a redesign of the simulation pipeline can achieve optimal scalability, repeatability and fidelity throughout virtual environments. Geospatial simulation is used as the main application case study to show how the properties of simulation space can be exploited to remove observable spatial jitter because both its demands for large environments and its reliance on measurement input magnify spatial jitter. A number of pathological cases that magnify spatial error were also found. It was further shown that spatial jitter grows exponentially with distance from the origin.

The thesis shows that the thinking behind real-world rules, such as for navigation, has to change in order to properly design for optimal fidelity simulation. Origin-centric techniques, formulae, terms, architecture and processes are all presented as one holistic solution in the form of an optimised simulation pipeline. The results of analysis, experiments and case studies are used to derive a formula for relative spatial error that accounts for potential pathological cases. A formula for spatial error propagation is then derived by using the new knowledge of spatial error to extend numerical relative error propagation mathematics. Finally, analytical results are developed to provide a general mathematical expression for maximum simulation error and how it varies with distance from the origin and the number of mathematical operations performed.

We conclude that the origin centric approach provides a general and optimal solution to spatial jitter. Along with changing the way one thinks about navigation, process guidelines and formulae developed in the study, the approach provides a new paradigm for positional computing. This paradigm can improve many aspects of computer simulation in areas such as entertainment, visualisation for education, industry, science, or training. Examples are: spatial scalability, the accuracy of motion, interaction and rendering; and the consistency and predictability of numerical computation in physics. This research also affords potential cost benefits through simplification of software design and code. These cost benefits come from some core techniques for minimising position dependent error, error propagation and also the simplifications and from new algorithms that flow naturally out of the core solution.

# Acknowledgements

I would like to thank Associate Professor Amitava Datta, my principle supervisor, for reviewing my papers and thesis, for insightful questions during my seminar that helped me frame the explanations in my thesis; and for giving me just the right technical advice and guidance in preparing the final draft of my proposal. I am also thankful to my co-supervisor, Dr Nick Spadaccini, for reviewing and commenting on my proposal and thesis.

Dr Andrew Squelch and Dr Ken Fowle showed continued interest and support and for suggesting some useful references, all of which was greatly appreciated.

My thanks also to the Western Australian Premier's Collaborative Research Project for providing scholarship support for my studies and for providing the opportunity to carry out a case study that provided useful results.

A good portion of this work's contributions to research have been presented in full and short papers at international conferences, workshops and meetings and some local conferences and workshops. Public has helped greatly in honing the text of this work and the many people I have met have provided additional insight into the concepts surrounding the work.

Of course, I am grateful to my wife and daughters for their patience and *love* without which this work would have been a far more difficult endeavor.

Finally, I wish to thank the following:  
My Mother, Myrtle Andrews, for reviewing drafts and help with Latex! Johnathan Knispel for supporting and believing in my work from the beginning, for friendship, good advice and pointers to relevant work at times during my thesis.  
Viveka Weiley for his long term support for my sometimes weird ideas.

Perth, Western Australia  
December 10, 2007

Chris Thorne

# Publications Arising from Thesis

- a. THORNE, C. 2008. *Effects of Spatially Dependent Error on Rendering, Interaction and Motion in Simulation Worlds*, accepted for publication in: *Journal of Ubiquitous Computing and Intelligence: Ubiquitous Computing in Cyberworlds*, Issue 2.
- b. THORNE, C. 2006. *Error Minimizing Pipeline for Hi-Fidelity, Scalable Geospatial Simulation*, Proceedings of the International Conference on Cyberworlds, Lausanne, Switzerland, November 28-29, pp 81-88.
- c. THORNE, C. 2005. *Using a Floating Origin to Improve Fidelity and Performance of Large, Distributed Virtual Worlds*, Proceedings of the International Conference on Cyberworlds, Nanyang Technical University, Singapore, 23-25 November, pp. 263-270.
- d. THORNE, C. 2005. *Using a Floating Origin to Improve Fidelity and Performance of Large, Distributed Virtual Worlds*, Proceedings of the 14th University of Western Australia School of Computer Science and Software Engineering Research Conference, Rottnest Island, Western Australia, 20-21 September.
- e. THORNE, C. 2005. *Exploiting an Evolutionary Accident in Web3D Communications to Integrate Application Components*, Proceedings of the SIGGRAPH 2005 conference on Web graphics: in conjunction with ACM SIGGRAPH: Web Graphics, Session: 3D, Conference CD ROM, Los Angeles,

CA, 31 July - 4 August.

- f. THORNE, C. 2004. *The Kata of Web3D*. Proceedings of the SIGGRAPH 2004 conference on Web graphics: in conjunction with ACM SIGGRAPH: Web Graphics, Session: 3D, Conference CD ROM, Los Angeles, CA, 8-12 August.

# Table of Contents

<b>Abstract</b>	<b>v</b>
<b>Acknowledgements</b>	<b>viii</b>
<b>Publications Arising from Thesis</b>	<b>ix</b>
<b>Table of Contents</b>	<b>xi</b>
<b>List of Figures</b>	<b>xv</b>
<b>Introduction</b>	<b>1</b>
1.1 Overview of Research Problem: Spatial Jitter and Scalability in Visualisation and Simulation . . . . .	1
1.2 Research Hypothesis . . . . .	3
1.3 Research Outline . . . . .	4
1.4 Document Outline . . . . .	7
<b>2 Review of the Literature</b>	<b>9</b>
2.1 Background on Simulation Fidelity and Scalability Issues . . . . .	10
2.1.1 Preliminary Definitions and Propositions . . . . .	13
2.2 Prior Approaches to Spatial Jitter . . . . .	22
2.2.1 Subdivision - Multiple Local Coordinate Systems . . . . .	23
2.2.2 On-the-fly Shifting of Coordinates . . . . .	25
2.2.3 Piecewise Shifting of Coordinates in a Continuous Virtual World	28
2.3 Further Discussion on Conventional Approaches . . . . .	31
2.3.1 The Precision Misinterpretation . . . . .	31
2.3.2 Difficulty in Understanding the Jitter Problem . . . . .	34
2.4 Other Effects of Positional Floating Point Error . . . . .	35
2.5 The Simulation Pipeline Context . . . . .	37

2.5.1	Simulation Output and the Graphics Pipeline . . . . .	37
2.5.2	The Distributed Simulation Pipeline . . . . .	38
2.6	Summary . . . . .	41
<b>3</b>	<b>Solution: The Origin Centric Approach</b>	<b>46</b>
3.1	The Floating Origin Technique . . . . .	47
3.1.1	Reversal of Navigation Thinking . . . . .	47
3.1.2	Past Difficulties Understanding the Spatial Jitter Problem and a Solution . . . . .	49
3.1.3	Design for Floating Origin Navigation and Rendering . . . . .	50
3.2	Level of Detail . . . . .	55
3.3	Progressively Refined Fidelity (PRF) . . . . .	57
3.4	Time Error Minimisation . . . . .	58
3.5	Dynamic Clip Plane Management . . . . .	60
3.6	Comparison of Floating Origin with Traditional Approaches . . . . .	61
3.6.1	Design Complexity and Performance Costs . . . . .	61
3.6.2	Performance and Efficiency Trade-offs . . . . .	62
3.7	Coordinate Systems and Representation . . . . .	79
3.7.1	Scalability . . . . .	81
3.8	Issues to Address when Applying a Floating Origin Design . . . . .	82
3.9	Characteristics of Position Sensitive Floating Point Error in Simulation Space . . . . .	83
3.10	Characteristics of Nonuniform Discrete Digital Spaces and Spatial Error	87
3.11	Summary . . . . .	89
<b>4</b>	<b>Experiments and Case Studies</b>	<b>94</b>
4.1	Measurement of Spatial Jitter with Distance From Origin . . . . .	95
4.1.1	Description . . . . .	95
4.1.2	Procedure . . . . .	97
4.1.3	Results . . . . .	98
4.1.4	Discussion . . . . .	99
4.2	Effect of Spatial Error on Rendering Accuracy . . . . .	102
4.2.1	Without a Rendering Pipeline, With Double Precision . . . . .	104
4.2.2	With Rendering Pipeline, Single Precision . . . . .	118
4.2.3	Low Level OpenGL Rendering . . . . .	122
4.2.4	Discussion . . . . .	127
4.3	Effect of Spatial Error on Z buffer Clipping and Ordering . . . . .	127
4.3.1	Measurement of Tearing Separation Distance Versus Distance from Origin . . . . .	128

4.3.2	Large Object at Great Distance . . . . .	136
4.4	Effect of Spatial Error on Physics Calculations and Visualisation . . .	142
4.4.1	Description . . . . .	142
4.4.2	Procedure . . . . .	142
4.4.3	Results . . . . .	142
4.4.4	Discussion . . . . .	144
4.5	Dynamic Effects of Spatial Error . . . . .	145
4.5.1	Motion Jitter . . . . .	146
4.5.2	Dynamic Effects of Spatial Error 2: Sensors and Interaction .	148
4.6	Execution Performance Measurements . . . . .	152
4.6.1	Graphics Transformations . . . . .	152
4.6.2	Distance and Proximity Calculations . . . . .	154
4.6.3	Summary . . . . .	156
4.7	Case Study: Applying Origin-Centric Approach to Large Scale, Dis- tributed Geospatial Application . . . . .	158
4.7.1	Description . . . . .	158
4.7.2	Procedure . . . . .	158
4.7.3	Results . . . . .	160
4.7.4	Discussion . . . . .	160
4.8	Case study: Applying Origin-Centric Concepts to Geospatial Data Preparation . . . . .	160
4.8.1	Description . . . . .	160
4.8.2	Procedure . . . . .	162
4.8.3	Results . . . . .	162
4.8.4	Discussion . . . . .	163
4.9	Case study: GeoVRML Error Estimation . . . . .	166
4.10	Summary of Experimental Results . . . . .	167
<b>5</b>	<b>An Optimising Simulation Pipeline and the Determination of Max- imum Simulation Error</b>	<b>170</b>
5.1	Incorporating Origin-Centric Techniques into the Simulation Pipeline	170
5.2	An Optimising Simulation Pipeline . . . . .	171
5.3	Origin-Centric Processes - Throughout the Pipeline . . . . .	175
5.3.1	Data Preparation and Application Processes . . . . .	175
5.3.2	Lazy Evaluation . . . . .	176
5.4	Fitting Origin Centric Techniques into the Pipeline . . . . .	180
5.4.1	Simple Implementation of Origin Centric Techniques in VRML	183
5.5	General Simulation Error Mathematical Results . . . . .	191

5.5.1	Special Cases Affecting Relative Spatial Error Calculation . . .	192
5.5.2	Combining Spatial Error and Conventional (Apositional) Relative Error . . . . .	197
5.5.3	Further Discussion . . . . .	203
5.6	Summary . . . . .	205
<b>6</b>	<b>Conclusion and Future Work</b>	<b>207</b>
6.1	Conclusion . . . . .	207
6.1.1	Summary of contributions . . . . .	211
6.2	Future Work . . . . .	215
6.2.1	Interaction and Perception Experiments . . . . .	215
6.2.2	Investigation into the nature and effect of Benford error . . . .	216
6.2.3	Spatial Variability . . . . .	216
	<b>Bibliography</b>	<b>217</b>

# List of Figures

1.1	Snapshots of two moving dots overlaid into one image. The dots should move in a smooth, straight path. However, their motion is jittery because it occurs far from the origin at $(1 \times 10^6, 1 \times 10^6, 1 \times 10^6)$ . Each upper snapshot sequence of dots is connected by a line, tracing its jittery path. A straight line underneath highlights the uneven motion of the dots. . . . .	2
2.1	An illustration of how the representable floating point numbers are spaced with increasing gaps the further they are from the origin. . . .	10
2.2	Components of the IEEE single precision floating point representation showing sign bit, exponent and mantissa. . . . .	11
2.3	Illustration of rendering a line. The mathematical model of line AB is perspectively projected onto a view plane such that it appears as it would be seen (A'B') in perspective from the viewpoint. . . . .	20
2.4	Changing the location of a rendering calculation. The eye and line AB can be shifted equally by $(-200,0,0)$ without changing the projection of the line onto the view plane. The projection operation is position independent. . . . .	21
2.5	A generic simulation pipeline showing the flow of information from input to output. As processing moves to the output stages the pipeline has decreasing precision and increased realtime constraints. . . . .	43

- 2.6 A generic graphics pipeline showing graphical models as input to the graphical *scene*. The scene is animated through a number of operations such as rotation, translation and view transformations and is then rendered from the users’s point of view. . . . . 43
- 2.7 A generic distributed simulation pipeline subdivided into Object System (OS) and Display System (DS), signified by the solid black boxes. The graphics pipeline overlaps early stages of the simulation because the graphical models are part of the input and graphical information can be interchanged between peer clients or servers across the network. 44
- 2.8 A generic distributed simulation pipeline showing the progression of coordinate spaces down the pipeline. . . . . 45
- 3.1 In an origin-centric world, the user remains at the origin while navigation reverse-translates the world. . . . . 48
- 3.2 Centered viewpoint navigation: in the scene on the right, when a centered viewpoint is selected, the eye remains at the origin and objects (represented by a cone) are moved towards the viewer. The display system World Coordinate (WC) of the object changes whereas its object system Application Coordinate (AC) does not. . . . . 48
- 3.3 Structure of the client in a distributed system, showing how centered viewpoint and navigation components affect the WT which, in turn, transforms objects. When objects are to take an active role in a scene, they are copied from the Object System (OS) store and added to the active scene in the Display System (DS). Once active objects have been positioned, reverse navigation deltas sent to active object framework keep their positions updated. Note how other navigation and behaviours do not affect the WT but act directly on the objects. . . . 52

- 3.4 As an observer navigates towards a stationary aircraft, the observer actually remains stationary and, as the arrow shows, the aircraft is moved toward the observer. In the process, the level of detail of the aircraft increases in steps until it is at its most detailed close to the observer. When employing PRF (14), the position coordinates of the object are recalculated each time (including composing with the reverse transform) with progressively greater accuracy. The final coordinates of the object sent to the graphics hardware, become smaller with decreasing distance from observer and thereby have greater accuracy at a given precision. . . . . 56
- 3.5 Architecture of the distributed virtual world application showing separation of client and server parts of the object system and how transformations and position streaming is communicated. . . . . 57
- 3.6 This Figure shows two tables measuring pure code execution time: without graphical output operations. In the left table, 10 measurements recorded the time taken to execute the three instructions for rotating an object about an observer situated at an arbitrary point. OpenGL C code was executed in a loop of 50 million iterations. The average was calculated and labeled A. In the right table are the measurements for executing 50 million rotations about an origin centered observer, using the single rotate instruction. The average was labeled B. As can be seen from the calculation below the table, the origin centered rotation is 27% faster. . . . . 67

3.7 This Figure shows two tables measuring pure code execution time: without graphical output operations. In the left table, 10 measurements recorded the time taken to execute the three instructions for rotating an object about an observer situated at an arbitrary point. The code used in this case was modelled after O’Neil’s algorithm. In the right table are the measurements for executing 50million rotations about an origin centered observer, using the single rotate instruction. The average was labeled B. As can be seen from the calculation below the table, the origin centered rotation is 31% faster. . . . . 71

3.8 This Figure shows two tables showing rotation about observer timings, including graphical output instruction execution. On the left are 10 time measurements recording the time taken to rotate an object, using OpenGL, about an observer situated at an arbitrary point. The average is labeled A. On the right are the measurements for rotating the object about an observer centered at the origin, the average labeled B. As can be seen from the calculation below the table, the origin centered rotation is 9% faster. . . . . 72

3.9 There are two main types of coordinate systems in the proposed origin centric approach: the application coordinate system of the object system (OS) and the world coordinate system used by the display system (DS). The application coordinate system may use one or more coordinate representations to encompass the entire application space. The world coordinate system is a cartesian (x,y,z) and time space. The mapping between the object system and display system must handle the conversion between application coordinates and world coordinates. This happens when objects in the object system are seeded into the active object framework of the display system, e.g. when the observer comes in range of something that can become visible, or might otherwise have some impact on the immersive environment. . . . . 80

- 3.10 The real number valued position  $(x,y,z)$  will sit somewhere within a cube formed by the eight closest representable floating point coordinates. The largest gap between points in the floating point field near the real number valued position  $(x,y,z)$  is the diagonal gap between a corner point of the cube and the diametrically opposite corner point (line AB). . . . . 85
- 3.11 A more correct illustration of how the representable floating point numbers are spaced with increasing gaps the further they are from the origin. The gaps between numbers only increase on every power of two. Therefore there are increasingly larger runs of numbers with the same spacing before the spacing doubles at the next power of two. . . . . 87
- 3.12 The top right area of this Figure illustrates how a real world coordinate  $(x,y,z)$  is quantised in a virtual world into one of the surrounding representable coordinates (red dots). The fidelity of virtual space is so high near the origin that the discrete red dots would appear as a continuous block if plotted. Spatial jitter is more evident in the "rough region" away from origin. . . . . 88
- 4.1 Image of the test model designed in VRML using a simple arrangement of viewpoint and coloured dots. The viewpoint was initially positioned at  $(1,1,1)$  and oriented to look at the origin. This gave a viewpoint that was not aligned to any axis thus ensuring there would be a spatial error contribution from each axis variable to the viewpoint position. An axis aligned box with checkerboard pattern was included in the model. This provided a visual reference showing the view direction was accurately positioned centrally and looking at the origin. The box also allowed for measurement of the x and y perspective foreshortening due to jitter in the z direction. . . . . 96

4.2	Scene displaced by (100000, 100000, 100000). Slight displacements of the coloured dots were measurable compared to the image of the model placed at the origin. . . . .	99
4.3	Scene displaced by (1million, 1million, 1million). Much larger displacements of the coloured dots were measurable compared to the image of the model placed at the origin. . . . .	99
4.4	This Figure shows a graph of jitter distance (in pixel units) versus distance from origin for each coloured dot in the test model. As expected, a general exponential upward trend with increasing distance is indicated by the graphed lines. Also, note the increasing random fluctuation 3million mark. . . . .	100
4.5	This Figure shows the results using an exponential curve of best fit. The exponential trend is not so clear for the yellow and white dot data sets due to a low jitter datum value in each. . . . .	101
4.6	In this Figure the exponential trend is more consistent after removing just two of the lowest values : the yellow dot datum at the 5,000,000 mark and the white dot datum at the 4,000,000 mark. . . . .	103
4.7	Raytraced scene rendered at the origin. This is the scene reference image to compare displaced scenes against. Visual comparisons can be made with the eye by comparing a displaced scene with this one. However, most differences will not be visible to the eye for small displacements and for these cases the difference images (see following figures) should be used. . . . .	107
4.8	The same scene as Figure 2 but rendered at a displacement of (150,150,150) from the origin. The entire scene, including camera and lights is moved with the objects. Differences, however, are not visible to the naked eye but the RMS difference is 0.127839 and the rendering differences are revealed when an image difference operation is performed, as shown in Figure 4.9. . . . .	107

- 4.9 Image showing the difference comparison of the scene displaced by (150,150,150) in Figure 4.8 and the reference image shown in Figure 4.7. 108
- 4.10 Image showing the ‘noise’ differences revealed when subtracting one image of the scene rendered at (0,0,0) and another image of the scene rendered at the same location but at a different time. Brightness and contrast were maximised to reveal the differences. This noise image was used as a control to compare other difference images against. Only those difference images that showed greater differences than this one were taken to have significant differences (above the noise level) due to simulation error. For this case, the RMS difference was 0.157205. . . 113
- 4.11 Figure a on the left is the same noise image shown in Figure 4.10 indicating the upper threshold of error possible in scene rendering at the origin. Figure b is a difference comparison of the scene in Figure 4.8 (the scene displaced by (150,150,150)) with the reference image shown in Figure 4.7. Brightness and contrast are maximised as for Figure 4.10. Notice that, compared to the noise image in a, the image in b has an apparently larger number of dots and stronger outlining of the shapes of objects in the scene. . . . . 114
- 4.12 The scene displaced by 100,000m. It is still not possible to see the differences with the naked eye but the RMS value of 0.173134 proves there are indeed some differences. . . . . 115
- 4.13 Difference image for the 100,000m displaced scene produced from the image in 4.12 and the reference image in 4.7. Many more differences are revealed. . . . . 115
- 4.14 Scene displaced by 1,000,000m. Note the strange truncation of the foreground eggs. This is the first example where differences are clear to the naked eye. . . . . 115
- 4.15 Difference image for the 1,000,000m scene produced from the image in 4.14 and the reference image in 4.7. . . . . 115

4.16	Scene displaced by 2,000,000m. Note: despite the dramatic change, this is not a difference image. . . . .	116
4.17	Difference image for the 2,000,000m displaced scene produced from the image in 4.16 and the reference image in 4.7. This Figure appears suspect: one would expect it to show virtually the entire scene's shapes appearing as differences. However, in this case the lack of detail is a side effect of maximising the brightness and contrast. Figure 4.18 shows the difference image without brightness and contrast adjustment and the expected detail is clearly shown. Maximising the brightness and contrast is mostly useful for making very small values visible. . . . .	116
4.18	Difference for the image for the 2,000,000m displaced scene produced from the image in 4.16 and the reference image in 4.7 <i>without brightness and contrast adjustment</i> . As this Figure shows, almost the entire scene is different. . . . .	117
4.19	Model of BMW positioned at (0,0,0) and rendered using single precision coordinates, BS Contact 6.1 VRML browser and RadeonX600 hardware. The image shown was taken as a snapshot from the screen before storing as the reference image. . . . .	120
4.20	Difference image between the BMW scene rendered at (520,520,520) and the reference BMW image. . . . .	120

4.21	Difference image between the BMW rendered at (100km, 100km, 100km) and the reference BMW image. Note that the image of the BMW rendered at (100km, 100km, 100km) appears to the naked eye exactly the same as the one rendered at the origin so there was no reason to reproduce it here. . . . .	121
4.22	Difference image between the BMW rendered at (1million, 1million, 1million) and the reference image. . . . .	121
4.23	Model of a Porsche positioned at (0,0,0) and rendered using single precision, C and OpenGL. Hardware was a VAIO notebook with RadeonX600 graphics chip. The image shown was taken as a snapshot from the screen before storing as the reference image. . . . .	123
4.24	Difference image between the Porsche scene rendered at (100, 100, 100) and the reference Porsche image. . . . .	124
4.25	Difference image between the Porsche rendered at (100km, 100km, 100km) and the reference Porsche image. . . . .	125
4.26	Difference image between the Porsche rendered at (5million, 5million, 5million) and the reference image. . . . .	126
4.27	Snapshots of the test model at the origin (left) and 3000m (right). Unexpectedly, different rendering artifacts than the anticipated misordering of red and white tiles were visible to the naked eye at the relatively small distance of 3000m. Notice the white edged horizontal highlights between the top two rows and vertical pale red highlights further down. In hindsight, such effects should have been anticipated because of minute x-y relative shifts of the tiles. . . . .	131
4.28	Snapshots of the test model at the origin (left) and 70,000m (right). Clearer separation of tiles has become apparent in the right hand image and bright red from tiles at the back shows through unobscured. . . .	132

- 4.29 Snapshots of the test model positioned at (300000, 300000, 300000) (left) and (700000, 700000, 700000) on the right. The left hand model shows the first occurrence of the expected incorrect z buffer ordering of tiles, and both the upper two rows are consistent. Some random shifting of tiles is now starting to become clear as well. In the right hand image, although pairs numbered 1 and 10 have the same separation of 0.1, they show different z ordering errors. This is consistent with the predicted random behaviour of z ordering logic when the jitter can displace objects in one direction or the other. Similarly pairs 2 and 11 and 5 and 14 show the same effect. In theory, the columns headed by tiles 3 then 4 should have showed red tiles before the column headed by tile 5 because they have smaller separations. However, again, consistent with the random nature of jitter, the results are somewhat random too. The image for the model displaced by 1million showed the same results. This is not too surprising because there is no power of two jump in the mantissa value between 700,000 and 1million. . . . . 133
- 4.30 Snapshots of the test model positioned at (2million, 2million, 2million) (a) and (3million, 3million, 3million) (b) . . . . . 134
- 4.31 Snapshot of the test model positioned at (5million, 5million, 5million). It is hard to believe these are all the same model as in 4.27a. Red tiles are showing in front in all the rows and the model is breaking up badly. 134
- 4.32 This Figure shows a graph of z displacement versus distance from origin for the red-white tiled grid. The highest numbered tile that was obscured by its red counterpart was recorded. Its separation distance was recorded and converted into pixel coordinates. The graph shows pixel coordinate displacement against distance from origin. . . . . 135

- 4.33 Snapshots of a rotating very large planet at a great distance from the viewer. As the planet rotates, polygons on its surface are randomly clipped behind the far clipping plane. Consequently, they alternately appear black (clipped) then white (not clipped), producing a flashing effect. . . . . 137
- 4.34 Example of depth ordering problem with large, very distant object. This image is a view of planet and a sun in the Eve-online game. The depth order of the very distant Sun was calculated as being in front of the planet (slightly bottom left of center of planet). The Sun shows through the solid planet instead of being hidden by it. . . . . 138
- 4.35 Z buffer tearing on space station in eve-online game. The tearing appears and disappears randomly, producing an unsightly flashing effect during animation. Note that the patchy texture makes the tearing a little difficult to see on the static image but can be distinguished by its apparent alignment which is at an angle to the viertical/horizontal alignment of the station's texture. . . . . 140
- 4.36 Z buffer tearing on a planet in the eve-online game. There is too little Z depth resolution to distinguish which of overlapping surfaces used to represent atmosphere and rings are in front or behind. . . . . 141
- 4.37 Snapshots from a python ODE simulation where the left hand block is stationary on a plane and the right hand block is dropped beside it. The right hand image shows effect of shifting the simulation by 10m. As can be seen, the right hand block's rest position in the two images is quite different. . . . . 143

- 4.38 Snapshots from a python ODE simulation where the left hand block is stationary on a plane and the right hand block is dropped beside it. The right hand image shows effect of starting the simulation at a different time = 8000 compared to the left hand simulation which starts dropping the right hand block at time = 20. In both cases the initial block positions were the same. As can be seen, the right hand block's rest position in the second image is quite different from the first. 144
- 4.39 This Figure is replicated from Chapter 1 for ease of reference. The Figure shows snapshots of two moving dots overlaid into one image. The dots are near (1million, 1million, 1million) and move from the distant upper right to the near lower left. Each upper dot snapshot is connected by a line, tracing its jittery path. A straight line underneath highlights the uneven motion of the dots. . . . . 147
- 4.40 Snapshots from the Hall of Jitter, with the initial view on the left and the view after moving inside the Hall entrance. Both of these views are taken from the Hall positioned at the origin. . . . . 150
- 4.41 Image of hall with HUD turned on. . . . . 151
- 4.42 Interaction test results from the Hall of Jitter test model. As can be seen from the Table, the mouse always functioned as expected but the keyboard response started to fail from 1,000,000 onwards. . . . . 151

- 4.43 This Figure shows two tables measuring pure code execution time: without graphical output operations. In the left table, 10 measurements recorded the time taken to execute the three instructions for rotating an object about an observer situated at an arbitrary point. OpenGL C code was executed in a loop of 50million iterations. The average was calculated and labeled A. In the right table are the measurements for executing 50million rotations about an origin centered observer, using the single rotate instruction. The average was labeled B. As can be seen from the calculation below the table, the origin centered rotation is 27% faster. . . . . 153
- 4.44 This Figure shows two tables measuring pure code execution time: without graphical output operations. In the left table, 10 measurements recorded the time taken to execute the three instructions for rotating an object about an observer situated at an arbitrary point. The code used in this case was modelled after O’Neil’s algorithm. In the right table are the measurements for executing 50million rotations about an origin centered observer, using the single rotate instruction. The average was labeled B. As can be seen from the calculation below the table, the origin centered rotation is 31% faster. . . . . 154
- 4.45 This Figure shows two tables, each showing rotation about observer timings, including graphical output instruction execution. On the left are 10 time measurements recording the time taken to rotate an object, using OpenGL, about an observer situated at an arbitrary point. The average is labeled A. On the right are the measurements for rotating the object about an observer centered at the origin, the average labeled B. As can be seen from the calculation below the table, the origin centered rotation is 9% faster. . . . . 155

- 4.46 Table showing, on the left, 10 time measurements recording the time taken to execute the `insideBox` function for calculating proximity from an observer at an arbitrary position. Measurements were taken using a C loop of 500million iterations. The average is given below and labeled A. On the right are the measurements for executing 500million `insideBox` calculations where the observer is at the origin. The average was calculated and labeled B. As can be seen from the calculation below the table, the origin centered distance calculation is 14% faster. 156
- 4.47 Table showing, on the left, 10 time measurements recording the time taken to execute the normal square root function for calculating distance from an observer at an arbitrary position. Measurements were taken using a C loop of 500million iterations. The average is given below and labeled A. On the right are the measurements for executing 500million distance calculations where the observer is at the origin. The average was calculated and labeled B. As can be seen from the calculation below the table, the origin centered distance calculation is only 0.95% faster. . . . . 157
- 4.48 Series of zooms from space down to ground level in Sydney. Navigation mode was changed to “Walk” and motion was smooth while walking up to two figures in front of the Museum of Contemporary Art. The yellow translucent object in front of the figures also animated smoothly. 161
- 4.49 From top to bottom: 3D Models on left, 2D images on right with successively larger coordinates. The bottom model shows bad floating point quantization, jitter when manipulated and pathological diversion/conversion of parts of the mesh. . . . . 164

- 4.50 A region of space showing two points and the surrounding 8 coordinates that each can have (the nearest red spheres). When each point moves in opposite directions, this is a pathological case of diversion. The largest diversion can occur when they diverge along the diagonals of the cubes. Alternatively, a pathological conversion can occur if each point is closer to the sphere common to both cubes. . . . . 166
- 4.51 Comparison of a simple, maximum mantissa value error calculation (e.g. used for GeoVRML) compared to the relative spatial error (RSE) calculation. The first table compares single precision floating point error for four coordinate values ranging from 100,000 to 10,000,000m. The ratio of spatial error to maximum mantissa value error is given in the right column. The second table shows the same comparisons but for double precision. Notice that the relative spatial error is always larger. . . . . 167
- 5.1 Elements of an error minimising geospatial simulation pipeline: cartographic input, geospatial models, calculations and projections. Origin centric techniques such as floating origin, lazy evaluation and progressively refined fidelity are incorporated to improve accuracy. The application of origin centric techniques to the geospatial domain is described in the Cyberworlds 06 paper [86]. . . . . 173
- 5.2 In this Figure the error minimising pipeline concept is broadened to the more holistic concept of an optimising simulation pipeline. An optimising simulation pipeline not only offers the benefits of an error minimising simulation pipeline but also improved scalability, repeatability of results such as physics simulation, consistency, graphics performance, floating point compression and code simplification. . . . 174

- 5.3 Simplified model of the transformation of coordinates traveling down the narrowing precision bottleneck of a simulation pipeline. Operations such as world transformation due to viewpoint selection or navigation are shown as well as the floating origin subtraction performed on object positions after projection to cartesian values. Object positions are relative to the object system application coordinate of the viewpoint, and thus become small valued coordinates relative to the world space origin in the display system. The viewer does not travel through the world - the world comes to the viewer. . . . . 179
- 5.4 This Figure shows how the techniques discussed in this thesis fit into the overall architecture of a distributed simulation application. A number of high level techniques such as PRF and LOD apply at the high precision server side of the networked application. This part of the application includes the store for objects with their high precision positions and the global simulation algorithms. The majority of techniques apply on the client side where there are spatial projections, local simulation algorithms, the core floating origin technique, centered navigation, centered viewpoints and other techniques. . . . . 182
- 5.5 This Figure shows the positive portion of the floating point number line. The step-wise increasing gaps between numbers represents how gap error doubles at each *exponent boundary*: where the exponent is incremented. Observe that the greatest density of exponent boundaries exists close to the origin. This implies that operations on two or more points near the origin will likely span multiple exponent boundaries. . . . . 192

- 5.6 A region of space showing two boxes delineating machine representable positions for coordinates. The corners of both boxes are the only representable coordinate values for the real valued coordinates  $(x,y,z)$  and  $(x',y',z')$ . The box surrounding  $(x',y',z')$  is twice as large as that for  $(x,y,z)$ , although it does not look like it because it is further away. For pathological diversion, arrows show the distance  $e$  that  $(x,y,z)$  and  $2e$  that  $(x',y',z')$  can diverge respectively.  $(x',y',z')$  will alias twice as far:  $2e$  as  $(x,y,z)$ :  $e$ . The line passing through point A represents the plane that marks the transition from coordinates with one exponent value to higher valued coordinates with exponents twice as large: an exponent boundary. . . . . 196