Developing an Algorithm for Visualizing Large Terrain Environments in Real-Time

A thesis submitted to the Department of Computer Science, Faculty of Computer and Information Sciences, Ain Shams University.
In partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences.

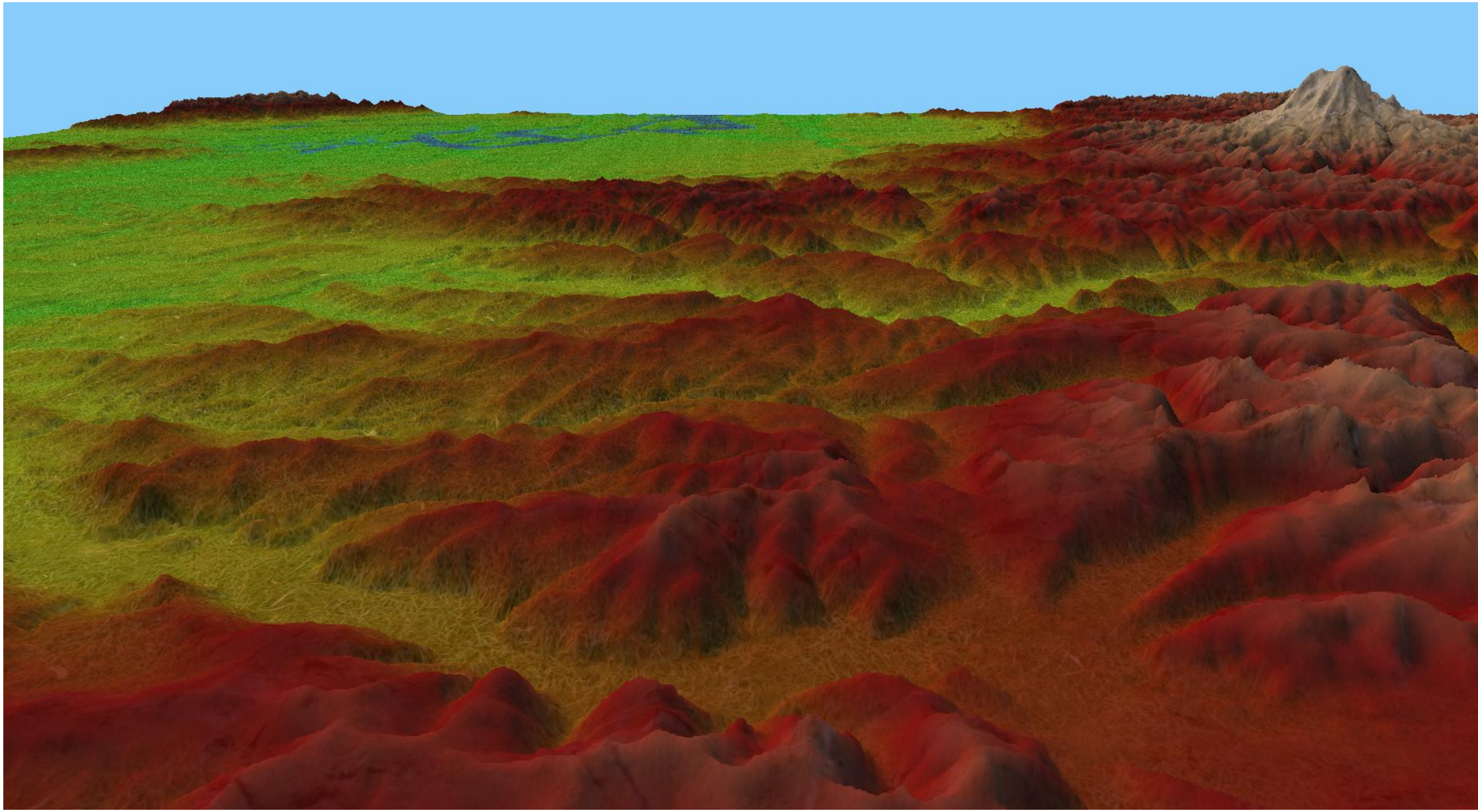
By:
Mohammad Ahmad Yusuf Al-Afandy
B.Sc. in Computer and Information Sciences (2006)

Under the supervision of:

Prof. Dr. Mostafa Gadal-Haqq M. Mostafa
Professor of Computer Science,
Faculty of Computer and Information Sciences, Ain Shams University

Prof. Dr. Taha Ibrahim Bassyouny Elarif
Professor of Computer Science,
Faculty of Computer and Information Sciences, Ain Shams University

Cairo – 2016



The Puget Sound area with 4097 × 4097 16-bit samples and inter-pixel spacing of 40 meters rendered by the designed algorithm

Acknowledgements

In the name of Allah, the Most Gracious and the Most Merciful

All praises to Allah for giving me the strength and blessing to complete writing this thesis.
My sincere appreciation and gratitude to Prof. Mostafa Gadal-Haqq M. Mostafa for all his efforts, and the time that he has devoted to support and assist me throughout this thesis creation.
I would also like to thank Prof. Taha Ibrahim Bassyouny Elarif for his help and guidance.
I thank my family for their support especially my wife Doaa who contributed directly and indirectly to this research and helped with most of the logistics.

Abstract

Terrain rendering has many important applications in the fields of modeling, geographic information systems, videos games, space modeling, flight simulation, synthetic vision systems (SVSs) and others. Due to this, it has been an active area of research for decades. Terrain rendering methods have been devised to handle the rendering of massive terrain datasets in real-time. These methods are designed to deal with large datasets at interactive frame rates to render terrains covering large areas in real-time.

In this research, several approaches to this problem are investigated and based on this investigation, a quad-tree level-of-detail (LOD) multi-activity based method for real-time terrain rendering is presented. The method uses two concurrent activities running in parallel. The errors activity decides what to be rendered and the rendering activity do the actual rendering. The two activities communicate with each other via a LOD hierarchy that is constructed by the errors activity and stored in main storage and each one of them is assigned a CPU thread and a GPU context. The rendering activity then renders the whole terrain as small blocks with different sizes using reusable vertex and index buffers with different scaling and translation parameters based on blurred versions of the height-field texture that are calculated on-the-fly on a vertex shader. Discontinuities are handled using incremental constant vertex and index buffers that cover all possible cases of LOD differences ensuring a tight mesh of terrain geometry where each LOD difference adds a number of

indices from the same index buffer used to handle all cracks.

A hybrid fuzzy texturing method that combines real and artificial details is also presented. It is based on an automatic fuzzy system for blending different detail textures with the terrain base texture to give varying artificial details based on terrain geometry. The original base color that comes from the real terrain texture is still preserved though by using a weighting scheme that favors the base color.

The results of the conducted case study show that although the proposed Hierarchical Error Map method achieves expected interactive frame rates at guaranteed very small screen-space errors of 1, 2, and 3 pixels, the CPU usage is been kept minimum due to the proposed notion of errors texture that is calculated entirely on the GPU.

Contents

List of Figures **5**

List of Tables **11**

1 Introduction **17**

1.1 Overview 18

1.2 Important Applications 20

1.3 Challenges 24

1.4 Problem Statement 29

1.5 Contribution 30

1.6 Thesis Organization 31

2 Background and Related Work **33**

2.1 Terrain Representation 33

2.2 Ray-tracing Methods 36

2.2.1 Ray-tracing Methods Survey 39

2.3 Triangulation Methods 41

2.3.1 Wavelet Methods 44

2.3.2 Geometry Clipmaps 51

2.3.3 Diamond Methods 58

2.3.4 Quad-tree Methods 61

3 The Hierarchical Error Map Method **89**

3.1 Overview 89

3.2 Tree Building Activity 94

3.3 Rendering Activity 100

3.3.1 Terrain Geometry 101

3.4 Discontinuities Management 104

3.5 Activities Synchronization 110

3.6 Scalability 111

3.7 Augmented Fuzzy Terrain Texturing 112

3.7.1 Basic Texturing 112

3.7.2 Proposed Texturing Method 113

4 Results and Discussion 119

4.1 CPU Usage Analysis 119

4.2 Frame-rate Analysis 121

4.3 GPU Memory Usage 134

4.4 The proposed Augmented Fuzzy Terrain Texturing Visual Quality 135

5 Conclusion and Future Work 141

References 145

Abstract (Arabic) 155

List of Figures

1.1 Classical Wooden models 17

1.2 Modern Virtual 3D Model 18

1.3 3D City Model 19

1.4 Manipulating a 3D Model 19

1.5 Traditional Airplane Cockpit 20

1.6 Enhanced Vision Systems 21

1.7 Flying Blind 22

1.8 Flying with SVS 22

1.9 A close-view of a SVS Cockpit 23

1.10 Distant cockpit view of a SVS 23

1.11 Tunnel Guidance 24

1.12 Satellite Map 24

1.13 Topological Map 25

1.14 Topographical Map 25

1.15 GIS Maps 26

1.16 Two different 3D terrain models 27

1.17 The most basic terrain rendering algorithm that takes as input (a) a height-field and (b) a base texture and renders c) the output 3D terrain model 28

2.1 A TIN 34

2.2 A Height-field 35

2.3 (a) Rays cast from the viewpoint (b) 2D projection of a ray 36

2.4 Ray Intersection Test 36

2.5 Hierarchical Ray-tracing 37

2.6 Trace Miss 38

2.7 Aliasing Artifacts 38

2.8 (a) Input Height-field (b) Converted Polygons 41

2.9 (a) Input Color Grid (b) Output 42

2.10 A terrain block in various LODs 43

2.11 Discontinuities as cracks in textured and wire-frame modes 43

2.12 Coefficients of two levels of wavelet transform 44

2.13 Wavelet reconstructed approximations 45

2.14 Wavelet reconstructed details 46

2.15 Creating the wavelet LOD hierarchy from [38] 47

2.16 Terrain Decomposition from [52] 47

2.17 Different Approximations based on different error thresholds from [52] 48

2.18 The output of the method used in [52] integrated into a SVS 49

2.19 The coefficients tree and the resulting coefficients texture used in [14] 49

2.20 Adjacency relations used in [17] 50

2.21 Tesselated output of [35] 51

2.22 Transitional regions to eleminate popping visual artifacst [35] 52

2.23 Transition Morphs in [35] 53

2.24 Frustum culling in [35] 53

2.25 Terrain Synthesis in [35] 54

2.26 Clipmaps Pyramid from [2] 55

2.27 Clipmap Ring used in [2] 55

2.28 Spherical Clipmaps in [7] 56

2.29 Different diamond configurations 58

2.30 Diamonds Various LODs from [25] 58

2.31 Diamonds Various LODs from [19] 59

2.32 Mesh construction from a DAG of Diamonds [53] 61

2.33 Terrain generated using an adaptive Quad-tree 62

2.34 A restricted and an unrestricted Quad-tree 63

2.35 Crack-free triangulation 63

2.36 Geometry mipmaps and cracks 64

2.37 Adaptive LODs from [44] 67

2.38 The rendering algorithm used in [6] 71

2.39 Cracks handling between two blocks with different LODs from [6] 72

2.40 A terrain patch that is divided into 5 regions from [6] . . 73

2.41 Nested Mesh Hierarchy used in [46] 75

2.42 Pyramidal representation used in [56] 80

2.43 A progressive patch (a) at a low level and (b) its active vertices to split and reach (c) the next level and (d) its triangulation from [56] 81

2.44 Sharp edges handling with texture blending from [56] . . 82

2.45 Adaptive subdivision from [27] 85

2.46 Handling cracks in [27] 86

3.1 A view of a 9×9 vertices terrain: (a) as a mesh and (b) as a quad-tree with $K_b = 3$ terrain blocks represented as leaves. 90

3.2 The Hierarchical Error Map Method System Architecture. 91

3.3 (a) A single mesh with $K_b = 3$ in used for rendering all terrain blocks of (b) a terrain with 9×9 vertices. The small shaded area is of LOD 0 and is rendered with $S = 1$ and $T = (6, 0)$ while the large shaded area is of LOD 1 and is rendered with $S = 2$ and $T = (4, 4)$ 93

3.4 Down-sampling process for: (a) a height-field with n=2, (b) its triangulation, (c) errors texture. Greyscale levels show which vertices exist in the different LODs 0, 1, 2 respectively from top to bottom. 96

3.5 (a) The errors texture of a height-map with $N = 5$ and (b) the built quad-tree from it. 100

3.6 Traversal process for (a) the errors quad-tree and (b) its associated terrain blocks tree to reach at (c) the final terrain blocks to send to the GPU 102

3.7 Cracking visual artifact (wireframe) 105

3.8 The visual distortion due to cracks 106

3.9 Two adjacent k=9 terrain blocks with LOD difference of 1 107

3.10 Two adjacent k=9 terrain blocks with LOD difference of 2 108

3.11 Two adjacent k=9 terrain blocks with LOD difference of 3 109

3.12 Detailed structure of the rendering fragment shader implementing the proposed augmented fuzzy terrain texturing method 114

3.13 (a) a fuzzy function and (b) a plot of three detail textures fuzzy membership functions 115

4.1 A plot of the frame rate against the experiment time with $K_b = 513$ and $\delta = 1$, $\delta = 2$, $\delta = 3$ for the Puget Sound dataset without frustum culling 123

4.2 A plot of the triangles count per frame against the experiment time with $K_b = 513$ and $\delta = 1$, $\delta = 2$, $\delta = 3$ for the Puget Sound dataset without frustum culling 124

4.3 A plot of the frame rate against the experiment time with $K_b = 513$ and $\delta = 1$, $\delta = 2$, $\delta = 3$ for the Puget Sound dataset with frustum culling 125

4.4 A plot of the triangles count per frame against the experiment time with $K_b = 513$ and $\delta = 1$, $\delta = 2$, $\delta = 3$ for the Puget Sound dataset with frustum culling 126

4.5 A plot of the frame rate against the experiment time with $K_b = 129$ and $\delta = 1$, $\delta = 2$, $\delta = 3$ for the Puget Sound dataset without frustum culling 127

4.6 A plot of the triangles count per frame against the experiment time with $K_b = 129$ and $\delta = 1$, $\delta = 2$, $\delta = 3$ for the Puget Sound dataset without frustum culling 128