Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

3D Terrain with Level of Detail

Written report for the module
BTI3041 – Project 2
by

Amar Tabakovic

Bern University of Applied Sciences
Engineering and Information Technology
Computer Perception & Virtual Reality Lab

Supervisor
Prof. Marcus Hudritsch

October 11, 2023

Abstract

TBD.

Contents

1 Introduction **2**
- 1.1 Goals of this Project . . . 2
- 1.2 Structure of the Report . . . 2

2 Existing Work and Literature **4**
- 2.1 Research Articles and Publications . . . 4
- 2.2 Level of Detail for Computer Graphics . . . 4
- 2.3 Focus on 3D Terrain Programing . . . 4
- 2.4 Virtual Terrain Project . . . 5

3 Terrain LOD in Real-world Systems **6**
- 3.1 Game Engines . . . 6
  - 3.1.1 Godot . . . 6
  - 3.1.2 Unity . . . 6
  - 3.1.3 Unreal Engine . . . 7
- 3.2 Geographic Information Systems . . . 7

4 Algorithms and Approaches for Terrain LOD **8**
- 4.1 Basics of Terrain LOD . . . 8
  - 4.1.1 Terrain Data Representation . . . 8
  - 4.1.2 Bintrees and Quadtrees . . . 9
  - 4.1.3 Potential Problems . . . 9
- 4.2 ROAM . . . 9
- 4.3 CDLOD . . . 9

5 ATLOD: A Terrain Level of Detail (Renderer) **10**

6 Results **11**

7 Discussion and Conclusion **12**
- 7.1 Further Work . . . 12
- 7.2 Reflexion . . . 12

Bibliography **14**

Chapter 1

Introduction

In the field of 3D computer graphics, rendering is one of the central tasks. Many practical applications of 3D computer graphics make use of terrains, such as flight simulators, open-world video games, and Geographic Information Systems (GIS) [1, p. 185]. At the same time, rendering large and constantly visible objects, such as the terrain, is computationally expensive and optimizations are necessary in order to avoid performance deficiencies.

One area which offers potential for optimizations is the *level of detail (LOD)* of objects. The concept of LOD is based on the idea that the farther away an object is, the fewer details are going to be visible to the human eye.

The problem of rendering terrains spawned numerous algorithms and approaches specifically for this purpose.

1.1 Goals of this Project

The main goal of this project is to gain an overview over the field of terrain LOD. This includes analyzing and comparing a selection of terrain LOD algorithms and approaches, including implementing as part of a demo computer application.

1.2 Structure of the Report

This report is structured as follows:

- Chapter 2 gives an overview of work that has already been conducted in the area of terrain LOD and relevant literature for this project.
- Chapter 3 Lists a few real-world examples where terrain LOD is being used, such as game engines or geographic information systems.
- Chapter 4 introduces the reader to various approaches for terrain LOD, beginning with some basic background information on terrain mod-

elling and LOD approaches in general. Afterwards, a selection of algorithms are presented in detail, including ROAM, CDLOD and more.

- Chapter 5 describes ATLOD, the demo application for terrain rendering implemented as part of this project.
- Chapter 6 TODO
- Chapter 7 TODO

Chapter 2

Existing Work and Literature

2.1 Research Articles and Publications

Terrain LOD is a well-researched topic and over the last three decades, numerous approaches have been published. TODO: list research

2.2 Level of Detail for Computer Graphics

Level of Detail for Computer Graphics by Luebke *et al.* [1] is a reference book for the topic of LOD published in 2002. The book builds on top of years of research in the area of LOD and provides an overview to many LOD techniques. For this project, chapter 7 "Terrain Level of Detail" of the book is especially relevant, as it dives into the topic of LOD for terrains specifically. It covers basic approaches and techniques for terrain LOD, common problems that can arise during rendering of terrains and some solutions to them, and a catalog of terrain LOD algorithms.

2.3 Focus on 3D Terrain Programing

Focus on 3D Terrain Programming by Trent Polack [2] is a book on terrain programming published in 2002. Part one of the book introduces the reader to the basics of terrains, such as height maps and texturing. Part two then covers some more advanced topics, including a selection of terrain LOD algorithms. The presented LOD algorithms are

- ROAM by Duchaineau *et al.* [3],
- the quadtree-based algorithm described in "Real-Time Generation of Continuous Levels of Detail for Height Fields" by Röttger *et al.* [4],
- and geomipmapping by de Boer [5].

The book also includes demo source code in C++ and OpenGL.

2.4 Virtual Terrain Project

The *Virtual Terrain Project* [6] was a project run from 2001 to 2013 that consisted of a collection of software, information and resources on terrain modelling and rendering, including a large overview of publications and implementations related to terrain LOD algorithms.

Chapter 3

Terrain LOD in Real-world Systems

3.1 Game Engines

3.1.1 Godot

Godot is a cross-platform game engine written in C#, C++ and its own scripting language GDScript. Terrains are supported in form of extensions developed by community members, which can be installed and used in Godot projects by game developers.

One such extension is Terrain3D by Cory Petkovsek [7] written in C++ for Godot 4. The LOD approach used in this extension is based on geometry clipmaps by Hoppe and Losasso [8]. The concrete implementation of the geometry clipmap mesh code was created by Mike J Savage [9].

Another extension for terrains is Godot Heightmap Plugin by Marc Gilleron [10] written in GDScript and C++. The extension uses a quadtree-based approach for terrain LOD.

3.1.2 Unity

Unity is another cross-platform game engine written in C# and C++, and has a built-in terrain system. The core engine source code of Unity is only accessible by owning an enterprise licence, therefore no information is given on which terrain LOD is used for the built-in terrain in Unity.

Nonetheless, there exists an open-source library for hierarchical LOD in Unity called HLODSystem developed by JangKyu Seo at Unity TODO citation . HLODSystem also supports terrains with its TerrainHLOD component, allowing for conversion from an Unity Terrain object to a HLOD mesh with configurable parameters, such as chunk size and border vertex count. HLODSystem allows the developer to specify the mesh simplifier to be used and currently the only supported simplifier is UnityMeshSimplifier,

an open-source mesh simplifier developed by TODO that utilizes the fast quadratic mesh simplification algorithm developed by TODO citation.

3.1.3 Unreal Engine

Unreal Engine is another cross-platform game engine written in C++ and features an integrated terrain system called the Landscape system. The technical documentation of Unreal Engine 5 mentions utilising geomipmapping for handling LOD for landscapes [11]. Geomipmapping is a terrain LOD approach developed in 2000 by de Boer [5].

3.2 Geographic Information Systems

Chapter 4

Algorithms and Approaches for Terrain LOD

4.1 Basics of Terrain LOD

4.1.1 Terrain Data Representation

One way of representing terrains is using *heightmaps*. A heightmap is a $n \times n$-grid that contains the height value y for each (x, z)-position. Positions are always spaced evenly in a grid-like manner, but the distance between two neighboring vertices is variable and can be set by the terrain programmer.

The main advantages of heightmaps are easy storage and manipulation of height values. Terrain height data can be easily stored as grayscale images, where low grayscale values represent low areas of terrain and vice versa for high grayscale values. Figure 1 shows a 2000×2000 heightmap of the mountain Dom in Valais, Switzerland.

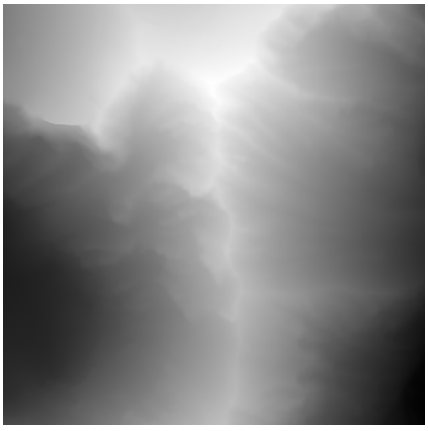

Figure 4.1: 2000×2000 heightmap of the mountain Dom in Valais, Switzerland retrieved from SwissTopo [12].

An alternative to the heightmap is the *triangulated irregular network (TIN)* data structure. A TIN consists of a collection of 3D vertices where the arrangement of vertices can be irregular. The main advantage of TINs is that fewer polygons need to be used for e.g. smooth terrain areas. However, view culling, deformations and terrain following are more difficult to implement. Also, the full (x, y, z) coordinates need to be stored, whereas with heightmaps, only the height value y needs to be stored.

4.1.2 Bintrees and Quadtrees

Binary triangle trees (bintrees) and quadtrees are recursive data structures based on triangles and quads respectively.

4.1.3 Potential Problems

4.2 ROAM

ROAM (short for **R**eal-time **O**ptimally **A**dapting **M**eshes) is an algorithm by Duchaineau *et al.* [3] published in 1997.

ROAM represents the terrain mesh using bintrees.

The algorithm is driven by two priority queues: a split queue and a merge queue. The split queue contains triangle split operations

The main advantage of this approach is that the terrain mesh from a previous frame can be used for the current frame. The mesh doesn't have to be built from ground up each frame.

4.3 CDLOD

Chapter 5

ATLOD: A Terrain Level of Detail (Renderer)

This chapter describes *ATLOD* (short for **A** **T**errain **L**evel **o**f **D**etail (Renderer)), the demo terrain rendering application.

Chapter 6

Results

Chapter 7

Discussion and Conclusion

7.1 Further Work

7.2 Reflexion

Bibliography

[1] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., USA, 2002.

[2] Trent Polack and Willem H. de Boer. *Focus on 3D Terrain Programming*. Premier Press, 2002.

[3] Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. In *Proceedings of the 8th Conference on Visualization '97*, VIS '97, pages 81–88, Washington, DC, USA, 1997. IEEE Computer Society Press.

[4] Stefan Röttger, Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. Real-time generation of continuous levels of detail for height fields. *Journal of WSCG*, 6(1-3), 1998. URL: `http://hdl.handle.net/11025/15845`.

[5] Willem H. de Boer. Fast terrain rendering using geometrical mipmapping. In *The Web Conference*, 2000. URL: `https://api.semanticscholar.org/CorpusID:7296927`.

[6] Virtual terrain project. `https://vterrain.org`.

[7] Cory Petkovsek. Terrain3d. `https://github.com/TokisanGames/Terrain3D`, 2023.

[8] Frank Losasso and Hugues Hoppe. Geometry clipmaps: Terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3), 2004. `doi:10.1145/1015706.1015799`.

[9] Mike J Savage. Geometry clipmaps: simple terrain rendering with level of detail. `https://mikejsavage.co.uk/blog/geometry-clipmaps.html`, 2017.

[10] Marc Gilleron. Godot heightmap plugin. `https://github.com/Zylann/godot_heightmap_plugin`, 2023.

[11] Epic Games. Landscape Overview — Unreal Engine 5.3 Documentation. https://docs.unrealengine.com/5.3/en-US/landscape-overview/.

[12] Federal Office of Topography swisstopo. swissALTI3D. https://www.swisstopo.admin.ch/en/geodata/height/alti3d.html.