Computers, Environment and Urban Systems 52 (2015) 21–33

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/compenvurbsys

An efficient terrain Level of Detail implementation for mobile devices and performance study

José P. Suárez [a], Agustín Trujillo [b], José M. Santana [b,*], Manuel de la Calle [c], Diego Gómez-Deck [c]

[a] Division of Mathematics, Graphics and Computation (MAGiC), IUMA, Information and Communication Systems, University of Las Palmas de Gran Canaria, Canary Islands, Spain
[b] Imaging Technology Center (CTIM), University of Las Palmas de Gran Canaria, Canary Islands, Spain
[c] IGO SOFTWARE, Department of R+D, Cáceres, Spain

ARTICLE INFO

Article history:
Received 9 June 2014
Received in revised form 20 February 2015
Accepted 21 February 2015

Keywords:
Virtual globe
Level of Detail
Earth navigation
Mobile devices
Height maps

ABSTRACT

On the basis of traditional Terrain Quadtree algorithms this paper introduces a new Level Of Detail (LoD) criteria which allows the visualization of a virtual earth on many kinds of mobile devices with a suitable accuracy. The earth rendering system is intended to run beneath the threshold that resources of current devices impose, regarding especially the graphics hardware capabilities and the memory usage. At the same time, the system deals flawless with typical screen-based user interaction allowing smooth flying and rapid orientation changes of the camera. The present work analyses the memory and graphics requirements from a theoretical perspective. Finally, we give a useful performance study that compares the globe on some mobile and desktop devices, focusing on LoD techniques, visibility test, creation of texture tiles, uploading tiles to GPU, and rendering.

1. Introduction

It is a known fact that terrain rendering has been an open problem for quite a long time now. The enormous amount of data to be fetched, the broad variety of data sources and types and the complexity of the rendering are only a few parts of this challenge.

At the same time the arise of the handheld computers such as smartphones and tablets has disrupted the GIS ecosystem offering new collective data sources (Kalantari, Rajabifard, Olfat, & Williamson, 2014) and new developing platforms. The difficulties only grow when dealing with mobile devices which short computational resources and multitouch interaction are still problematic. Many systems have been developed with approaches to the solution, most of them using techniques such as chunked terrain.

1.1. Modern approaches to terrain rendering

The quadtree structure was first proposed by Lindstrom et al. (1996) as a way to render height maps. Since then, many systems have adopted this approach to create Virtual Earths and map applications. Some of them belong to the closed-source category such as Marble, Apple's Maps, and Nokia Here. Systems like the popular Google Earth, based on Keyhole's EarthViewer (2004), rely on their own data sources allowing the user only to input data via KML files. Besides, closed software like this does not allow user-created plugins.

Modern projects are so diverse that a virtual globe engine should be as customizable as possible. The open-source community holds some good examples of this, with projects as World Wind (Bell et al., 2007), CesiumJS,[1] OpenWebGlobe[2] or WebGLEarth.[3]

Nasa's World Wind is based on Java so it can be used as a desktop application or embedded in a web environment. In addition, latest versions allow it to be run on Android. This project is driven now by the community and is completely customizable offering the chance of using imagery and elevation provided by the user.

Another good example of an open virtual globe engine would be CesiumJS which implements the latest techniques on the field aimed at a WebGL environment. Other projects as WebGLEarth use under the hood CesiumJS as a rendering engine. However, map viewers on web taking advantage of graphical hardware is a major trend, specially for 2D map projects, such as the work of Jenny, Šavrič, and Liem (2014).

However, at the time this paper has been written, it is hard to find an open solution that allows the creation of virtual globes in mobile devices and WebGL. The space becomes narrower when it

* Corresponding author.
E-mail addresses: josepablo.suarez@ulpgc.es (J.P. Suárez), atrujillo@dis.ulpgc.es (A. Trujillo), josemiguel.santana@ulpgc.es (J.M. Santana), mdelacalle@igosoftware.es (M. de la Calle), diego@consultar.com (D. Gómez-Deck).

[1] Cesium, http://cesiumjs.org/.
[2] OpenWebGlobe, http://www.openwebglobe.org/.
[3] WebGLEarth, http://www.webglearth.org/.

http://dx.doi.org/10.1016/j.compenvurbsys.2015.02.004
0198-9715/© 2015 Elsevier Ltd. All rights reserved.

comes to dealing with several kinds of terrain imagery and models of terrain. A good taxonomy of all kinds of GIS software is offered in the work of Steiniger and Hunter (2013) where it is noticeable the scarcity of mobile solutions. Glob3 Mobile[4] is offered as a solid solution, dealing with most of the use cases and providing a single software project for three platforms.

The problems encountered rendering terrain on handheld devices are discussed in detail in previous works (Siti Aida Mohd Isa, Mohd Shafry Mohd Rahim, & Kasmuni, 2010). Regarding this topic, it is noteworthy our previous work in the field, in which Glob3 Mobile was introduced.

1.2. The Glob3 Mobile project

Glob3 Mobile is a platform independent engine aimed at the creation of visual representations of geospatial data in the most general way possible. It is meant to be run on iOS, Android and WebGL platforms which means that it has to provide a smooth rendering in a broad variety of devices. The nature of the deployment process has been matter of previous publications (Trujillo et al., 2013) but, for the sake of this paper, it is noteworthy that it is used a native implementation for each platform, which means C++ code for iOS, Java for Android and Javascript for the Web as seen in Fig. 1.

This same-core development philosophy has been recognized by the Eclipse Foundation that included the Glob3 Mobile Project in its LocationTech[5] working group.

It is also important to realize that working on many platforms involves a different management of the memory resources in each one of them. For the Objective-C/C++ version the memory has to be allocated and deallocated by the programmer explicitly. This allows us to control when and how the heap memory is going to be deallocated. On the other hand, virtual machines such as Java (for Android) or the Javascript environment (for the Web version) delegate that task to a garbage collector system that is very pleasant to use but ends up in a quite uncontrollable memory management. In addition, such virtual machines add a memory overhead to each allocated instance, so the memory resources are consumed even faster.

Another key aspect is the graphics capabilities of the platforms. They vary largely in the number of GPU cores and memory architecture. For instance, iOS devices have a Unified Memory Architecture (same memory is used for graphics and general computation), which impacts on the amount of available memory and transfer latency. It is also important to know that, due to the mobile focus of the engine, it should rely on the Embedded Systems version of OpenGL (OpenGL ES) or WebGL on the web. The current versions of these APIs have many restrictions compared to their desktop counterpart, and they do not support hardware accelerated tessellation process. Therefore, our rendering techniques should rely on meshes computed on CPU to represent the terrain.

Each one of the devices we work with has its own restrictions in terms of the access and storage of data, the amount of available RAM, the number of triangles that can be rendered per frame and, of course, the user interaction. The core of the system is the terrain renderer, which is based on a quadtree of tiles with a splitting mechanism.

1.3. The Level Of Detail test importance

The tile splitting test impacts extremely on the needed resources and the general performance of the whole engine. Therefore it has been a key step in the development of Glob3 Mobile to achieve a mechanism that allows to show the required terrain detail satisfying the hardware constraints.

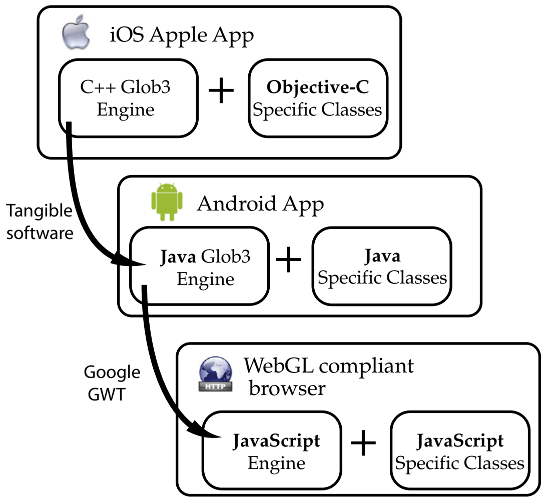

Fig. 1. Glob3 Mobile life cycle. Development process.

Our approach to the problem has consisted in the development of an algorithm that permits to maintain the number of vertices representing the terrain below a given threshold. This splitting algorithm will be explained in detail in Section 2 and the theoretical hardware requirements will be calculated in Section 3.

The transitions between levels of detail should be as unnoticeable as possible to the user. This becomes tricky when we are talking about applications that perform flights along large distances. Moreover, according to our experience, multitouch users tend to perform more random paths through the scene than mouse-keyboard users. Specially they tend to perform rapid camera turns that imply that the scene out of the camera frustum must be prepared for a quick rendering. Our LoD algorithm is independent of the user orientation to achieve a smooth user experience.

All this efficiency concerns will be taken into consideration in a performance study at the end of this document. At Section 4 the rendering process will be decomposed to show the steps needed to generate the final image. Each one of these stages will be analyzed regarding its impact on the global performance, and it will be possible to identify bottlenecks for each platform.

The actual algorithm performance is going to be extracted from experimentation in Section 5, ensuring that any possible camera position ends up in a feasible terrain representation. This guarantees that, given a certain hardware configuration, memory usage will be affordable. At the same time, proper splitting of the rendered terrain and some frustum culling techniques will be necessary to ensure that the graphics hardware can deal with the number of triangles to be displayed.

Finally, the document points out which are the overall performance of the engine in each family of devices. With that in mind, an experiment has been conducted performing representative flights in different devices, which results are shown in Section 6. These results will show the efficiency achieved by our engine for every platform rendering terrain on real-time.

2. A perspective-based Terrain LoD test

One of the main issues when rendering large portions of the earth is how to deal with the extremely big amount of data that forms the terrain model. Many approaches have been developed to solve this problem (Goosen, 2013) but most of them rely on showing different representations of the terrain depending on the camera point of view.

[4] Glob3, an open source 3D GIS multiplatform framework: http://glob3.sourceforge.net.
[5] LocationTech by Eclipse: https://www.locationtech.org/.