Performance evaluation of a 3D engine for mobile devices

Federico Cristina, Sebastián Dapoto ✉, Pablo Thomas, Patricia Pesado

Instituto de Investigación en Informática LIDI,
Universidad Nacional de La Plata – Argentina
Centro Asociado Comisión de Investigaciones Científicas de la Provincia de Buenos Aires

{fcristina, sdapoto, pthomas,
ppesado}@lidi.info.unlp.edu.ar

Abstract. Currently there are several frameworks for the development of 3D mobile applications, but all of them have a common issue: performance is a critical aspect to consider, even more relevant than in desktop computers, which generally have a greater computing power. The profiling or performance analysis tools that these frameworks have can help, to some extent, to determine the possible existence of bottlenecks in the execution of applications. However, this type of tool has certain limitations such as covering only a spectrum of the possible causes of the problem or limiting the analysis to certain scenarios in particular. This article proposes an evaluation and impact measurement of the key features related with performance on 3D mobile applications.

Keywords: Unity engine, mobile devices, 3D applications, performance

1 Introduction

Nowadays, mobile devices are increasingly sophisticated and their technological evolution allows them to execute complex applications with rigorous hardware requirements.

However, when these applications are visual and include three-dimensional graphics, it is possible to notice some degradation in the flow of execution. This loss in execution efficiency is due to the characteristics of the device in which the application is executed, but also to the characteristics of the development tool and / or its implementation.

The number of frameworks for the development of interactive 3D mobile applications is constantly increasing. Each of these frameworks have different characteristics that make them suitable for different types and magnitudes of project.

By choosing a particular framework it is possible to base itself on various criteria such as: the existing community, the supported coding languages, the ease of use, the quality of the resulting 3D graphics, among others. However, one of the main points of interest is to obtain a good performance in terms of visualization and fluency.

In mobile devices, the limited processing capacity (for example, compared with that of desktops) is more relevant. For this reason, it is important to know the capabilities and limitations of the framework to be used in terms of visualization performance.

In order to achieve this objective, this paper proposes an evaluation that allows isolating, analyzing and dimensioning the incidence of the main characteristics related to the visual performance of 3D mobile applications.

This evaluation provides support to the software engineer of 3D mobile applications, enabling the identification of factors that slow down the visual performance of these applications, and allowing them to adjust the critical points until achieving the desired fluidity.

The rest of the paper is organized as follows: section 2 describes the motivation of the proposed analysis; section 3 presents the evaluation in detail; section 4 shows the experimentation carried out and, finally, in section 5 the conclusions and the future work are exposed.

2 Motivation

The gestation of the performance analysis proposed by this work begins during the development process of two mobile applications: R-Info3D [1] and InfoUNLP3D [2]. The first application is a 3D learning environment of basic algorithmia, while the second one is a virtual 3D scenario of the Facultad de Informática - Universidad Nacional de La Plata. Both projects resulted in immersive applications, which by their nature present a high computational cost, mainly in terms of visualization [3].

Both in the implementation process and in its subsequent execution in different mobile devices, limitations were found regarding the obtained performance. We proceeded to recognize the critical points that affected visual fluidity and determined certain thresholds that should not be overcome. This analysis led to modify the applications to achieve better performance in different types of mobile devices.

Prior to the beginning of the development of the aforementioned tools, an analysis of the main free-use engines for the development of 3D mobile applications was made. The considered engines were: Unity [4], one of the most popular and simple to use; Unreal Engine [5], which compared to other engines is somewhat more complex to use and with higher hardware requirements; CryEngine [6], thought mainly for first person 3D developments, its installation and use are not trivial processes.

R-Info3D and InfoUNLP3D were developed with the Unity framework. Given that there is no optimal framework in all possible aspects, the choice of Unity over the rest of the 3D mobile development frameworks analyzed was based on a series of factors, among which stand out:

- Tutorials: there are a lot of tutorials and examples that guide the new user in the learning process of the use of the framework. These tutorials are categorized according to the type of development, they are audiovisual and are provided with all the necessary elements needed (objects, audios, images, scripts, etc).
- Documentation: the user manual [7] is comprehensive, easy to understand and properly subdivided into different categories. It contains a lookup engine that facilitates the search of a particular topic or functionality.

- Software / hardware requirements: the system requirements are considerably lower compared to other similar frameworks. This, accompanied by a simple and fast installation, encourages the user / developer to start quickly with the use of the framework.
- Available components: it has a repository (Asset Store) where it is possible to find a wide variety of components that can accelerate the development of applications. It contains a component lookup engine that allows complex searches through different types of filters.
- Ease of learning / use: the framework is versatile and allows working in two different languages: C # and javascript. This, added to the documentation, the tutorials, the repository and the existing community, generates an ideal scenario to quickly learn to use the framework and solve any problem that may arise in the process.
- Community: the great popularity of Unity is a major factor when choosing it. It is the most used 3D framework and its community is composed of more than two million users [8]. It contains a forum subdivided into categories where it is possible to raise the situations or problems that may arise during the application development process.

There are different proposals for performance analysis of 3D engines for mobile applications. One of these proposals [9] evaluates the performance of the applications taking into account the consumption of CPU and / or GPU that they generate. Other works [10, 11] base their evaluation on an analysis of the main features and functionalities of the frameworks, resulting in a comparative table or list.

None of these works adopt the approach proposed in the present paper, that is to give some kind of guidance for the development of 3D mobile applications based on the final achieved visualization performance.

3 Proposed evaluation

The main objective of the proposed evaluation is to isolate each of the main characteristics covered by a 3D application; especially those that have a direct impact on the performance, response time and flow of execution of the applications generated with the engine [12].

Characteristics such as the number of polygons, lights and shadows, the use of textures and / or transparencies, the visualization of particle systems and the calculation of the physics of objects that make up the scene are examples of the main items to be evaluated.

Although the incidence of these characteristics on performance varies according to the software and hardware on which the application is executed, the tests carried out show that there is a common pattern of performance degradation in relation to the increase in visual requirements.

Based on the previously mentioned set of characteristics, a series of independent tests is defined to evaluate the performance. These tests are listed below:

1. Basic Mesh Rendering: simple objects without texture in motion are presented on screen in a scene without lighting or shadows. Objects must rotate continuously at

constant speed. The number of objects on the screen will grow according to the time elapsed. The number of frames per second (FPS) is accounted throughout the simulation according to the number of objects.
2. Complex Mesh Rendering: it consists in visualizing a complex object in movement, which must contain a high number of polygons. The rendering distance (clipping plane) increases as the test progresses. The FPS is accounted throughout the simulation depending on the rendering distance.
3. Lights & Shadows: a simulation similar to the basic mesh rendering is performed, but in this case the scene contains lighting and objects with projection and reception of shadows. The FPS is accounted according to the number of objects.
4. Textures: a simulation similar to the basic mesh rendering is performed, but in this case the objects have complex textures, such as transparencies, reflections, etc. The FPS is accounted according to the number of objects.
5. Particle Systems: a scene is created where new instances of a particle system are progressively presented (for example smoke, sparks, explosion, etc.). The FPS is accounted for according to the number of particle systems.
6. Physics: a simulation similar to the basic mesh rendering is performed, but in this case the objects are subject to physics rules, such as gravity. The FPS is accounted throughout the simulation according to the number of objects.

The use of this set of tests simplifies considerably the task of determining the critical points in the developed applications, thus enabling a better calibration of the analyzed characteristics.

In particular for this work, cubes were used as simple objects, while for the rendering test of a complex object, the building model of the Facultad de Informática was chosen. The model used in the InfoUNLP3D application, contains more than 500,000 polygons and a large number of windows. Due to the latter, for the textures tests, "glass transparency" was applied to the elements. As for the particle system, one similar to that used by the robot in R-Info3D when executing a repositioning instruction (Pos) was generated.

4 Experimentation

The experiments consisted in performing the test set previously detailed over a set of mobile devices with different characteristics. The devices used in the tests were three smartphones and two tablets: Samsung Galaxy S2 (smartphone), Samsung Galaxy J5 (smartphone), LG L5 II (smartphone), Asus MemoPad FHD10 (tablet) y Acer B1-730 (tablet). These devices present considerable differences preformance-wise, and have different hardware architectures, such as ARM and x86.

Figures 1 and 2 show the average values of the results obtained from the tests executed over all the devices, considering two different quality rendering configurations that Unity provides: *Fastest* (the lowest) and *Fantastic* (the highest) [13].

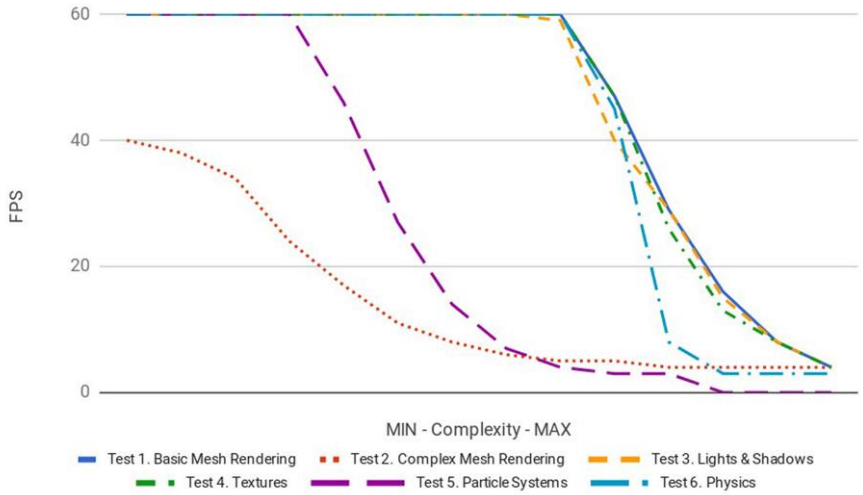

Fig. 1. FPS. Evolution of each test, *Fastest* quality.

In all cases and depending on the type of test, the information is normalized according to the possible range of minimum and maximum values for each test. In tests 1, 3, 4, 5 and 6 the number of objects on screen starts at 1 and gradually increments throughout the simulation until reaching a high value, such as 10000 objects. For the test 2, the evolution in this case is the rendering distance, that can vary from a minimum of 200 up to a maximum of 2000 distance units. Additionally, the ideal frame rate considered is 60 FPS.

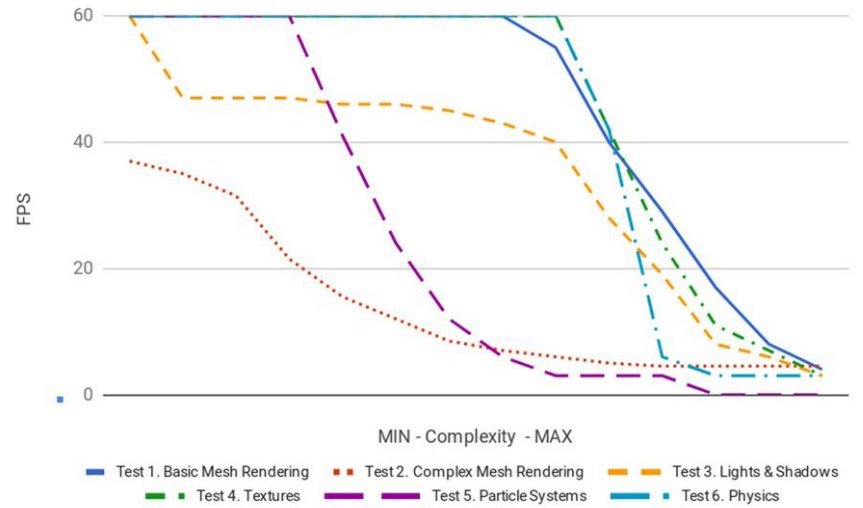

Fig. 2. FPS. Evolution of each test, *Fantastic* quality.

The results of the tests offer an interesting set of conclusions, which allow the possibility for the correspondent optimization of the application. The following observations results from applying the proposed evaluation specifically for the current case of study.

- Basic mesh rendering using both *Fastest* or *Fantastic* quality does not present performance differences. Both curves show almost the same evolution throughout the simulation.
- Light affects the performance under *Fantastic* quality, but not under *Fastest* quality, configuration which seems to completely ignore lights calculations. Lights/shadows rendering is one of the key factors in respect to visual performance, as it will be later explained in this paper.
- The amount of polygons (+500k) under the complex mesh simulation is excessively high for the set of mobile devices used in the tests in order to achieve a fluent visualization. Both *Fastest* and *Fantastic* quality settings suffered the same degradation.
- Textures application does not seem to considerably affect general performance, at least with the selected texture (which includes transparency), applied through a standard shader, and with a moderate texture size.
- Being 2D, particle systems are not considerably affected by quality change; but a high number of systems prohibits the correct execution of the simulation, given the large amount of individual calculations required for each of the particles of each system.
- Both Collision detection and Physics have a relatively minor impact over performance, when a reasonable number of objects are displayed in the scene (for instance, less than 2000 cubes).

Figure 3 shows the FPS coefficient between *Fastest* and *Fantastic* quality configurations. A value of 1 implies that the test behaves the same under both configurations; whereas a value greater than 1 implies a factor of degradation under *Fantastic* relative to *Fantastic*, and a value less than 1 implies the opposite. Thanks to test 3 is possible to clearly appreciate the performance degradation under *Fantastic* configuration, with an average of 50% less fluent than *Fastest* configuration (which obviously relegates graphical quality in this matter).

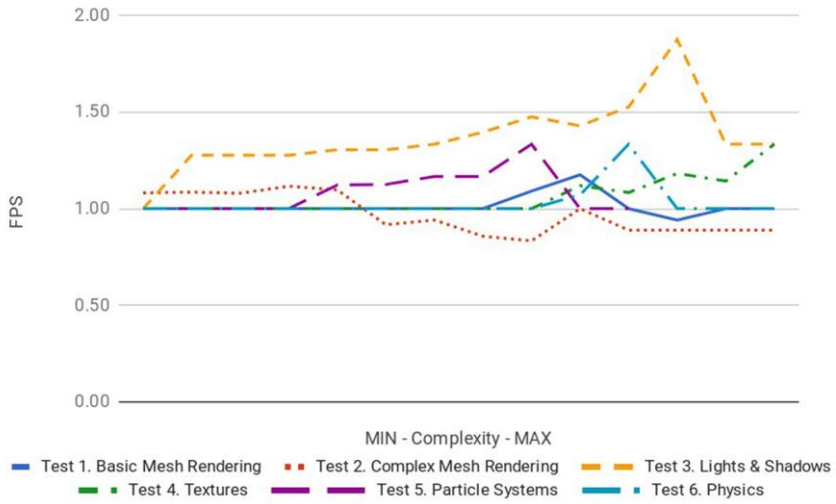

Fig. 3. FPS. *Fastest* vs. *Fantastic*.

Additionally, as previously explained, the complexity of the object to be renderized (test 2) equally impacts on both *Fastest* and *Fantastic* configurations, and after certain threshold there is practically no distinction between each configuration.

Thanks to this analysis, it was possible to achieve an acceptable balance between graphical quality and visualization fluency in each of the developed applications in Unity. Figure 4 presents a scene of InfoUNLP3D application with the highest possible level of detail, considering characteristics such as textures, lights and shades; but with a extremely poor FPS performance. Figure 5 shows the same scene once modified in order to achieve máximum fluidity, although relegating in excess the characteristics previously mentioned. Figure 6 presents the achieved results thanks to the selected optimal calibration, based on the results of the tests. In this calibration both image quality and visualization fluidity is considered. For instance, the use of spot lights is omitted in order to avoid shadow processing, given that the objetive of the application is not to present a photorealistic rendering, but to serve as a reference guide for students.

Fig. 4. InfoUNLP3D. Maximum level of detail.

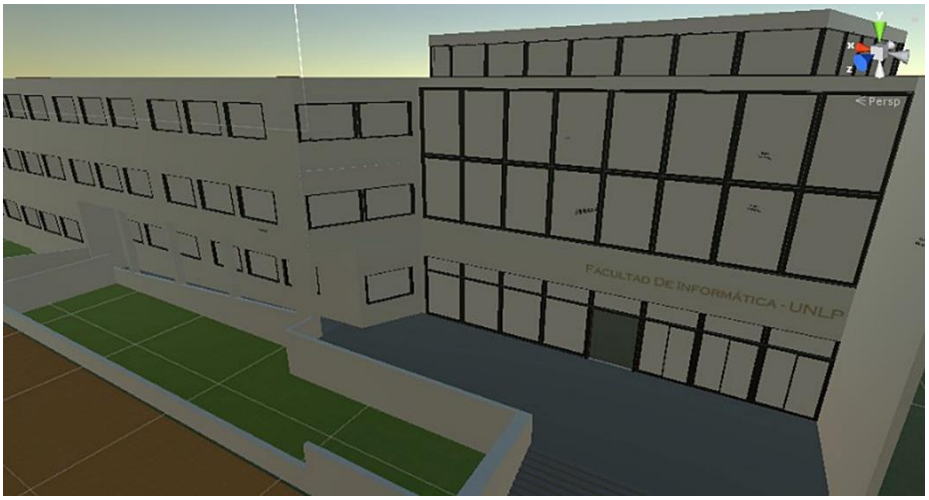

Fig. 5. InfoUNLP3D. Minimum level of detail.

Under this context, a new feature was added to the Info UNLP 3D application that allows the user to configure the rendering distance (and therefore the number of polygons on the scene). By avoiding rendering the most distant elements in the scene, fluidity is gained without losing functionality, which is based on navigate the building gathering information about the nearby rooms. This new option allows the use of the application on devices with lower processing power.

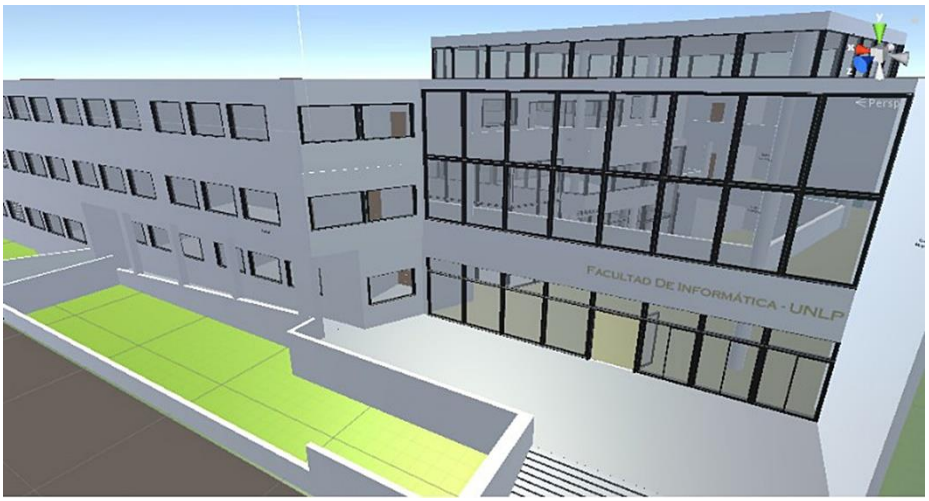

Fig. 6. InfoUNLP3D. Selected optimal calibration.

5 Conclusions and future work

The present paper proposes a set of heuristics that allows to establish the key factors that degrade visualization fluidity of 3D mobile applications.

Thanks to the proposed evaluation method it is possible to identify these factors in order to redesign an application and considerably reduce performance problems.

As a proof of concept and to test its effectiveness, the evaluation method was applied over Unity engine. Simulation tests were carried on, gathering as a result the required information in order to execute the correspondent optimizations on the applications developed with the mentioned engine.

As starting point, a 3D mobile application developer has a set of guidelines to consider in order to aid visual performance optimization.

As a future work, the same evaluation method will be applied over other 3D mobile application engines such as Unreal Engine or CryEngine, and ideally will be perform a comprehensive performance comparison between these frameworks for each of the evaluated characteristics.

6 References

1. Cristina, F.; Dapoto, S.; Thomas, P.; Pesado, P. “3D Mobile Prototype for Basic Algorithms Learning”. Computer Science & Technology Series - XXI Argentine Congress of Computer Science. Selected Papers. ISBN: 978-987-4127-00-6, pages 239-247. 2016.
2. Cristina, F.; Dapoto, S.; Thomas, P.; Pesado, P. “InfoUNLP3D: An interactive experience for freshman students”. Computer Science & Technology Series - XXII Argentine Congress of Computer Science. Selected Papers. ISBN: 978-987-4127-28-0, pages 249-256. 2017.
3. Linowes J. “Unity Virtual Reality Projects”. ISBN-13: 978-1783988556. 2015.
4. Unity. https://unity3d.com

5. Unreal Engine. https://www.unrealengine.com/
6. CryEngine. https://www.cryengine.com/
7. Unity online manual. http://docs.unity3d.com/Manual/index.html
8. Unity Blogs: The Unity Community. https://blogs.unity3d.com/2014/03/11/we-got-karma/
9. Messaoudi F., Simon G., Ksentini A. "Dissecting Games Engines: the Case of Unity3D". International Workshop on Network and Systems Support for Games (NetGames). Electronic ISSN: 2156-8146. 2015.
10. Akekarat Pattrasitidecha. "Comparison and evaluation of 3D mobile game engines". Chalmers University of Technology. University of Gothenburg. 2014.
11. Petridis P., Dunwell I., Panzoli D., Arnab S., Protopsaltis A., Hendrix M., de Freitas S. "Game Engines Selection Framework for High-Fidelity Serious Applications". International Journal of Interactive Worlds. Article ID 418638. DOI: 10.5171/2012.418638. 2012.
12. Optimizing Graphics Performance. https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html
13. Unity Manual: Quality Settings. https://docs.unity3d.com/Manual/class-QualitySettings.html