

Effectiveness of virtual world timers in educational physics simulations

Jorge Lima

ECT/UTAD – University of Trás-os-Montes e Alto Douro, Vila Real, Portugal, jlima@utad.pt

Leonel Morgado, Benjamim Fonseca, Paulo Martins, Hugo Paredes

GECAD/UTAD – University of Trás-os-Montes e Alto Douro, Vila Real, Portugal, {leonelm, benjaf, pmartins, hparedes}@utad.pt

Abstract

In this essay we will present a Physics education project developed in virtual worlds. After prototyping in Second Life, it was decided to port it to its free-software alternative, OpenSim. Several challenges were encountered, namely the ability of sampling an object's position and velocity more than twice per second. This performance was unacceptable for our project, which motivated a deeper study of timer effectiveness in virtual worlds, including also ScienceSim. Results indicate that only Second Life can support effective timers able to supply Physics simulations with more than two samples per second.

Introduction

The Physics education project at hand was initially developed in the Second Life virtual world, for its ability to simulate gravity and collision detection between complex objects, without any effort on our side. We intended to develop several physics simulations based on gravity, such as Newton tubes, projectiles, and parachutes.

The term “virtual world” has been applied to a vast range of environments that can provide their users with creative freedom, through built-in tools for developing and sharing content [1]. As an example of this diversity, we can present several two-dimensional worlds (WorldsAway, Habitat, The Palace), three-dimensional ones (ActiveWorlds, Second Life), isometric (also known as 2.5D) (Furcadia, Habbo Hotel), and even text-based, if we retrospectively apply the term to its predecessor (MUD). Virtual worlds differentiate themselves by offering usability [2], socialization, and collaboration [3]. In a virtual world users can experience a physical, persistent, shared, interactive, remotely accessible environment, which provides an immersive experience [4]. User can convey their identity to others through the use of customizable “avatars” [5], a term that appears in Hindu mythology as the representation of a god on Earth.

We shall restrict the term “virtual world” to platforms that support rapid collaborative development and enjoyment of interactive 3D content, that support

physics simulations such as ours, in a shared social space where students and teachers can exchange ideas, with access control features to let educators decide by whom and how the software will be used, using client-server architecture, so as to enable students to remotely access the educational activities.

A well-known example of such a platform is Second Life, which, after having gained significant media exposure, was adopted as a development tool for business and educational projects [6]. Its nature as a privately-hosted, costly platform eventually led to the development of compatible alternatives, OpenSimulator and ScienceSim being amongst the most used.

OpenSimulator was developed according to the “free software” philosophy according to which any person may contribute to its development [7]. ScienceSim, while based on the OpenSimulator source, has the advantage of being sponsored by Intel and 3Di [8], and their significant programming resources. Their goal is to maintain a separate version of OpenSimulator and correct some issues with it. [9]. Nowadays, the educational sector makes extensive use of these platforms, producing an increasing amount of studies and projects [10].

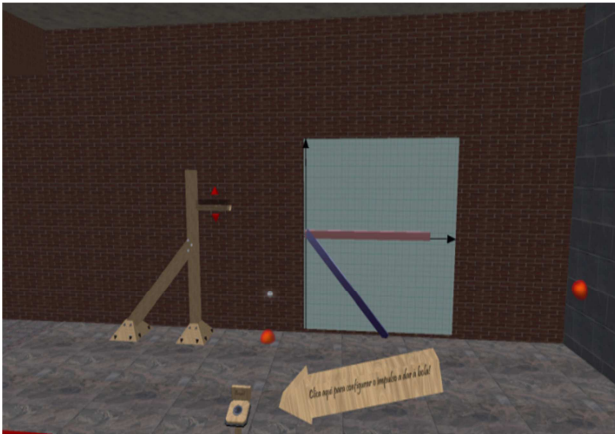
Background

Virtual worlds enable the development of a vast array of projects. Using Second Life incurs steep costs, which makes the search for viable alternatives ever more appealing, as their development matures and stabilizes. Even though there is no licensing fee, there may still be hardware, hosting and maintenance costs associated with using your own server. The choice of platform depends on the type of project. In order to contextualize these choices and the results obtained in this study, we must analyze a specific case - a physics education project. All virtual worlds that meet our aforementioned definition allow physics simulation, facilitating the implementation of small experimental activities. These enable autonomous learning by students that can remotely access the server to study various parts of the syllabus, or even solely in the formal context of a classroom, by using access control features. Since virtual worlds also exhibit collaborative capabilities, it is possible to use them as a tool to support group assignments. We shall present a clear example of one such incompatibility and its

substantial impact on this project. However, even though these virtual worlds aim to be compatible, some differences still exist which will be the subject of our research.

The context: Physics Education

This study arose from the development of an eLearning solution for physics education based on virtual world technology. Specifically, in order to explore the differences between all three platforms, the development of a simple case was attempted: a ball rolling down a diagonal chute. The configurable variables were the ball's initial position relative to the chute, and the chute's distance to the ground, with the end goal of measuring the ball's velocity upon leaving the chute, registering its point of impact on the ground, and drawing graphs for the velocity vector's X and Y components.



Picture 1 – the chute experiment

Materials

The test computer used was an Intel Core2 4300 at 1.8 gigahertz, with 3 gigabytes of RAM. Tests were executed in OpenSimulator version 0.7.0.2 running under Windows XP SP3. The OpenSimulator server was empty aside from the test script, and used the default configuration. We accessed the server by running a graphical client on the same machine (Hippo OpenSim Viewer) as well as a lightweight text-only one (Radegast). All programs were closed except for vital windows tasks, and Task Manager was used to set execution priority to Low on everything but the server, which was set to High. It was not possible to obtain this information about the Second Life and ScienceSim servers, since they are not hosted by us.

Method

Mathematical analysis – solving equations – is even harder for computers than it is for humans, which leads to an alternate approach: Simulation, approximating a smooth curve through periodical calculation of points.

This is an entirely valid and widely-used approach[12], especially if we consider that Physics itself has recently produced results that, according to some, may indicate the Universe itself is composed of points of space and time like a computer simulation [13][14].

In Physics, the unit of frequency is the Hertz, defined as one cycle per second [11], of which megahertz and gigahertz are multiples. If we increase frequency we can produce more samples in the same period of time, which brings our set of points closer to the original curve. This increase in Hertz may lead some to believe that computers with more Hertz are always faster, but this is not always true, there are more factors to consider, such as software efficiency.

The word “**algorithm**” is the method with which we solve a problem[15], and can be classified into different orders of magnitude of efficiency with a mathematical construct dating back to 1892: the “**big-O**” notation.[16] If we have one any N number of things to buy, and go to the supermarket N times to buy one at a time. This order of magnitude is represented as $O(N)$, whereas taking only one trip would be represented as $O(1)$. Using $O(N)$ methods where $O(1)$ is possible, divides computing power by N.

Humans can only get everything in one trip depending on how much load they can carry. Beyond this limit they will still divide their purchases into N trips. Algorithms also have load limits, and much like water suffers a phase transition to ice at 0°C , they can suffer a phase transition after their limits.[17] We can expand the load limit for humans with a car, but only N items will fit in your trunk, after which you will still have to divide your shopping list into N trips – when the algorithm is inefficient, upgrading hardware makes little difference. The computer equivalent would be memory, and a program limited by memory would be **memory-bound**, but algorithms can also be bound by the CPU's processing power or Input / Output transfer speeds.

There is also the matter of time. We could drive across Europe, or all the way to Asia, but perhaps we would need to get there faster by catching a plane. Driving would not be **effective** for researchers that need to attend conferences in different countries scattered all over the world (and be back in the lab as soon as possible).

Similarly, Physics simulation projects have special requirements: Precise, timely, deterministic calculations of periodic phenomena.

We define **effectiveness** as the ratio of observed to ideal behavior, executing a number of samples in the requested amount of time.

Timers are mechanisms used by operating systems to schedule periodic tasks. The number of tasks we can schedule, how they are sorted, and how fast they can be processed depends on the scheduler.[21] Task scheduling is itself a challenge which has been studied for a long time, and is still an active research field. Many algorithms exist that schedule tasks with different efficiency, effective for different purposes.[18] [19] [20]

If we set up a timer to take a sample ten times per second, during ten seconds, an effective timer would have taken ten samples after one second had past, twenty after two seconds, and a hundred in total.

This would be an ideal timer, 100% efficient and very effective for our project, with no deviation from the specified behavior.

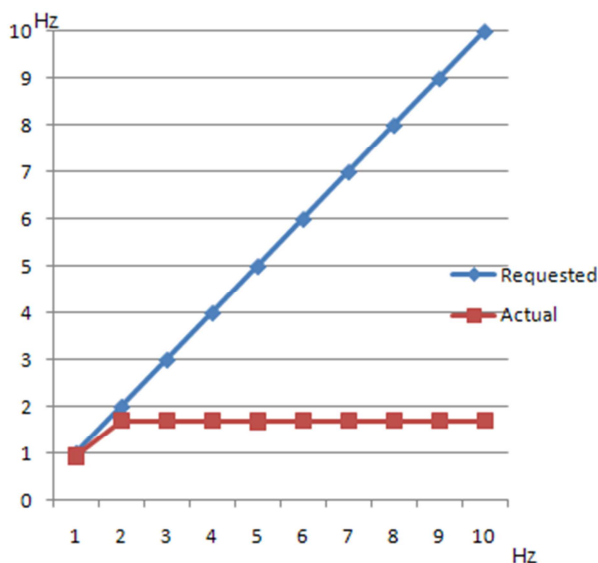
Even with 2001 computing resources, a timer running under the UNIX operating system was able to get responses from a Web Server in 2 to 18 microseconds, which would correspond to a response frequency of 2 to 18 megahertz. [23]

This would be an extremely effective timer for our project, far more than we need, if we consider that the PAL standard used by televisions updates the screen at 60Hz. [22]

Let's see how this compares to virtual worlds.

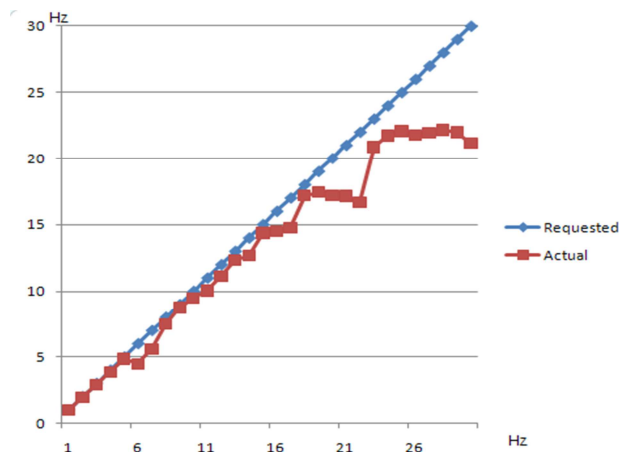
Results – Virtual World Timers

We will now compare two sets of data collected, one for OpenSim, the other for Second Life. We ran the standard built-in timers with an automated test that gradually tried to increase the sampling rate. In the case of OpenSim, we ran it from 1Hz to 10Hz, which was the originally intended performance.



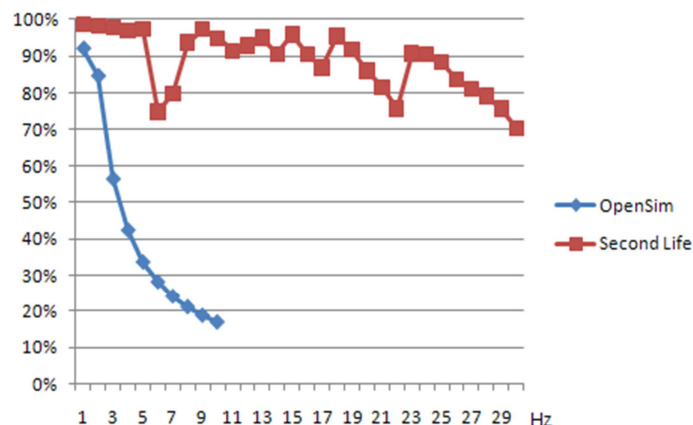
Graph 1 – OpenSim performance

With Second Life sampling, it became clear that it would not converge as quickly. There were many fluctuations – we were not able to obtain a perfectly isolated test environment, so many variables were not controlled for. Nevertheless, results clearly show that it is possible to obtain higher performance, for more frequent sampling, resulting in more points of data and solid physics results.



Graph 2 – Second Life performance

In order to provide a better picture of compared efficiency, we took the previous data and divided actual performance by ideal performance. This ratio, as a percentage, is depicted below:



Graph 3 – Comparison of OpenSim and Second Life

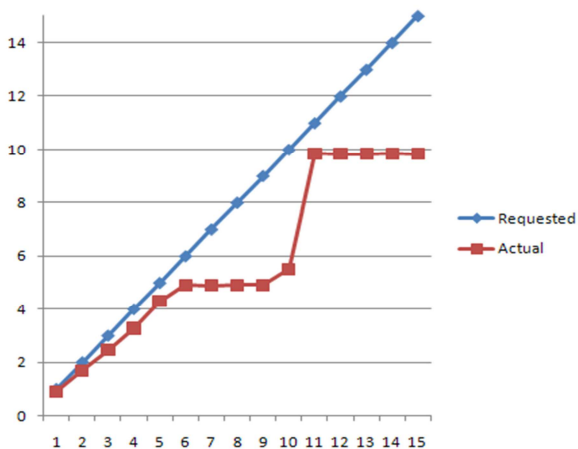
Unconventional timers

Though standard programming practices encourage us to implement our own solution in the absence of a viable one, this is hardly ever possible in the virtual worlds we used, since they are designed to merely react to events inside them, not supporting standard language mechanisms such as external function libraries. The only way to modularize code is by sending messages to scripts running in parallel. However, while processing an event, the system will not be able to receive any more, and this includes all kinds of messaging. They are handled on a first-come first-serve basis, and there are limits to how many can wait in the queue. After a very long time, we nevertheless attempted to implement our own timers using very simple code:

```
while(--numSamples > 0)
{
    llSleep(1/samplingRate);
    doSomethingPeriodically();
}
```

There are many drawbacks to this implementation. It consumes all available CPU, and for our dual core test computer, only 2 sampling scripts can be run simultaneously without overloading the computer. Preliminary testing demonstrated a reliability of 99% between 2 and 30Hz. We hypothesize that an undocumented issue with the llSleep function could be preventing us from executing slower or faster, since execution can sometimes fail silently outside these sampling rates. Further examination will be required.

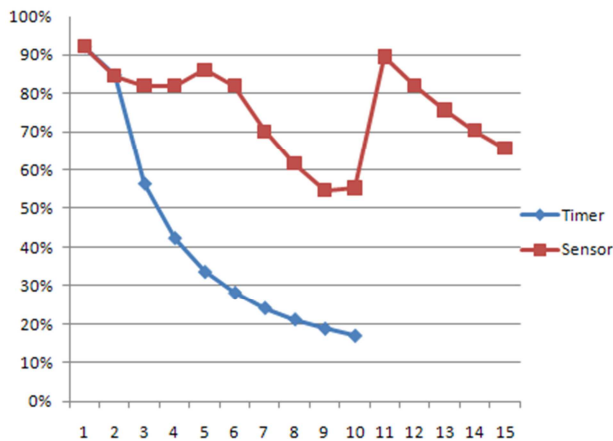
llSensorRepeat can detect data in the surrounding environment on a regular basis. We also implemented a timer by ignoring any returned data, though care must be taken to supply it with the correct input. Our results, shown below, indicate a sudden shift in behavior around 10-11Hz, to which we must adjust.



Graph 4 – Sensor-based timer sampling

Unlike Second Life, we were fully in control of the server this time, and care was taken to prevent interference, as described in Materials and Method.

We don't have, at present, a solid explanation for this phenomenon. A closer inspection of the source code could reveal its cause, as well as further testing under an operating system with real-time guarantees such as RTAI or RT-Linux.



Graph 5 – Sensor efficiency vs. Timer

As we can see from this table, our sensors are indeed very efficient compared to the default configuration of OpenSim.

Afterword

Upon inspection of the OpenSim source code, however, it was discovered that a hidden option, not present in the default configuration files, is preventing users from creating timers that trigger more frequently than 0.5 seconds.

```

LSL_API.cs:
protected float m_MinTimerInterval = 0.5f;
(...)
public void llSetTimerEvent(double sec)
{
if (sec != 0.0 && sec < m_MinTimerInterval)
    sec = m_MinTimerInterval;
(...)
}

```

Changing this variable would have solved our problem, had we not adopted a completely different approach in the mean time. The phenomena we were sampling were linear in nature, therefore possible to extrapolate from one sample, producing graphs that perfectly match physics theory and thus serve their educational purpose, even if they do not represent the inner workings of the OpenSim physics engine, Open Dynamics Environment.

Conclusions

Even though compatible alternatives exist, these do not present the same functionality, usability and stability as Second Life. In the absence of financial constraints, Second Life seems to offer more guarantees. OpenSimulator in its default configuration does not support projects strongly dependent on timeliness, such as our Physics simulation, but there are many ways to work around its restrictions.

The Sensor-Timer implementation here described can be used to circumvent restrictive configurations when developing a project on a server hosted by a third party, not under our control.

It is not intended to extrapolate these conclusions to the general case of projects developed on these platforms, a lot of which do not depend on this type of functionality.

Future work

Our conclusions were drawn from a very specific type of project with specific needs. Each platform's functionality encompasses a much wider range of possibilities that could be the subject of further studies. Exploring other physics education problems could bring a deeper understanding of the differences between these platforms. It's possible there will be ways to work around these problems, and further testing is required to pinpoint their cause.

Acknowledgements

We would like to thank the physics teachers, Paula Batista and Helena Fernandes, from Escola Secundária Camilo Castelo Branco, for their guidance in this study.

References

- [1] C. Ondrejka, "A Piece of Place: Modeling the Digital on the Real in Second Life", 2004
- [2] D. A. Bowman, J. L. Gabbard, D. Hix, "A Survey of Usability Evaluation in Virtual Environments: Classification and Comparison of Methods", 2002
- [3] S. Benford, C. Greenhalgh, T. Rodden, J. Pycock, "Collaborative Virtual Environments", Communications of the ACM July 2001/Vol. 44, No. 7
- [4] E. Castronova, "Virtual Worlds" pp.2-4, 2001
- [5] E. Castronova, "Virtual Worlds" pp.7-9, 2001
- [6] W. A. Warr, "Social software: fun and games, or business tools?", Journal of Information Science, 2008
- [7] OpenSim community, http://opensimulator.org/wiki/Main_Page, (Accessed on 19/01/2011)
- [8] ScienceSim foundation, <http://www.sciencesim.com/wiki/doku.php/sponsors>, (Accessed on 19/01/2011)
- [9] Fundação ScienceSim, <http://www.sciencesim.com/wiki/doku.php/foundation/deprocedure>, (Accessed on 19/01/2011)
- [10] L. Morgado, "Os mundos virtuais e o ensino-aprendizagem de procedimentos", Educação & Cultura Contemporânea, ISSN 1807-2194, 13 (6), 35-48. 2009
- [11] Bureau international des poids et mesures, "Le Système international d'unités (SI), 8e édition", 2006
- [12] A. Law et al, Simulation Modeling and Analysis, pp. 1-19, 1991
- [13] Gregory Chaitin, "Epistemology as Information Theory", 2005
- [14] Mike Stannett, "The Computational Status of Physics", 2008
- [15] D. Knuth, The Art of Computer Programming, pp. 1-9. 1973
- [16] D. Knuth, The Art of Computer Programming, pp. 104-107. 1973
- [17] A. Braunstein et al, Complexity transitions in global algorithms, 2002
- [18] A. Tanenbaum, Operating Systems Design and Implementation 2nd ed., pp. 82-93, 1994
- [19] A. Tanenbaum, Modern Operating Systems, 2nd ed., pp.132-152, 2001
- [20] O. Khalid et al, Dynamic Scheduling of Virtual Machines Running HPC Workloads in Scientific Grids, 2009
- [21] A. Tanenbaum, Modern Operating Systems, 2nd ed., pp.327-332, 2001
- [22] Recommendation ITU-R BT.470-6, Conventional Television Systems, International Telecommunications Union, 1998
- [23] A. Tanenbaum, Modern Operating Systems, 2nd ed., pp. 332-333, 2001