Game engines

Mohamed Ghazal

April 30, 2022

In this document an overview of many 3D physics engines will be made. This will facilitate the arduous process of choosing a suitable 3D physics engine for our application. This is completely dependent of the efficiency of the implementation and its openness, and the following criterion:

- Usability.
- Technical support and documentation.
- Performance.
- Language: the language used to implement and extend the engine.
- Portability: the supported platforms.
- Licensing.

1 Unreal

Unreal engine 4 uses the C++ as a programming language, but working with it doesn't require C++. Instead the developer Unreal engine 4 uses its scripting system and blueprint[1]. To write these scripts and use blueprints one can install plugins to many existing editors and suitable IDEs, such as Unreal Engine Tools by Tiago Patrocinio for VS Code. Unreal Engine 4 being open source offers a more flexibility in the process of development, in comparison with the Unity 3D engine, which offers open source only for users with a professional licence[2].
Just like the Unity 3d engine, unreal has its own programming editor Kismet to streamline the development process. In addition to that, Unreal engine supports importing/exporting content from/to Maya, 3D Studio MAX and blender and includes all possible developer toolkits such as Windows Phone SDK, iOS SDK, Android SDK and Android GDK. Furthermore Unreal Engine offers free technical support. Lastly, Unreal Engine 4 cannot be installed on Mac Os and Linux, and only compatible with windows systems. Thus, making the portability of Unreal engine projects into other systems partially questionable[2].
As for the development environments, Unreal engine 4 projects can be developed in Mac Os, Linux and windows. According to Eleftheria Christopoulou and Stelio Xinogalos, Unreal engine 4 has a more complex interface having an editor with many windows in comparison with Unity 3D. Furthermore, this engine uses a Blueprint Visual Scripting system that is based on node-based interface. This is a graphical editor for defining objects and classes, which can be very useful for non-programmers. However, Unreal engine is more suited for experienced users, as it support the visual scripting system, thus having a very complex graphical environment with a steep learning curve. Moreover, Unreal engine requires high

performing hardware :q[2]. Finally, it can be said that Unreal engine has a wide community offering a lot of tutorials and help for newbies and veterans alike.

2 Blender

Blender is a 3D graphics software used for animation, visualization and modeling, and it has a real-time 3D viewer and GUI. In addition to that, blender has a scriptable interface to import/export data from/to it[3]. Morten Lind and Amund Skavhaug describe blender as (mesh) modelling and animation software, that integrates the bullet physics library, referenced in 3. Furthermore, Morten Lind and Amund Skavhaug see the game engine that comes with blender as advanced and efficient. Besides, with the embedded Python interpreter developers can implement real-time interaction of external control software with the internal game engine logic. This game engine can be used to simulate some mechanical phenomenon observed on objects under motion, like sliding or falling.[4]
Morten Lind and Amund Skavhaug used blender to emulate production devices in real-time, and used CPU-Workload and frame-rate to determine the systems performance. In addition to that, they used the feature *Enable All Frames*, which drives the emulation of the terrain in free drive. Mainly, this features enables blender to only limits the simulation by the limits of the hardware it is running on, i.g. RAM, CPU, GPU, etc[4]. Morten and Amund see this adaption of blender to do a task it wasn't considered by the designers of blender to be an intended use of the software[4] is a huge indication of the adaptability of the software and the performance that was achieved, with and without *Enable All Frames* active, was consistent and its complexity slightly sub-linear[4], which was a welcome surprise to the two researchers. This result wasn't explained nor thoroughly investigated, but they attributed these results to two plausible explanation[4]:

- Using Laptops that are hardware targeted for optimization in everything related to power consumption, made users biased toward them a less performant than desktop PCs, but reducing the number of power conserving features implemented to increase time on batteries and reduce the amount of heat generated resulted in an observable increase in performance[4].
- The limiting factor on systems on systems that performed badly on low problem complexity were those with GPUs with low quality. This resulted in an overload of the CPUs, that aren't specifically designed to handle the this type of workload, which is GPU-intensive[4].

Morten Lind and Amund Skavhaug see these results as consistent with the sub-linear behaviour overall, and that they lead to the conclusion, that there are no super-linear effects that would negatively influence the performance of the system in higher complexity orders. Thus, they concluded that their approach is applicable in a great amount of use in the emulation of physical objects, and the mechanical phenomenon observed on them under motion, given the available hardware can support this application[4].

3 Bullet

As mentioned in the section 2 Bullet is the physics engine used in blender to simulate physical phenomenon. A. Boeing and T. Bräunl evaluated real-time physics simulation systems, and Bullet is one of these systems, and compared them with similar systems. In contrast with the previously discussed physics engine, Bullet doesn't have a GUI interface to interact with, and for this reason it is used as the internal physics engine for light weight application. A. Boaing and T. Bräunl describe Bullet as Physics

Library with an open licence. Being relatively new to the publication date of this evaluation, Bullet didn't offer any new features in comparison with other Physics Engines studied in this evaluation, and it is described as a hybrid impulse and constraint-based engine that supports both variable and fixed time steps. Bullet includes a partial GPU physics implementation[5].
Bullet is free for both educational and professional use, implemented in C++, and compatible with most platforms, Linux/Win32/ Mac[5], and can be described as very portable across devices and operating systems.
A. Boeing and T. Bräuml described these systems as an abstraction layer providing a uniform, extensible interface to complexer physics engines, like the relationship between Blender and Bullet. They also determined that Bullet, as an case study in their evaluations, provides the necessary properties for game development. Overall Bullet performed best among the open systems tested in the evaluation, and outperformed many commercial systems in some aspects[5].
A more resent study of the Bullet physics library is done by E. Izadi and A. Bezuijen in regard to sheer tests, which proved that Bullet can be used by researchers to in various fields of study to simulate a growing number of physical phenomenon. Bullet proved to be very accurate in this study, even though Bullet wasn't designed to simulate the dynamics of rigid bodies. Bullet successfully captured the majority of properties observed in the real-world experiments performed in the laboratory, e.g. the chain force network, displacement vectors, etc[6]. This Study was in very specialized field of study, but more recent evaluations, such as [7], [8] and [9], in other specialized filed of study targeting other physical phenomenon proved the promising nature of these physics engines in general, and especially Bullet.

4 Unity

Unity 3D is created by Unity Technologies and it can be used to create both online and offline games. This physics engine is well documented and offers a variety of online tutorials for better entry opportunities. It utilizes the concept of assets, which can be copied as files in the source directory or imported directly. This versatility allows the developers to gain much control over the assets, that they are using. Their modular design is one of the main advantages of this physics engine. In addition to that, it is very portable, and works on many platforms, such as Mac os, Windows, Linux and mobile.[10]
As for the usability of the engine, it has a dedicated editor MonoDevelop and a GUI interface, and it was implemented with C#, Boo and JavaScript.[2] The later makes Unity 3D compatible for the usage in Web Development to implement Website incorporating rudimentary animations, physically accurate simulations. Furthermore, the game engine is with supports importing and exporting content from/to the best known CAD-Platforms, such as Blender, Maya and 3D Studio Max.[2].
Eleftheria Christopoulou and Stelios Xinogalos in their comparative analysis of game engines ranked Unity 3D as the most usable game engines by including a lot of tutorial, examples and assets. However, their source remain closed for users of their personal version. Furthermore, the community of Unity 3D is very large giving access to more more online support for new users[2]. Eleftheria Christopoulou and Stelios Xinogalos consider Unity 3D a suitable game engine for beginners for having a simpler and refined user interface and providing a lot of resources for them. In addition to that, the hardware requirements aren't too steep and they see the development and export of mobile games with Unity 3D as very straightforward and easy[2].

5 pychrono

pychrono is a Python version of chrono, which is a physics-based modelling and simulation infrastructure implemented in C++. This physics engine can simulate many physical phenomenon, e.g. multibody dynamics, collision detection, and granular flows, etc. In addition to that it proved to be very performant in the filed of vehicle dynamics, robotics, and machine design,[11] which is very comparable to the application we need to implement in this bachelor thesis. To evaluate pychrono, we first must evaluate chrone itself, as it is the underlying infrastructure and pychrono an wrapper to make it usable in python applications.
pychrono is an open source software infrastructure, that can be used to simulate[11]:

- the dynamics of large systems of connected bodies governed by differential-algebraic equations
- controls and other first-order dynamic systems governed by ordinary differential equations
- fluid–solid interaction problems governed, in part, by the Navier-Stokes equations
- the dynamics of deformable bodies governed by partial differential equations

In addition to that, chrono can handle multi-physics problems. chrono, as mentioned above, is written entirely with C++ and is compiled into libraries used by other third party applications. Functions implemented with chrono can be invoked using an API(Application Programming Interface), coming in two flavors C++ and Python(pychrono). chrono is configured and built using CMake, making it compatible with Windows, Linux, and Mac Os, and robustly cross-platform compatible. Thus easing the adaption with minimal specific hardware and software requirements. Furthermore, chrono is designed to leverage parallel and GPU-Computing, and depending on the operating environment, hardware and software, it can use special implemented software for commercial hardware, i.g. CUDA for Nvidia, etc. In terms of usability, the documentation of the main modules of chrono is very extensive, and was generated from the annotations in the C++ Code. Besides, chrono is distributed with multiple ready for use toolkits for ease of adaption by users interested in using it without deep understanding of the underlying implementations. These toolkits are essentially templates encapsulating best-practice solutions, that can reduce modeling time and improved the robustness of the generated simulations[11].

6 Panda 3D

Panda 3d is graphics engine and programming environment developed by disney's VR. It is very flexible and supports a range of applications, such as real-time graphics applications to the development of high-end virtual reality theme park attractions or video games. The name itself is an acronym listing the primary features of Panda 3D, platform-agnostic, networked, display architecture[12]. A fully featured version of this graphics engine include[12]:

- particle systems
- physics and collisions,
- soft- and hard-skin character animation,
- 2D and 3D nonuniform rational B-spline support,
- 2D graphical user interface support,

- multipass rendering, and
- a real-time shader framework.

Pandas was released as an open source project, and was implemented as a platform-agnostic software by extensively using abstraction layers to facilitate portability. This can be achieved by representing services in an abstract form, so that they can be interchangeably used across-platforms. For terms of usability, Panda 3D was written in C++, uses standard programming tools and techniques to provide a low entry point to novice users and decrease the learning curve. Furthermore, Panda 3D takes advantage of the C++ standard library to further advance the portability of this graphics engine. On the other hand, this implementation methods allow users and third party developers to implement Python's Tkinter/Pmw wrappers around the Tcl/Tk API to generate graphical user interfaces[12].

References

[1] J. Hämäläinen, "Game development with unreal engine 4," 2020.

[2] E. Christopoulou and S. Xinogalos, "Overview and comparative analysis of game engines for desktop and mobile devices," 2017.

[3] B. R. Kent, *3D scientific visualization with blender*. Morgan & Claypool Publishers San Rafael, CA, 2015.

[4] M. Lind and A. Skavhaug, "Using the blender game engine for real-time emulation of production devices," *International journal of production research*, vol. 50, no. 22, pp. 6219–6235, 2012.

[5] A. Boeing and T. Bräunl, "Evaluation of real-time physics simulation systems," in *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, 2007, pp. 281–288.

[6] E. Izadi and A. Bezuijen, "Simulating direct shear tests with the bullet physics library: A validation study," *PLOS one*, vol. 13, no. 4, e0195073, 2018.

[7] S.-J. Chung and N. Pollard, "Predictable behavior during contact simulation: A comparison of selected physics engines," *Computer Animation and Virtual Worlds*, vol. 27, no. 3-4, pp. 262–270, 2016.

[8] F. Fazioli, F. Ficuciello, G. A. Fontanelli, B. Siciliano, and L. Villani, "Implementation of a soft-rigid collision detection algorithm in an open-source engine for surgical realistic simulation," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2016, pp. 2204–2208.

[9] D. F. Silva and A. Maciel, "A comparative study of physics engines for modeling soft tissue deformation," in *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*, IEEE, 2012, pp. 1–7.

[10] R. C. Mat, A. R. M. Shariff, A. N. Zulkifli, M. S. M. Rahim, and M. H. Mahayudin, "Using game engine for 3d terrain visualisation of gis data: A review," in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing, vol. 20, 2014, p. 012 037.

[11] A. Tasora, R. Serban, H. Mazhar, *et al.*, "Chrono: Multi-physics simulation engine," *Astrophysics Source Code Library*, ascl–2009, 2020.

[12] M. Goslin and M. R. Mine, "The panda3d graphics engine," *Computer*, vol. 37, no. 10, pp. 112–114, 2004.