How game code and logic can be made more portable between game engines

COMP160 - Software Engineering Essay

1603748

March 29, 2017

Porting video games to different platforms and engines can be a very tedious task as it requires a lot of re-writing and re-compiling source code. In this essay I present a potential solution to porting games without having to re-write any code. It is a way of separating the logic of the game from the rest of the system to make it possible to use the logic on different engines.

Introduction

In this essay I plan to talk about how game logic and code can be more easily ported to other platforms and engines which will result in more cost effective portability of games to multiple different platforms. I will discuss the benefits of portable logic as well as the ways logic can be made portable between different game engines. I will discuss the feasibility of separating out the logic by representing it using ontologies and rules, and by introducing middleware (an events space) between the logic and the game engine [1]. I will also highlight the advantages and disadvantages of using blueprints to determine whether they are still worth using.

1 Standard source code porting

Porting of source code usually consists of analysing the source code to see which parts need to modified and which parts need to be re-written [2]. Having good maintainable and readable code will make this process much easier. As well as re-writing and modifying code sometimes you need also consider any architectural changes required in order to make the porting process easier [2]. In addition to to re-writing code and re-architecting it is also necessary to re-compile the code, fix any problems that come up, run tests and debug. This whole process of porting can take a long time.

2 Porting using an events space

Games are usually ported by rewriting and recompiling major parts of the game, however it is possible to use the same logic on different game engines by using an events space. The events space is used to separate the logic from the engine. This method of porting logic has been demonstrated using the Torque engine and a bespoke simulation engine. Logic for an accident scenario created as part of a knowledge-gathering exercise for police was serviced to both game engines, thus demonstrating logic portability [1]. The STB (set-top box) game architecture design [3] uses a similar method to port STB games to different STB environments. The game architecture is divided into three subsystems: the game space, the adapters and the STB environment. Just like the events space, the game space contains the game logic and game state which is separate from the rest of the system. "The effect this has on portability is that when migrating to a new STB environment, the elements in the game space (i.e., the game state, object model, and game logic) can stay intact"[3].

3 Benefits of using an events space

The logic of any game is the core of the game, it is what determines pretty much everything about the game other than aesthetics. Game engines require the logic to be its own format, for example when making games in the unreal engine it requires you to use blueprints or format your C++ code in the unreal format which can only be executed in the unreal engine. If the logic was separate from the rest of the system it is possible to port the logic to multiple game engines. This would increase logic re-usability amongst projects, as a person could migrate it to a familiar engine and thus avoid the time required learning a new engine [1]. "Reuse could reduce development costs and improve quality and productivity, as it often does in mainstream Software Engineering" [4]. It would also increase the scalability possibilities for the logic, depending on the future development of game engine capabilities [1]. In mobile applications porting is an essential part of development, particularly in games [5, 6], so having portable logic and code would make that activity significantly easier.

Conclusion

In conclusion, logic and code can be easily made portable to different game engines by using an events space. Although it is possible to use the same the logic on different engines it is not yet something that anyone can do. Only people who have this architecture or know how to make it can use it. So is it still worth using blueprints in the unreal engine? For now, yes, but in the future this new way of porting games may be used in the games industry as it would reduce time and cost of porting games significantly.

References

[1] A. BinSubaih, S. Maddock, and D. Romano, "Game logic portability," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ser. ACE '05. New York, NY, USA: ACM, 2005, pp. 458–461. [Online]. Available: http://doi.acm.org.ezproxy.falmouth.ac.uk/10.1145/1178477.1178580

[2] M. Wilcox, *Porting to the Symbian Platform.* John Wiley and Sons, Incorporated, 9 2009.

[3] J. Huang and G. Chen, "Digtal stb game portability based on mvc pattern," in *2010 Second World Congress on Software Engineering*, vol. 2, Dec 2010, pp. 13–16.

[4] W. Scacchi, "Practices and technologies in computer game software engineering," *IEEE Software*, vol. 34, no. 1, pp. 110–116, Jan 2017.

[5] V. Alves, I. Cardim, H. Vital, P. Sampaio, A. Damasceno, P. Borba, and G. Ramalho, "Comparative analysis of porting strategies in j2me games," in *21st IEEE International Conference on Software Maintenance (ICSM'05)*, Sept 2005, pp. 123–132.

[6] T. Bhowmik, V. Alves, and N. Niu, "Porting mobile games in an aspect-oriented way: An industrial case study," in *2013 IEEE 14th International Conference on Information Reuse Integration (IRI)*, Aug 2013, pp. 458–465.