Software Engineering Essay

COMP160 - Software Engineering Essay

1605240

April 1, 2017

The issue of porting software has been around for a long time. Game developers face the issue of deciding whether to commit to a specific game engine. In this essay I will look at the problems game developers face with porting game eninges, the benefits or porting and potential solutions for porting. I will find out if game developers should stick to using the same engine or if they should port to different engines.

1 Introduction

Software engineers "must possess a wide range of skills and talents." [1] This essay will be reviewing to see whether or not game developer should commit to a specific game engine or toolset when trying to develop games. This all stems from the issue that is software portability. In the field of computing the issue of portability has been around since the late 1950s [2]. Software can be classed as portable if it has the ability to be used on another system with efficient cost and effort [2]. The issue of portability is only being made more complex [3]. Portability is a huge issue for game developers which can lead to a number of problems. For game developers portability is when you move from one engine to another without having to redevelop or modify the game to suit that engine [4]. There are lots of key issues with them trying to port to a different engine which I will look into. However there are benefits of being able to port games too from engine to engine but that comes with a cost.

2 Issues With Portability

There are many issues with portability especially for game developers. The first of these issues is if developers want to port a game from one engine to another there will be usually be a delay or an extra cost [5] which can lead to them wasting time / money and even losing out on money if it takes them long enough. This is bad as the aim of a company making software is generally to make money as well as make a game people enjoy. Although if done correctly it means they can get more sales as more people will be able to play their game [5] and research has shown that if implementations are done in multiple environments a program is likely to be more successful and easier to develop and maintain [5]. I believe this to be a good idea as game engines can become old or even discontinued [6] and if its on multiple systems itd make it easier to continue building it without having to port it.

3 Benefits of Portability

As hardware costs are getting cheaper and software is becoming more expensive to produce [7] portability brings lots of benefits. Probably the biggest benefit of game developers porting from one engine to another, as is mentioned above, is that if more people can buy the game thats been developed it leads to lower software costs [5], which lead to a higher profit margins most of the time. This is great news for them as if they want to they can hire more staff, depending on the success, to help make their game even better.

Customer loyalty plays a large part in the success of a game [5] so if game developers ported their game it would enable their customers to switch platforms and still be able to play their game.

4 Potential Solutions

Although the majority of game engines will allow you to use assets from another engine [8], another pressing issue with several game engines is that when they need to ported its a lot more difficult to do so because the logic has to formatted in their proprietary format [9][4]. This causes problems as the logic is the core of the game and where most of the game is stored [9]. Although this is still a problem it is also a potential solution as researchers have tried to separate it using middleware between the logic and game engine [9]. This approach has worked for smaller projects as when they compared a game they built through traditional methods and a game they built when the logic was separated and tested for thirty minutes the average FPS drop was 11.69 percent [6]. If they continue trying to perfect this method and reduce the fps drop for larger games this would be an easy solution for porting games. However, at the current FPS drop this is not suitable for game developers as that would have a major impact on ported games and even the smallest of changes on performance/storage isnt alright [5]. Although this method would help to make logic more reusuable as you could take it to a game engine of your choosing [9] this gives game developers more freedom or the engine they could use.

A solution which sort of links to the other one is "isolating the system-dependent code into as few modules as possible" [10], if this was done for game engines and linked to the previous potential solution I said where they separate the logic this could be a solution as most of the code could be ported between engines, apart from a few tiny bits which would remain engine specific. This means they would make it so you could easily be able to transfer code and assets etc, to another engine with little hassle.

Using a standard interface for every game engine [4] would help game developers develop their games a lot easier on other engines and it would seem as if they hadn't even changed engine. This in turn would lead to less training needed when switching engines but in the beginning a fair bit of training could be required.

5 Conclusion

To conclude, from my findings I believe that in the current state of game engines I think game developers should stick to using a single game engine. They should only port games if they have the time and the money to spend porting the game. Also until researchers perfect the method of trying to separate the logic from the game engine is up to scratch, this could play a massive part of how many games get ported and could lead to the success of lots of smaller companies as they could potentially have so many more users exposed to their game. Similarly if all game engines had a standard interface this would help port games as you could easily redo what needed to be done in another engine.

References

[1] M. Jazayeri, “The education of a software engineer,” in *Proceedings of the 19th IEEE international conference on Automated software engineering.* IEEE Computer Society, 2004, pp. 18–xxvii.

[2] K. Garen, “Software portability: Weighing options, making choices,” *The CPA Journal*, vol. 77, no. 11, p. 10, 2007.

[3] P. Tanner, “Software portability: still an open issue?” *StandardView*, vol. 4, no. 2, pp. 88–93, 1996.

[4] A. BinSubaih, S. Maddock, and D. Romano, “An architecture for portable serious games,” in *Doctoral Symposium, hosted at the 20th European Conference on Object-Oriented Programming ECOOP*, 2006, pp. 3–7.

[5] J. D. Mooney, “Developing portable software,” in *Information Technology.* Springer, 2004, pp. 55–84.

[6] A. BinSubaih and S. Maddock, “Game portability using a service-oriented approach,” *International Journal of Computer Games Technology*, vol. 2008, p. 3, 2008.

[7] M. S. O. Franz, “Code_generation on_the_fly: A key to portable software,” Ph.D. dissertation, Citeseer, 1994.

[8] A. BinSubaih and S. Maddock, “G-factor portability in game development using game engines,” in *Proceedings of the 3rd International Conference on Games Research and Development*, 2007, pp. 163–170.

[9] 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. ACM, 2005, pp. 458–461.

[10] M. Stern, "Some experience in building portable software," in *Proceedings of the 3rd international conference on Software engineering.* IEEE Press, 1978, pp. 327–332.