How can incremental development affect level production in-engine?

COMP130 - Software Engineering

1706165

March 20, 2018

This essay will explore the impact incremental development can have towards the developers suited in level production in the engine environment. Focusing on some of the main pitfalls that an average game company will experience when using the waterfall method and how incremental development can help ease some of those troubles whilst maximising work capacity and performance out of the game designers.

1 Introduction

By definition, incremental development is [1] "is a method of software development where the product is designed, implemented and tested incrementally (a little more is added each time) until the product is finished". Meaning that both development and maintenance is done simultaneously as the product progresses. However, in a context of level production this would mean adding objects one step at a time and creating scenes, also one step at a time. In this writing piece, I will argue that incremental development provides more feasible and robust game levels over using a waterfall paradigm that instead focuses to be a [2] "less iterative and flexible approach, as progress flows in

largely one direction ("downwards" like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, deployment and maintenance". I will explore this idea by analysing the two methods above with their application to game level production and the designer themselves to see which method is more suitable over the production life-span.

2 Incremental Development and its impact on level production

Following the incremental development pattern allows for more flexibility and time for review as such, when a level is being developed in this iterative method, it is very simple to modify and improve some of the main aspects of the scene. With constant review, the designer can quickly spot the area of the improvement and change it as they please. For example, a group of students following the same iterative cycle have found that [3] "In order to understand their flaws and strong points, students receive a review after completion of each production cycle. Then,they can try to improve their weaknesses and enhance their strengths. This evaluation helps students to experience finding and solving problems in game development, which was difficult in conventional curriculum" Having a direct correlation to learning by fixing some of the main problems found in review, is very helpful in the long run as these students and designers get to learn from their mistakes from the get-go and not when the product is complete and facing complete scrutiny over the tiniest mistakes made in development of the stage itself.[3] "In the end of the process participants (undergraduate students) achieved good results, finding employment and completing their thesis" This experience is crucial especially if the designer is fairly new to the industry, whom with this method will eventually make less and less errors as a designer of levels. In addition, this teaches that making mistakes is fine as long as you iterate on them afterwards, by process of elimination, most issues should eventually disperse after a few cycles and in the long term, the same mistakes are hopefully to never be made again because they are something that the developer has

faced before.

The designers themselves, have a rigorous task of creating an enjoyable level that can be replayed as many times as possible without losing its value with each play. Within an iterative method this means breaking down the stage development into different steps as such, [4] Christian Rubino and John Power suggest using the following formulae: "Plan, Sketch, Layout and Hardware Requirements" before even starting any form of development in-engine. In this particular way of developing a stage, the designer should have an idea of the level that is being built before any assets are even imported which in turn, supports iterative development as any shortcomings will be eliminated in the very first stages of creation. Which will result [4] " in more optimized levels, faster frame rates, smoother playability, and the opportunity to supply more content, leading to a richer game experience". These are staple things to strive for within a level and with an Agile development method, breaking down the seemingly impossible mountain tasks into manageable targets like making the level smoother to play and improving the players gaming experience is the goal every designer should strive for. On the contrary, ignoring the planning or the review stage can lead to breakage of game-flow and will result in an incoherent gaming experience. Similarly, the waterfall method would still apply the planning stage however, the review stage is being visited a lot farther down the project development meaning that although the levels are planned well, the lack of a review stage can still impact the level itself negatively in the long run. Any issues that are faced during the production, have no real time to be fixed as this method can only be used on pre-defined requirements, any changes will result in a malfunction.

3 Waterfall method and its impact on level production

Before the iterative method such as Agile came along, the Waterfall method was common amongst the programmers and the designers as no input from the customer or the investor was needed and as such, it was prevalent and preferred over the other. Fast

forward to today, the systems are very close to the product owner and the investors as [5] [5]"Software systems are so much closer to the user that their voices cannot be ignored; they will reject the system if it doesn't meet their needs." To put this into context of level production, if a level that is specified to be made a certain way and it doesn't fit what the product owner had in mind, it will immediately be discarded as it is not what the owner has demanded. Furthermore, changes must be accounted for as [5]"The modern reality of software development is that change is unavoidable and must therefore be explicitly accommodated in the life cycle." Which means that, the waterfall method falls short in the modern way of creating levels or any type of software as the changes are only made once the product is evaluated and not a second sooner. Keeping a consistent dialogue with the product owner and his/her needs would allow the level to be tailored to exactly what the specifications say and if any mishaps happen, they can instantly be fixed whereas the waterfall method doesn't exactly allow the same flexibility. The main advantage that methods like Agile have over waterfall is that prototyping different versions of the same level is a possibility as the designer can create variations and see which is more fit for purpose as the objective of prototyping is [5] "The objective of prototyping is to explore an idea or technology, or to demonstrate a capability, feature, or interface very different objectives from those described." This provides a lot of breathing room for the designers to have alternatives ready if the level is deemed unfit for purpose which in turn, can be swapped out for one that is, something that waterfall method doesn't do very well. However, the prototypes are very disposable as said by Tom Glib [6] "these prototypes will typically be "throw away", and may see little or no real useful work by real users being done on them" Meaning that although useful for testing purposes, they serve no real value to the actual product. In addition, prototyping saves a lot of time in the long run as [7]"Through the interaction with the prototype, the user can get an actual feel of the system, and better understand the requirements" which in the end leads to better wants and needs that the designer can then execute on as the

common problem being that the requirements aren't specific enough as they lack the technical understanding of the subject, this helps with tackling that particular issue. Although it is common to see prototyping be used within a waterfall development cycle as [7] Mohamed Khalifa states "Researchers who investigated the use of prototyping and the waterfall model within the various phases of the life cycle reported that prototyping was frequently used for a single phase within a phased or waterfall approach" meaning that although prototyping is a principle used within another development method, it can still be applied somewhere else. Having the ability to showcase your work to potential investors is very important as it not only shows your individual capability, it also proves that you can manage time with an iterative development method. In industry this is a crucial skill as managing industry deadlines is very much a requirement. Knowing how to take on a difficult to make level in separate chunks will demonstrate both your professional knowledge and various management skills.

Being stubborn by sticking with the waterfall method would be like a [6] "Greek island house builder who builds white painted stone houses in the tradition to his forefathers, and will not admit to flexible technologies such as the sound-proof sliding wall of the hotel conference facility". Seeing the better solution to developing a better stage and not going through with it because of principles that no longer apply to today's world. Similarly, this applies to incremental level production as the concept is straight forward; develop levels small and iterative steps at a time instead of doing the entire bulk of work at once with no time to review all the changes that the designer has made. There is a term called Phase Planning which states [6] "How much can we accomplish within some critical constraint (budget, deadline, storage, space)?" This brings another topic to mind, it being the mental state of a designer whom is following a waterfall method, he/she must produce nearly complete levels at each production cycle whereas with Agile, each stage is being built on simultaneously which does not only provide a mix of work to complete instead of the blandness of one stage and an eventual creativity block.

With that being said, the developer will have a lot more time to put more thought and structurally sound ideas behind the levels to make them more unique whereas, the other method doesn't allow for such freedom. One's use over the other is highly dependable on the conditions as [7] Mohamed Khalifa stated "More specifically, the extent of use of the software development method was significantly influenced by facilitating conditions" although one could theoretically be better than the other, the ultimate deciding factor is that they both work for different uses. In the case of making high-quality game levels, it is better to split the workload into parts and tackle the parts whilst at 100 percent capacity rather than tackling each level as just another job. This also makes the work more manageable, seeing it as a set of smaller tasks instead of one big task could be a massive motivational booster to developers whom struggle with self-motivation. This is a common issue however, the Agile Principle tackles it in a different way as stated by [8]Craig Larman "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation". This tackles the issue of depressed or lacking in self-confidence developers by facing the issue head-on and exploring different ways out of the situation they are in by the use of SCRUM practices. The developer is forced to communicate any road-blocks or issues that he/she has faced along the way which is useful as it develops your inter-personal skills to withstand problems more comfortably.

4 Conclusion

In conclusion, I have discussed some of the main differences between the two and hopefully persuaded you to think that an iterative development method is incredibly more efficient in it's practicality especially concerning a complex task such as producing large-scale levels that require every detail possible. As [8] Craig Larman also says the main goal of Agile is "Working software is the primary measure of progress". Having a level that is playable yet still being developed is something that all the other members

of the group will appreciate as it advances their workload further i.e. testing new mechanics on the level, making sure the theme works and etc. Finally, having a somewhat developed level can be used as groundwork for improvement that the developer can look at and evaluate as the level is being completed, asking questions such as; what can I do differently next time to improve the look of this level? This extra time being spent on review is very crucial in becoming a fully-fledged professional as it teaches problem solving skills that are very common within the industry.

References

[1] *Wikipedia Definition of Incremental Development.*

[2] *Wikipedia Definition of Waterfall Paradigm.*

[3] *Game Jam based iterative curriculum for game production in Japan*, 2015.

[4] *Level design optimization guidelines for game artists using the epic games: Unreal editor and unreal engine 2*, 2008.

[5] *The Demise of the Waterfall Model Is Imminent*, 2004.

[6] T. Gilb, *Evolutionary Delivery versus the waterfall model*, 1985.

[7] *Drivers for Software Development Method Usage*, 2000.

[8] C. Larman, *Agile and Iterative Development: A Manager's Guide*, 2004.