Spring 4-5-2022

Developing Indie Games With Agile

Camden Obertop
University of Nebraska - Lincoln

Developing Indie Games With Agile

An Undergraduate Honors Thesis

Submitted in Partial fulfillment of

University Honors Program Requirements

University of Nebraska-Lincoln

by Camden Obertop

Bachelor of Science

Software Engineering

School of Computing

April 5, 2022

Faculty Mentor:

Christopher Bohn, PhD, School of Compuuting

Abstract

Agile software development has ushered in major improvements to the development of software in the 21st century. Video game development is a form of development that is unique from other types of software engineering, as it can involve work from artists, musicians, voice actors, and others. This paper explores the question whether agile software development as Scrum is an effective tool for creating video games. Ultimately, it can be seen that agile is a very important asset to game developers.

Key Words: video game development, scrum, agile, programming, modeling, gray boxing

Dedication

I would like to dedicate this thesis first to Dr. Bohn, Dr. Person, and Dr. Garvin for helping foster my passion for software development and setting me up for a successful career after graduating. I would like to also dedicate this thesis to my senior design team - Jaden, Tanner, Matt, Ethan, and Erik - who have made developing a game an unforgettable and invaluable experience.

Developing Indie Games With Agile

Introduction

Many people believe that doing something creative is something that happens sporadically, where artists receive bursts of inspiration to create and act at seemingly random times to accomplish their goals. The truth is quite the opposite–most art requires extensive planning, design, iteration, and development. One of the most recent creative outlets in human history is the creation of video games–an amalgamation of art, sound, and code. Many people labor under the assumption that the development of video games is similar to what I mentioned earlier–ad hoc development to add features whenever inspiration hits. Again, in reality, video games require extensive planning and up-front dedication to ensure that the best possible product is created. The real question is: what is the best way to plan and develop video games? In programming, which is a major part of game development, there are many versions of iterative agile development. Are agile processes such as Scrum valuable for developing video games? Should creators consider agile processes to help plan and scope out non-technical parts of game development? I will explore these questions to see, based on the history of agile and game development, if they are a compatible pair.

Agile Development

Agile development is a concept with software development that is utilized to expedite the process of writing and iterating on code. The goal of agile methods is to help software development organizations to quickly develop and change their products and services (Cao, Mohan, Xu, et al.). Typically, agile processes are put in place for projects ranging in sizes from small to large, as they allow for developers to abstract and delegate responsibility effectively. One of the most popular frameworks for implementing agile in development is known as *Scrum.* (Dingsøyr, Moe). Usually, Scrum involves development by a team known as the *Scrum Team.* To better understand exactly what agile is, one could look at Scrum as an example of an implementation.

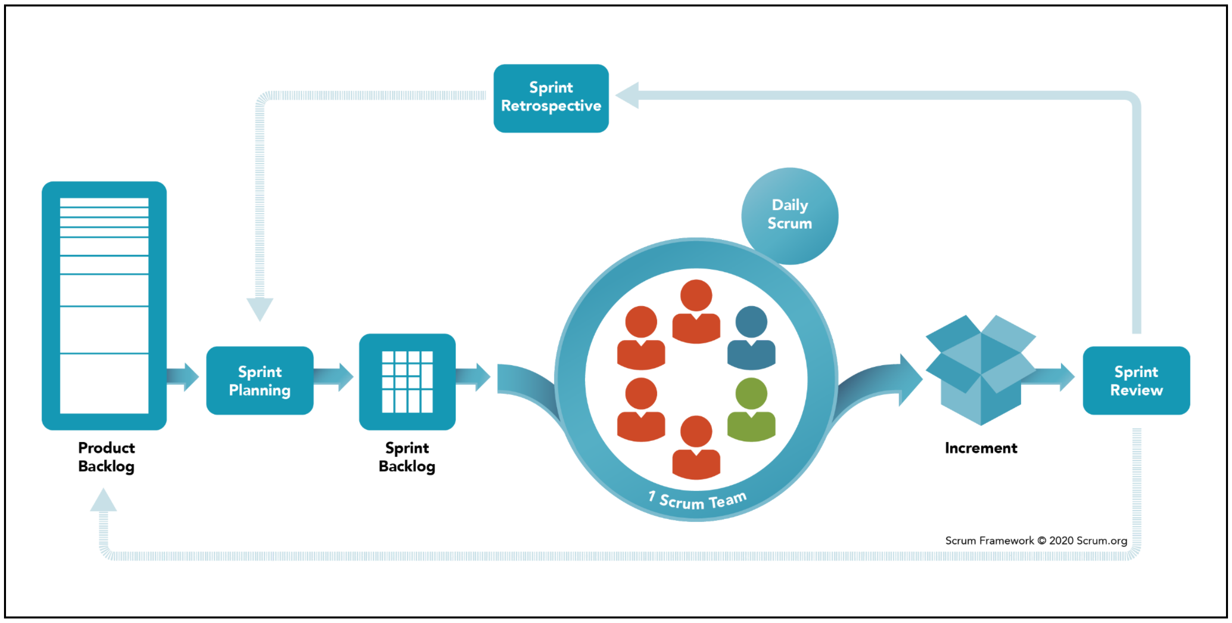


Figure 1: The Scrum cycle according to *Scrum.org*

Scrum is cyclic in nature, so there is not one good starting point, but first we will talk about Sprint Planning. In Sprint Planning, the Scrum Team will split requirements for a project into a set of issues, and will evaluate these issues based on difficulty and time it will take to complete (Scrum.org). These tickets are put into a backlog, and then tackled in intervals of time known as *sprints*. Sprints can be any interval of time, but it is typical to see teams tackle the issues in sprints that are two weeks in length. During this time period, the Scrum Team will meet daily in a *standup* to discuss what they accomplished yesterday, and what they are planning on accomplishing today. When a sprint is finished, typically the updates will be packaged into a new product release, coupled with the Scrum Team performing a *sprint retrospective* (Scrum.org). During these retrospectives, the team will analyze what worked well in the sprint, what didn't work well, and what to do differently moving forward. As can be seen, agile is the implementation of processes in software such that a product can be continually integrated on.

Positives and Negatives of Agile

It is important to consider both the positive and negative aspects of agile. This section will explore how there are more positives, but it is still important to keep in mind that there is no such thing as a perfect development system. One major positive of agile is the ability of stakeholders and customers to provide and receive continuous feedback (Dyba, Dingsoyr). Because of the cyclic nature of iteration, the developers don't have to wait until the end of the development cycle to communicate with the stakeholders. With each sprint and potential release of software, the actual stakeholders can get their hands on an actual version of the application. Many people consider agile as a process that significantly improves productivity (Dyba, Dingsoyr). Another positive is that "agile teams are characterized by self organization and intense collaboration, within and across organizational boundaries." (Cockburn, Highsmith). The main negatives of agile are that it can ultimately take more time from project initiation to project completion, and that sometimes breaking it down into these small sections can cause the team to fall off track. It is harder to consider some big picture ideas when developing with agile.

Experiences with Agile

I have had a few experiences implementing agile, more specifically implementing Scrum as agile. I will discuss a few of these in this section, how I felt about it, and what I would have done differently. I will talk about my 3 consecutive internships at NorthPoint Development as well as when we used it in my class at UNL SOFT 261.

My first internship at NorthPoint Development took place during the summer of 2019. At this time, the software development team was only one person, my supervisor, and me, the intern. My supervisor basically implemented Scrum in the following way: we would have a standup daily, every morning at 8:30am to talk about what I had accomplished the previous day, and what I was working on that day. Sprints were effectively one week long, where we would end each one with a code review. The code review would also act as a sprint retrospective, where we could discuss what I thought was working and what wasn't working. This was my first introduction to agile processes, and I think that it was an effective tool at teaching me how to be an effective software developer.

My second internship at NorthPoint Development took place during the summer of 2020. This internship took place remotely. While the team had grown, things were still fairly ad-hoc as to how projects were finished. Instead of using strict agile, we adopted some components of it such as sprints and retrospectives. Things would have improved this summer if there was more direct management of issues.

My third internship started in summer 2021 and continued into the school year. This summer involved the most in-depth agile processes so far. We utilized the ticket managing software called Jira to create and estimate tickets. Each project has its own label and section of the Jira board which lets us view things together when necessary. Agile has been extremely beneficial at increasing my productivity while working through projects, as it gives me more tangible goals to work through each sprint. Compared to the previous summers, it was extremely beneficial to be using processes that were tried-and-true instead of random ad-hoc processes.

Another time that I implemented agile development was during my SOFT 261 course at UNL. We worked on an open source project called *Ptivi,* a video editor for Linux systems. Implementing Scrum was extremely helpful at breaking up our contribution into smaller pieces that team members could tackle individually. These experiences taught me a lot about Scrum, and as I will mention in future sections they set me up for effectively developing video games.

Video Game Development

Game development is what happens when developers, artists, composers, and more come together to create an interactive video game experience. When making typical software products, there is a common software development life cycle that goes from analysis, to design, to coding, to testing (Ramadan, Widyani). When working on games, the life cycle tends to look completely different. A common game development life cycle usually will have an entire stage for pre-production/design, followed by a continuous cycle of developing and re-evaluating the work that has been done (Ramadan, Widyani). At the end of the game development life cycle will usually come the release of a game. One thing that has changed in recent years is the introduction of early access and beta versions of games, where the testing stage is carried out by future customers instead of an internal testing team (Ramadan, Widyani). Development of video games introduces many unique challenges to the table that might not be present during development of typical softwares. Usually, combining the efforts of developers, artists, musicians, actors, and more will result in a lot of overlap of creativity, and a lot of collisions in goals and desires for the created game (Kanode, Haddad). As mentioned previously, one great challenge in game development is the introduction of the pre-production stage. This can include concept art, coming up with the story, designing the world of the game, designing the mechanics of the game, and so much more (Kanode, Haddad). Solutions to these problems will be discussed in a future section.

History of Game Development

Game development is a relatively new form of art. Video games have only existed for about 80 years. “The first recognized example of a game machine was unveiled by Dr. Edward Uhler Condon at the New York World’s Fair in 1940. The game, based on the ancient mathematical game of Nim, was played by about 50,000 people during the six months it was on display, with the computer reportedly winning more than 90 percent of the games” (Chikhani). Very early on, developing games required much more knowledge in mechanical and electrical engineering than it does in modern times. Developers would have to work at the hardware level frequently to display what they wanted to display. At the time, they would have been working more so with mechanical and electrical engineering standards for development. Later in the 20th century, games started to get developed for arcade cabinets and personal computers, such as the Commodore 64 and the Apple II (Wallach). At the very end of the 20th century, game development must have been facing some of the challenges that other companies were facing with development - the *waterfall* development methodology. This methodology involves going through a project sequentially - starting with some requirements, working through development, ending with testing and delivering to the end users. This methodology was found to take an absurd amount of time for many companies - an average of 3 years spent on a project being the working figure (Varhol).

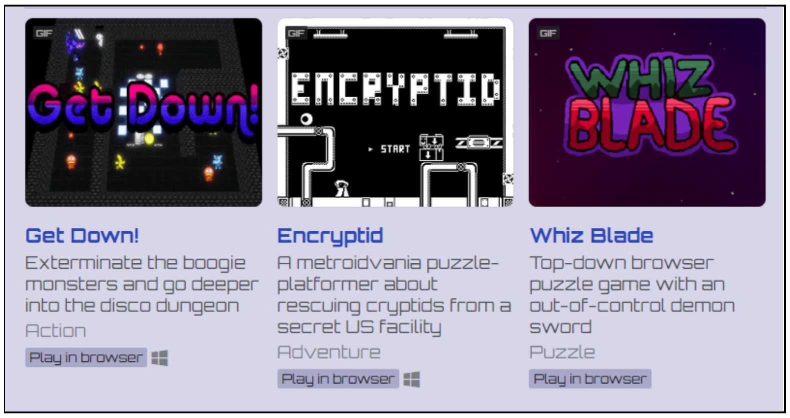
Modern Game Development with Agile

In the modern world of video game development, many teams employ agile, using Scrum specifically. Issues are created in the form of stories, typically considering what the player would be able to do at a certain point in a game. This could range from a player wanting to stay oriented when colliding with bushes to having a clear HUD and seeing what all of their current options are (Cohn). While normal development might have user stories with how they interface with an open, the parallel version here is what players want to be able to do. Some game developers with separate teams will implement separate instances of Scrum running in parallel. There could be a recurring meeting where the programming team, audio team, and art team meetup to discuss progress (Cohn). There are many who think that agile is a great asset to game development. Agile works so well with game development specifically because of the access it can give to customers, and the feedback that developers can get from those customers (Starloop Studios). One big problem that can be tackled with agile is the pre-production/design stage of video games. While programming can be challenging, coming up with a fun, unique world with interesting mechanics can be very tricky for many developers. These tasks can be incorporated into a custom version of Scrum as agile where people can tackle design issues with estimated story values. Many video games have one major release, with potentially some pre-releases/beta versions before the main release and several updates/patches after the release. With Scrum, the iteration can still be performed before release and after release (Eden). Some sprints might become more difficult closer to the scheduled release, but that is a problem that game developers have to deal with.

Experiences with Agile Game Development

There have been a few instances where I have applied agile processes to game development. The two that I will explore in this section are the general use of mini-sprints during game jams, and the scrum setup that I have been using this year in senior design while working on our game *Neural Nest*.

Over the past two years, I have participated in several game jams with my friends. A game jam is a competition where you will receive a theme, and then have a set of hours to create a game from scratch. The time allotted can range from 24 hours to 72, at least in the competitions that I have participated in.



Get Down!
Exterminate the boogie monsters and go deeper into the disco dungeon
Action

Encryptid
A metroidvania puzzle-platformer about rescuing cryptids from a secret US facility
Adventure

Whiz Blade
Top-down browser puzzle game with an out-of-control demon sword
Puzzle

Figure 2: Some games I helped work on during game jams

During these game jams, we decided that we would split up the tasks required into issues, and perform a mini sprint over the course of the jam. For example, if it was a 72 hour game jam, we would divide up stories among the teammates, have a standup every 12 or so hours, and re-evaluate as we were going constantly. We also used this format while designing components of the game. Using the small version of agile helped our teams focus on the issues we knew we needed to complete. The main place where I have used agile in game development is in the production of a game the same group of friends is working on this year - *Neural Nest*.

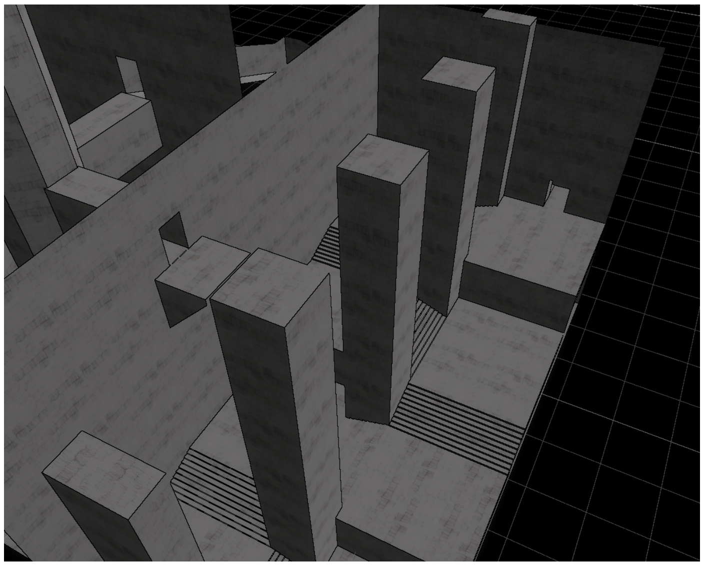


Figure 3: Screenshot from the development of *Neural Nest*

We have set this up with much more traditional Scrum standards - two week sprints for accomplishing our tasks, standups at every team time, and retrospectives at distinct intervals. Scrum has been extremely useful for helping us delegate responsibilities and tasks that aren't even just related to programming. For example, I have been working a lot on the modeling of the "gray boxed" versions of levels. What this entails is designing the structure of a level with gray boxes so that it can at the very least be played in. While this isn't something that involves editing any code, it does involve using the game development software to position game objects and cubes in particular ways. Having an issue in our backlog is helpful for estimating the difficulty of what I am designing and giving me information to work off of. Another non-programming task that is delegated as issues is the creation of 3D enemy models and applying textures. This is done in other softwares, but it is still helpful to have the ticket artifacts to manage the development of the game. One aspect that was giving us a lot of trouble early on was designing the enemies, the main character, and writing down all of the aspects of the game that would inform how we carry out coding tasks.

Overall, my experience with using agile and Scrum and agile while doing game development has proven how valuable these are as tools not just for creating code artifacts but for every aspect of a video game.

Conclusion and Looking Ahead

The questions that I set out to answer in this paper were as follows: Are agile processes such as Scrum valuable for developing video games? Should creators consider agile processes to help plan and scope out non-technical parts of game development? After looking at several other developers' opinions on the subject and weighing my own values, I believe that Scrum and even just agile itself are invaluable to the process of developing video games. The cycle of continuous iteration and feedback is extremely helpful for a process that requires the continuous integration agile provides. Many developers will use agile mainly for the developing and the creation of the game, but it can be seen that agile can also be a helpful framework for accomplishing pre-production tasks and designing components of a video game. Overall, video game development is a form of software engineering that goes so well with agile it should be considered for any game development project.

Bibliography

1. A. Cockburn and J. Highsmith, "Agile software development, the people factor," in Computer, vol. 34, no. 11, pp. 131-133, Nov. 2001, doi: 10.1109/2.963450.
2. Cao, L., Mohan, K., Xu, P. et al. A framework for adapting agile development methodologies. Eur J Inf Syst 18, 332–343 (2009). https://doi.org/10.1057/ejis.2009.26
3. Chikhani, Riad. "The History Of Gaming: An Evolving Community." TechCrunch, https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/.
4. Cohn, Mike. "Agile and Scrum for Video Game Development." Mountain Goat Software, https://www.mountaingoatsoftware.com/presentations/agile-and-scrum-for-video-game-development.
5. C. M. Kanode and H. M. Haddad, "Software Engineering Challenges in Game Development," 2009 Sixth International Conference on Information Technology: New Generations, 2009, pp. 260-265, doi: 10.1109/ITNG.2009.74.
6. Dingsøyr T., Moe N.B. (2014) Towards Principles of Large-Scale Agile Development. In: Dingsøyr T., Moe N.B., Tonelli R., Counsell S., Gencel C., Petersen K. (eds) Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation. XP 2014. Lecture Notes in Business Information Processing, vol 199. Springer, Cham. https://doi.org/10.1007/978-3-319-14358-3_1
7. Eden, Martin. "Agile In Game Development." Melior Games, https://meliorgames.com/game-development/agile-in-game-development/#:~:text=In%20a%20nutshell%2C%20Agile%20in,which%20are%20normally%20called%20features.

8. R. Ramadan and Y. Widyani, "Game development life cycle guidelines," 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS), 2013, pp. 95-100, doi: 10.1109/ICACSIS.2013.6761558.
9. Scrum.org, “What Is Scrum?” https://www.scrum.org/resources/what-is-scrum.
10. Studios, Starloop. “Best Agile Practices in Game Development.” Starloop Studios, https://starloopstudios.com/best-agile-practices-in-game-development/.
11. T. Dyba and T. Dingsoyr, "What Do We Know about Agile Software Development?," in IEEE Software, vol. 26, no. 5, pp. 6-9, Sept.-Oct. 2009, doi: 10.1109/MS.2009.145.
12. Varhol, Peter. “To Agility and beyond: The History—and Legacy—of Agile Development.” TechBeacon, https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development.
13. Wallach, Omri. “The History of the Gaming Industry in One Chart.” Weforum, https://www.weforum.org/agenda/2020/11/gaming-games-consels-xbox-play-station-fun/.