Pipeline Analysis & Results

Jake O'Connor

UAT MS509

Pipeline Project: Part 5

Overview

Introduction

This report will document the development pipeline of the UAT undergraduate game studio project team for Victorian Motocross, as well as how that team's pipeline compares to observations and experiences of the pipelines used by professional game development teams. Victorian Motocross is a prototype for a Victorian-era themed racing game and the team developing it is incredibly small, with only a single team member working in each domain.

Structure

This report will be separated into multiple sections intended to ease comprehension and readability. The first section will include a background on the resources and experiences used to develop the ideas contained herein. The second section will establish what I will call the *Four Pipeline Stages* for the remainder of the document, as a foundation to discuss game development pipelines in the abstract. The following four sections will each detail one of the distinct pipeline stages, what they entail, how they are commonly implemented within the game development industry, how the Victorian Motocross team's process compares to that implementation, and how their process could potentially improve.

Background

The identification of distinct stages within the game development pipeline as a whole, as well as the recommendations outlined herein for the Victorian Motocross team, are based in multiple different sources. First, my personal experiences in the game development industry across multiple studios of different sizes and teams of different makeups. Second, the personal experiences of multiple coworkers and peers as they reflected on their time within the game development industry. Lastly, both peerreviewed articles and public presentations of game companies' pipelines at conferences such as GDC. This breadth of context and information should serve as a passable foundation to establish commonalities between and best practices within the game development pipelines of different teams.

Four Pipeline Stages

Throughout my experience in the games industry and my research into the experiences of others in the games industry, I believe that I've established that there can be no one true, defined pipeline. Even within one team within one project within one company, pipelines and processes are in a state of constant flux and adjustment, and even the most rigid pipelines bend and flex based on current timelines, staffing, and type of content.

To try and make some general sense of these inherent complexities, I've decided to abstract the development pipeline into four distinct stages. These four stages are logical groupings of steps that are frequently taken either in parallel or in series before moving onto the next stage. The steps within each stage of development can vary wildly by content type and overall company process, but the stages themselves always follow the same flow. By abstracting out varying game development pipelines into these four pipeline stages, we can better discuss improvements to the game development process on the micro and macro level without being mired down by the day-to-day workflow process of team members.

Inception Stage

Description

The first stage of the abstracted pipeline is the *Implementation Stage*. This stage is the portion of the pipeline which serves as the impetus for the pipeline's content. Regardless of a team's or company's makeup, or the type of game being developed, the content, features, and improvements which need to be added to the game all stem from some initial inception step. This initial inception often takes the form of a designer or product owner initiating a feature request but is just as likely to take the form of a regulatory change or some inherent need of the current project’s genre.

Industry Process

Industry process in the *Inception Stage* varies greatly depending on the size of company, the type of product they’re creating, and the company’s culture. Smaller teams working on more creatively driven projects are more likely to be much more open during this stage, accepting good ideas from anyone on the team regardless of role or specialty. Larger, more rigid companies, and those working on third party products with less creative control, often have a more limited scope of team members and organizations which can initiate the pipeline. Teams that work in more of a support role, such as a tools or engine teams, often see the inception of ideas in their pipelines coming from the needs of the teams they support (Creating a Tools Pipeline for Horizon: Zero Dawn, 2019).

VMX Process

Since the Victorian Motocross team is so incredibly small and working on such a strict timeline, their *Inception Stage* process is very open to new ideas from any of the few team members. The majority of this stage was dominated by necessary game mechanics and features inherent to the racing genre, but there were various other ideas during this stage sourced from the team members themselves, such as the unique gameplay actions and taunts.

Recommendations

My sole recommendation for the Victorian Motocross team’s *Inception Stage* was the introduction of documentation to the stage. The majority of ideas generated by the project team were only ever mentioned within one of the two weekly video calls, which were themselves only ever attended by a portion of the team’s members. Documenting these ideas is an integral part of the game development pipeline and is especially important in the undergraduate game studio program since the project teams often shift members from semester to semester. Generating new and unique ideas means little if the ideas are lost as time goes on.

Expected Results

Since these recommendations came so late into the development process, weeks after the team had already established a feature lock on the project, no meaningful results could be gathered for the *Inception Stage* of the Victorian Motocross team. In a project with a larger scope, team, or timeline, I would expect this recommendation of documenting new ideas to be beneficial for a number of reasons. First and foremost is the establishment of an idea backlog, a catalog of potential feature and content inclusions for the game which can be sized and prioritized in order to build a roadmap for the game's development.

Ideation Stage

Description

The *Ideation Stage* of the abstracted game development pipeline is the second stage. This stage takes place after an idea has been generated and selected for inclusion into the game. During this stage, an idea will be taken and molded into the form or forms which can be consumed by the remainder of the pipeline stages. The *Ideation Stage* can vary a great deal from team to team, mostly as a result of differences in the type of idea and the domain of the team whose pipeline it is in. This stage in an engineering pipeline might include steps for identifying industry best practices and establishing success criteria, while within an art pipeline it might include creating concept art and establishing a color palette which fits into the game's overall art direction.

Industry Process

There exist a wide range of ideation steps that exist in the games industry's various development pipelines. At Riot Games' offices, during the creation of a new League of Legends champion, the *Ideation Stage* includes input from many different subject matter experts ranging from engineering to art to lore design, all of which come together in order to ensure that a new champion fits within the game's overall design and can be implemented cleanly with the systems in place (Creating League of Legends Champions: Our Production Framework Revealed, 2019). In my personal work developing tools for artists and designers, the *Ideation Stage* often includes personal interviews with the potential consumers of the tools in order to establish their needs and expectations.

VMX Process

Like the *Inception Stage,* the majority of the Victorian Motocross team's *Ideation Stage* is handled exclusively through twice-weekly video calls with some portion of the team. This process is highly collaborative but not a lot of documentation artifacts are created, aside from JIRA tasks, which limits the ability for the team to reference concrete decision made previously.

Recommendations

My recommendations for the Victorian Motocross team in the *Ideation Stage*, much like in the *Inception Stage,* were centered around the documentation of the output of the ideation process. Ideas generated in the first stage should be documented, and then those ideas which are fleshed out in the second stage should be even more heavily documented.

Expected Results

Since the recommendation to document the *Ideation Stage* output better came so late into the team's development cycle, there was no time to implement any changes. The Victorian Motocross team had entered feature lock well before the recommendations were crafted, which left only low-ideation tasks and content to work on, such as fixing bugs, balancing gameplay, and polishing existing art assets. If the Victorian Motocross project exists beyond the semester, the team adopting a culture of better documentation during the *Ideation Stage* should result in both a clearer understanding of expectations and success criteria for the features and content being created as well as a better grasp on the scope of work required to achieve these tasks.

Implementation Stage

Description

The third stage of the abstracted game development pipeline, the *Implementation Stage,* is the stage in which individual contributors and subject matter experts take the output from the *Ideation Stage* and use it to add features and content to the game. The steps within this stage will vary a great deal depending on the type of task and the domain of those tasked with the implementation, but the output of this stage will always be functional changes to the game being made.

Industry Process

As the *Implementation Stage* varies so wildly by domain, the industry process is hard to pin down. For more engineering-oriented tasks, many companies' process includes the necessary addition of debug tooling and/or the functionality for the new feature to be data-driven (George, 2021). For more art-oriented tasks, the steps within this stage of the development pipeline differ depending on the type of art assets being created, but often involve multiple artists with different specialties (Moy, 2021).

VMX Process

The *Implementation Stage* of the Victorian Motocross pipeline is one of the hazier stages, due to the team's small size and relative isolation. No structured process exists for the implementation of features or content into the game, save for that implementation being tracked through JIRA tasks and that the changes make it into the game's build. There are no expectations of debug tooling or unit test writing, and due to the team's size there are no multi-person chains of implementation.

Recommendations

My only recommendation for the Victorian Motocross team's *Implementation Stage* is a push towards the inclusion and expectation of new code being equipped with automated tests. On a project team with only a single dedicated engineer, automated testing can ensure what limited manpower does exist isn't spent correcting mistakes and fixing unintended changes made by team members. The engine being used by the project team, Unreal Engine, supports various types of automated testing out of the box (Ninh, 2021).

Expected Results

The Victorian Motocross team did not have a chance to enact any of the recommended changes to the *Implementation Stage,* but had they done so I would have expected an overall decrease in the number of bugs introduced into the project, specifically those which are regressions in functionality. Automated testing may seem like a lot of extra scaffolding to build on top of a simple game made by such a small team, and it is, but on a team with only a single dedicated engineer and the potential for anyone to submit script changes due to Unreal Engine's Blueprint visual scripting system, it is even more important for integral features to be heavily tested to ensure their continued functionality.

Validation Stage

Description

The fourth and final stage of the abstract game development pipeline is the *Validation Stage*, which sees various members of the game development team confirm the correct functionality of the content and features implemented during the *Implementation Stage.* This stage is integral to the overall process as it not only ensures that tasks are completed and functional within the build, but that they meet all success criteria outlined during the *Ideation Stage*. This stage varies only slightly between companies and teams of different sizes, the minor differences usually revolving around the number of validation steps in place.

Industry Process

A majority of professional game studios operate within some form of an Agile development framework, which means they share at least one *Validation Stage* step in common: the product owner review, in which the Agile team's lead confirms the completion of the task. Before the product owner's review, the *Validation Stage* varies based on the type of content. Art-oriented tasks often undergo one or more approval steps by the art director(s) or art team as a whole (Moy, 2021). Content and functionality changes often undergo testing in the form of manual quality assurance passes as well as automated unit- and integration-tests (Olan, 2003). Engineering task pipelines frequently include a step for peer code review, in which one or more engineers must review and approve all code changes before they can be included in the project.

VMX Process

The *Validation Stage* process for the Victorian Motocross team only includes a single step aside from the product owner review, which is ad hoc testing done primarily by the task's author. As the team had no dedicated designer until recently, no dedicated testers, and no form of automated testing, little validation was included in the pipeline.

Recommendations

My recommendation for the Victorian Motocross team's *Validation Stage* is the inclusion of automated testing and peer reviews. Automated testing not only of code functionality, but also of content correctness, would go a long way to reduce the number of issues submitted to the project. Peer reviews would be of limited use, due to there being only a single team member in each domain, but the necessity of packaging and unpackaging asset changes for peer review often uncovers issues all on its own.

Expected Results

The Victorian Motocross team did not have the time or resources to add any new steps to the *Validation Stage* of their development pipeline. Had they added automated testing, I would expect the number of simple bugs generated to be reduced, leaving the team members more time and energy to work on complex problems or the implementation of new features. Peer reviews would have been minimally useful due to the team's current composition, but I expect engraining that step into each team member's personal process would have improved the overall submission quality.

Conclusion

What little exists of the established pipeline for the Victorian Motocross team is to be expected for a team of its size and working on the scope of game that it is. The steps common to the processes and pipelines among professional game development teams are generally not present in the pipeline of the Victorian Motocross team, mostly due to the short history of the team and the limited scope of their project not requiring too much explicit process. My recommendations to the team primarily focused on additional documentation with the aim to improve the handling of design decisions and the generation of gameplay content ideas, especially as the project potentially passes onto a different team in the coming semesters.

References

The Animation Pipeline of Mario + Rabbids Kingdom Battle. (2019). YouTube. https://youtu.be/qxLR8qbD5JE.

The Animation Pipeline of Overwatch. (2018). YouTube. https://youtu.be/cr7oO8kDu8g.

Appelman, R. L. (2009). Defining the Development Pipeline for Meaningful Play. In Proceedings of International Simulation and Gaming Association Conference. Singapore.

Colby, R., & Colby, R. S. (2019). Game design documentation: Four perspectives from independent game studios. Communication Design Quarterly Review, 7(3), 5-15. http://sigdoc.acm.org/wp-content/uploads/2019/09/CDQ_7.3_2019.pdf#page=5

Creating a Tools Pipeline for Horizon: Zero Dawn. (2019). YouTube. Creating a Tools Pipeline for Horizon: Zero Dawn.

Creating League of Legends Champions: Our Production Framework Revealed. (2019). YouTube. https://youtu.be/dhJXtPPfKbg.

Free code review tool for helix core. Perforce. (n.d.). https://www.perforce.com/products/helix-swarm.

George, Ryan. (2021, July 16). Senior Software Engineer at Wizards of the Coast, former Software Engineer at NetherRealm Studios. Personal communication [Online chat].

Making the World of Firewatch. (2016). YouTube. https://youtu.be/hTqmk1Zs_1I.

Moy, Kevin. (2021, July 16). Senior Digital Production Artist at Wizards of the Coast, former Senior Environment Artist at Turbine. Personal communication [Online chat].

Narrative and Mission Design in Assassin's Creed III. (2016). YouTube. https://youtu.be/pOh8cGFu4Os.

Ninh, A. (2021, April 7). An introduction to automated testing for an Unreal engine project. Kobiton. https://kobiton.com/automation-testing/an-introduction-to-automated-testing-for-an-unreal-engine-project/.

Olan, M. (2003). Unit testing: test early, test often. Journal of Computing Sciences in Colleges, 19(2), 319-328. https://www.researchgate.net/profile/Michael-Olan/publication/255673967_Unit_testing_Test_early_test_often/links/5581783608aea3d7096ff00c/Unit-testing-Test-early-test-often.pdf

Truong, K. N., Hayes, G. R., & Abowd, G. D. (2006, June). Storyboarding: an empirical determination of best practices and effective guidelines. In Proceedings of the 6th conference on Designing Interactive systems (pp. 12-21). https://ellieharmon.com/wp-content/uploads/2014-02-27-Truong-Storyboarding.pdf

Up Sh*t Creek: Pro Tips for Managing the Unmanageable Project. (2019). YouTube. https://youtu.be/dNlEZZlmIcw.