Pipeline Recommendations

Jake O'Connor

UAT MS509

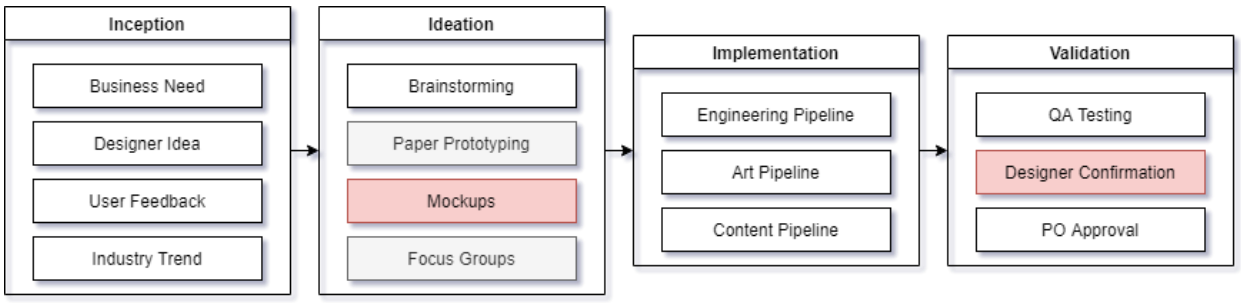
Pipeline Project: Part 3

Overview

This document aims to outline my personal recommendations for improvements to the *Victorian Motocross* team's current pipelines and processes. These recommendations come partially from compiling best practices from reports and conference presentations, and partially from a decade of personal industry experience as a game developer. As the *Victorian Motocross* team is so small and on such a short timeline, my recommendations will be specific to those things which can be reasonably implemented by the team without causing undue overhead.

This document is broken down into four sections, each which corresponds to a rough responsibility on the game team: features, content, engineering, and art. Within each section, the pipeline for that section is further broken down into four broad steps: inception, ideation, implementation, and validation. Each pipeline and step will be briefly explained, and recommendation for the *Victorian Motocross* team will be highlighted and explained in further detail.

Feature Pipeline



In broad terms, the feature pipeline is the flow of changes to the game project which fundamentally alter the mechanics and require multiple teams or disciplines to work together. A feature example from my personal experience on Magic the Gathering: Arena would be the addition of the Momir gameplay mode into the game, which required multiple teams from art to design to engineering

and altered the way in which the game was played. An example of a feature within *Victorian Motocross* would be the addition of characters or bikes which behaved differently or had different moves available.

Inception

The inception step of the feature pipeline is the broad category of instigators of change at the feature level. These instigators are most often business need, such as the feature of adding a store to the game. They can also come from a variety of sources both within the game team itself and from outside, such as changes in law or policy that require certain needs to be met. For the *Victorian Motocross* team, which is still small and not within some larger company, there are no necessary changes in my estimation. The current inception step of *Victorian Motocross* in mostly instigated by the team itself and the requirements of the game genre.

Ideation

The ideation step of the feature pipeline is the broad category of actions that game studios commonly take after the inception and before the implementation of a feature. This commonly includes brainstorming and focus grouping either inside or outside of the company, as well as prototyping and mockups. One example of this step in the pipeline is Riot Games' ideation on character design, both graphically and mechanically, as detailed in their GDC talk from 2019 (*Creating League of Legends Champions: Our Production Framework Revealed*, 2019). *Victorian Motocross* already does a good job of ideation through brainstorming, specifically during weekly video calls, but I believe the feature pipeline for the team would be improved if features received mockups and documentation before progressing to implementation. Mockups and documentation of expectations will make it much easier for the team to maintain a consistent, shared vision as they break into disciplines to begin work.

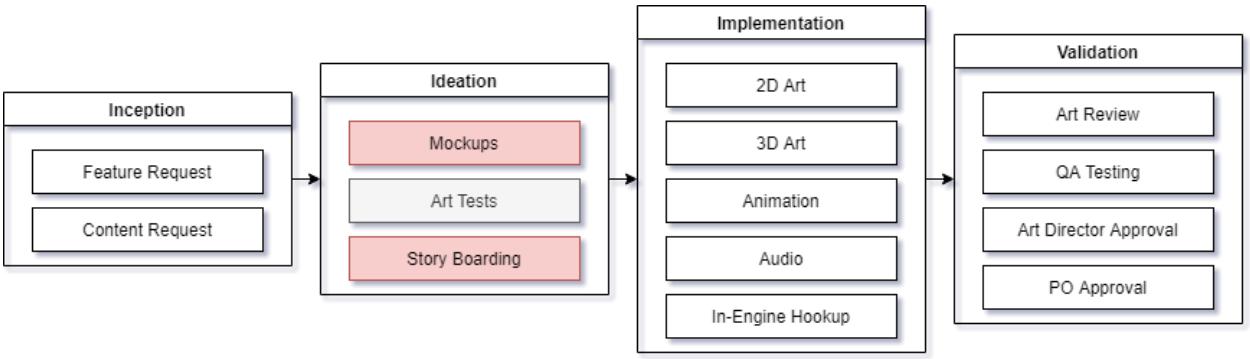
Implementation

The implementation step of the feature pipeline does not contain any actions within itself, but instead relies on one or more of the other pipelines within the game team in order to accomplish its goals. Depending on the type of feature being created or modified, any or all the disciplines within the game team might be involved within the implementation of a feature. Using the previous example of League of Legends, after the design of a new champion has been decided, implementation of new code features would go through the engineering pipeline, the champion's character model, animations, and other aesthetic aspects would go through the art pipeline, and the champion's backstory and game balance would go through the content pipeline.

Validation

The validation step of the feature pipeline is the category of actions which the game team uses to confirm that the feature implemented in the previous step works as intended and satisfies the expectations of both the designer of the feature and the product owner for the team. For *Victorian Motocross*, which relies largely on team input to generate feature ideas, I would recommend the addition of a designer confirmation step for features which originate from the team. Not only should this result in more ownership over features within the game but should ensure that features are implemented more in line with specifications and reduce overall churn.

Art Pipeline



In broad terms, the art pipeline is the flow of changes to the game project's aesthetic appeal, be they through visuals, animations, sound effects, or music. For example, on *Victorian Motocross*, the addition of character voice over effects is something that would fall into the art pipeline.

Inception

The inception step of the art pipeline is the broad category of events that instigate the creation or modification of art assets within the game. These instigators are almost always either the feature pipeline or the content pipeline. Less often, the art pipeline will be instigated by external teams such as marketing.

Ideation

The ideation step of the art pipeline is the broad category of actions that are commonly take before the actual creation of the final art assets themselves. These actions include creating rough mockups and color palettes, art tests, and storyboarding, among many others. The art ideation step helps to ensure that significant talent and time are not wasted implementing assets which do not satisfy the original requirements or are otherwise unusable in the product. For *Victorian Motocross,* my recommendation for this step of the art pipeline is to make use of more mockups and storyboarding. While it does take extra time up front, and the team has precious few artists, taking that extra time will

ensure that what limited resources are available are devoted to work which will meet the needs of the project. Additional documentation and mockups will also increase the likelihood that future members of the team will have access to information necessary to maintaining the art direction of the project.

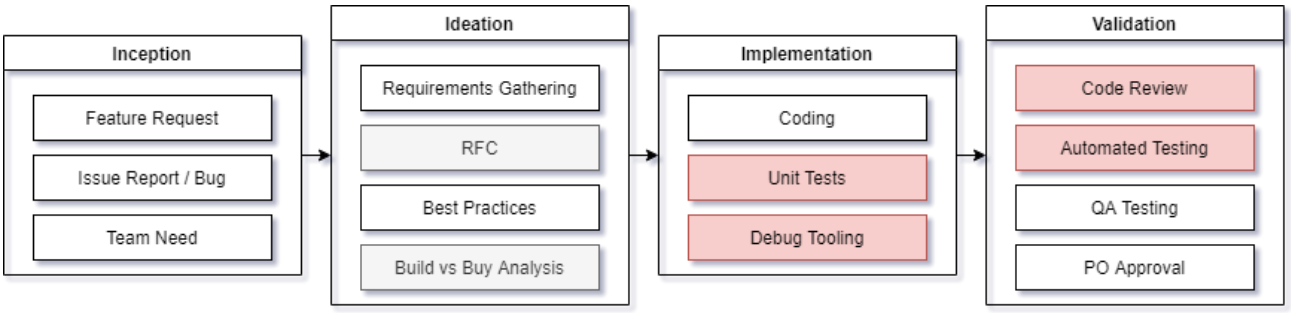
Implementation

The implementation step of the art pipeline is the broad category of actions undertaken to create the art assets themselves. As there are dozens of different types of art that each require unique skills and understanding, this step of the pipeline could take many different forms depending on the needs of the design. On larger teams with more diverse skills, this step may involve multiple artists within different disciplines. One extremely important action within this step of the pipeline is the in-engine hookup of the newly created art assets. Without seeing and testing the assets in game, it is never possible to truly know when ‘done’ is.

Validation

The validation step of the art pipeline is the step in which the newly created or modified art assets are validated, tested, and approved. This generally includes testing by QA, art review by peers within the team, and approval by both the art director and product owner. This validation step can be complicated and slow in larger, more rigidly structured teams, but for *Victorian Motocross* the current process of simply presenting the completed work to the team and product owner is likely sufficient.

Engineering Pipeline



In broad terms, the engineering pipeline is the flow of changes to the code for a game projects client, servers, or tools. This pipeline covers anything from the addition of code hooks for artists to use to link assets, to new functionality in the core gameplay loop. On *Victorian Motocross*, an example of the engineering pipeline is the addition of rubber-banding AI-controlled non-player characters.

Inception

The inception step of the engineering pipeline is the broad category of instigators for a code change within the project. Like within the art pipeline, the primary instigator of the engineering pipeline is requests from the feature pipeline. In addition to feature requests, the engineering pipeline is also often instigated from bug reports and other need within the team for tooling. Regardless of instigator, engineering tasks pass through the inception step of the pipeline into the ideation step.

Ideation

The ideation step of the engineering pipeline is the broad category of actions which take place after a task has been generated but before implementation begins. This step can vary wildly between studios, as well as between different types of engineering tasks. The most common actions during this step are requirements gathering, where the engineering team determines what is required in order to implement the requested task, and best practices analysis, where the engineering team consults with one another and outside resources for solutions to common problems before reinventing the wheel.

Less common actions include RFCs (request for comments) and performing a build-buy analysis, both of which are more likely to appear in very large teams and companies. For the scope of *Victorian Motocross* and the size of its team, it is appropriate for the engineering team to only perform requirements gathering and consult best practices before advancing to the implementation stage.

Implementation

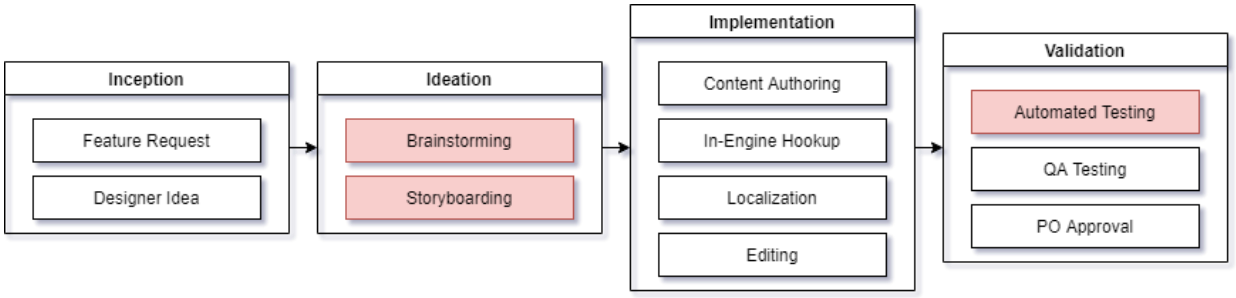
The implementation step of the engineering pipeline is the step in which code is written in order to satisfy the requirements of the task. This is the step in the engineering pipeline which needs the most modifications for *Victorian Motocross.* Two actions within this step, the creation of unit tests and the creation of debug tooling, would be good additions to the *Victorian Motocross* team's engineering pipeline, especially with such a small team. Unit testing is an industry standard way to ensure that code functions correctly on a modular level so that bugs and unintended changes are caught earlier in the process. Unreal Engine, like most modern game engines and IDEs, supports integrated unit testing within the engine itself and facilitates the creation of tests with both C++ and Blueprints (Ninh, 2021). Unit tests not only serve as a method of guaranteeing functionality but can also act as both documentation and defined success criteria for future engineers interacting with the code.

Validation

The validation step of the engineering pipeline is the step in which newly added or modified code is tested and approved. This validation often includes peer review, automated testing, manual QA testing, and finally confirmation by the product owner that the task was satisfied. For the case of the *Victorian Motocross* team, my recommendation is that both code reviews and automated testing are instituted as standard practice. Unreal Engine supports both unit tests and integration tests, which can be run automatically from within the engine and can test both core code functionally as well as full 3D simulation. These tests should always be run as part of the validation step of the engineering pipeline.

Since the *Victorian Motocross* team uses Perforce Helix for their source control, the addition of peer code review is also simple. Perforce supports Helix Swarm, which is a widely used code review tool within the games industry and should be used by the engineering team of *Victorian Motocross* to add an additional layer of validation atop their current process (*Free code review tool for helix core*).

Content Pipeline



In broad terms, the content pipeline is much like the feature pipeline. Unlike the feature pipeline though, the content pipeline focuses on changes to the game project which do not alter the foundational rules of the game or generally require multiple teams or disciplines. An example of a task within the content pipeline from the *Victorian Motocross* team would be the modification of the riders' physics by tweaking the exposed physics values in the engine.

Inception

The inception step of the content pipeline is the broad category of events that can instigate a change to content within the game. Generally, these instigators are larger feature requests from the feature pipeline. In game projects with more content driving designs, a major instigator of the content pipeline is the designers and content authors themselves.

Ideation

The ideation step of the content pipeline is the broad category of actions which take place in order to turn an idea into something that can be implemented. The ideation step of most companies in the industry involves at the very least brainstorming and documenting, and depending on the type of idea may include storyboarding of the full concept. For *Victorian Motocross* the most important recommendation for this step is to document the results and stages of the brainstorming process when ideating on tasks. This will help the rest of the team, as well as future members of the team, understand the process that was taken to arrive at a final design as well as ensure that the final design and its expected behavior are documented for testing purposes. I also recommend the use of storyboarding during the content ideation step, as it is one of the best ways to convey complex ideas and the relationships between actions (Truong, Hayes, & Abowd, 2006).

Implementation

The implementation step of the content pipeline is the broad category of actions taken in order to make a content change functional. This will depend greatly on the type of content change, but generally includes content authoring, localization, and editing of the content. In-engine hookup is also an integral part of the implementation step of the content pipeline, as no task is finished until it is functional.

Validation

The validation step of the content pipeline contains the actions which are performed in order to validate and approve the completion of the content change within the project. This generally always includes QA testing and product owner approval, and for most industry studios includes the use of automated testing. Much like was explained in the engineering pipeline, Unreal Engine supports both

unit tests and integration tests within its test harness, so the *Victorian Motocross* team should ensure there are ample tests in place to validate any newly added or modified content.

References

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

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.

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. Creating League of Legends Champions: Our Production Framework Revealed.

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].

The Motion Capture Pipeline of The Last of Us. (2016). YouTube. https://youtu.be/2GoDlM1Z7BU.

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. Up Sh*t Creek: Pro Tips for Managing the Unmanageable Project.