Enhancing Collaboration Between 3D Artists and Programmers in Game Development with Agile Practices

Lucas Rister Machado[1], **Mateus Beck Rutzig**[2], **Lisandra Manzoni Fontoura**[3]

[1]Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal de Santa Maria (UFSM)
Caixa Postal 1.000 – 97105-900 – Santa Maria – RS – Brasil

[2]Departamento de Eletrônica e Computação (DELC)
Universidade Federal de Santa Maria (UFSM) – Santa Maria, RS – Brasil

[3]Departamento de Computação Aplicada – DCOM
Universidade Federal de Santa Maria (UFSM) – Santa Maria, RS – Brasil

lucas.rister@ecomp.ufsm.br, {mateus,lisandra}@inf.ufsm.br

Abstract. *Game development differs from traditional software flow since it should align stakeholders' needs with 3D artist and programmer teams. Stakeholders' requirements, such as high-quality 3D models and new functionalities, conflict with the constraints that programmers should deal with, such as hardware graphics processing limitations and resource optimization, which can increase project costs and time to market. In this work, we propose the employment of a set of agile practices to facilitate the development of 3D assets that, in addition to satisfying the demands of stakeholders, also address the requirements and constraints of the 3D artist and programmer team. Results show that the proposed practices have reduced the time to respond to new requests or modification of assets, avoiding redoing tasks and wasting developer time.*

Keywords. *3D assets, Scrum, Game development, Agile Practices, Case Study.*

1. Introduction

The global gaming market has increased enormously; projections indicate that by 2025, it will achieve a revenue of $268.8 billion (Clement). As the market increases, the game development complexity increases as well, dealing with development issues such as lack of documentation, crunch time, and rework (Politowski et al. 2021)(Petrillo et al. 2008). These issues affect the final product's quality and result in overtime and excess resources during development.

While game development differs from traditional software development, many companies employ software engineering practices to support game development (Aleem et al. 2016). However, most of these practices primarily focus on managing the programming team's activities (McKenzie et al. 2019), often overlooking a crucial team for the success of game development: the artistic team, especially 3D artists.

The development of assets by 3D artists involves a continuous and well-defined workflow (Hristov e Kinaneva 2021). Figure 1 illustrates the pipeline for producing a 3D model, in which the flow of activities developed by the 3D artist occurs sequentially from left to right. Inside the dashed box, the ideal 3D modeling pipeline begins in the

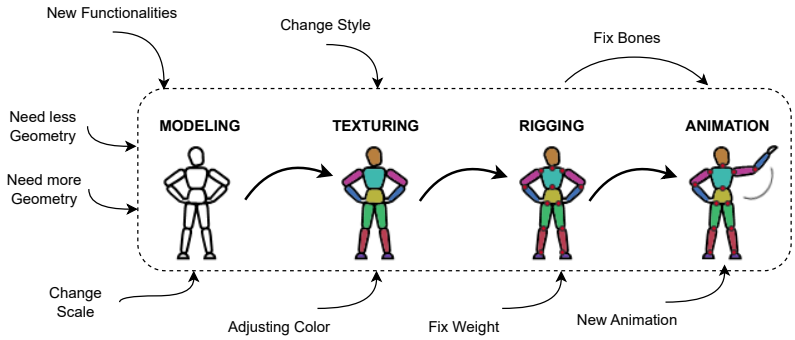

Figura 1. Pipeline expectations for 3D modeling.

"Modeling"stage, where stakeholders describe all physical aspects of the model. The subsequent stage, known as "Texturing", employs specialized software (e.g., Substance 3D Painter) to paint over the faces of the 3D model, adding visual details such as colors, wear, and dirt. The third stage, called "Rigging", involves adding a structure known as an armature to enable control over joints in the model. The final stage, "Animation", adds animations to the 3D model. Once this step is complete, the asset is sent to the programming team for integration into the game development process, where all the necessary programming is carried out to incorporate the asset into the application's functionalities.

However, this 3D development pipeline does not proceed seamlessly in this manner. New requirements typically emerge after the pre-production stage, and issues related to texture, modeling, and animation are often identified only when the asset is finalized and delivered to the programming team, as shown by requests outside the dashed box. These are just a few examples of problems encountered throughout the development of an asset that needs rework in several stages, leading to time and resource wastage.

According to (Keith 2020), a typical example of debt in game development is in character creation. Characters are often designed, modeled, rigged, and animated before the team fully understands what they want them to do or their budget. It occurs due to schedule pressure or a plan that optimizes allocation relative to the delivery of work assets. (Keith 2020) concludes that by the time the team figures out what they want the characters to do, they realize that the rigging and animation requirements have changed, and instead of having to retool and reanimate one character, they have to do it for 20 times.

Regardless of the development phase, the demand for changes emphasizes the importance of agile practices as described in the Agile Manifesto: "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage" (Beck et al. 2001). Additionally, the collaboration between 3D artists and the programming team to meet the needs of stakeholders highlights two of the four values of the Agile Manifesto: "Individuals and interactions over processes and tools" and "Customer collaboration over contract negotiation" (Beck et al. 2001).

This article describes the lessons learned throughout the project and proposes a set of practices that help overcome the challenges faced by 3D artists when developing assets incrementally. These practices are designed to meet the needs of programmers

and stakeholders, improve collaboration, avoid crunch time, enhance the quality of asset development, and reduce wasted resources and rework (critical for changes to 3D models). Additionally, they aim to assist future research in optimizing asset development work.

This article is organized as follows: Section 2 presents essential concepts for understanding this work and related works. Section 3 outlines the methodology of this research. Section 4 discusses the results obtained. Section 5 delves into a discussion about the results. Finally, Section 6 presents the final considerations of the research and suggestions for future work.

2. Background and Related Work

As depicted in Figure 1, the workflow for creating an asset involves a series of sequential and well-defined steps, each of which will be detailed below.

The pre-production stage, crucial before 3D modeling begins, involves gathering requirements to align assets with stakeholder needs. However, hardware limitations may hinder fulfilling all requests. Addressing the language barrier among stakeholders, 3D artists, and programmers is vital for collaboration. The different languages developers, designers, and artists speaking create communication challenges (Keith 2020). This stage, often led by team leaders and stakeholders, can result in vague descriptions of 3D artists' tasks, leading to uncertainties in development.

At the Modeling stage, 3D artists utilize specialized software to create virtual representations of real objects using polygons. They employ various techniques such as extrusions, cuts, and rotations to craft objects. This stage offers flexibility between low poly and high poly development. High Poly Models, with their intricate detail, are suitable for rendering images and game art, while Low Poly Models prioritize efficiency for real-time processing. Furthermore, variations of the same model can be used to implement Level of Detail (LOD) techniques, managing processing demands by adjusting model detail based on proximity. Still, at this stage, it is necessary to generate UV mapping, which is essential for texture application. This process involves projecting object faces onto axes to minimize distortions. As models grow in complexity, UV mapping becomes more challenging.

The Texturing stage involves the use of specialized software to paint 3D model faces, where the model is initially imported from modeling software. Within the painting software, various brushes and materials allow artists to paint the model's faces in a three-dimensional view, adding visual details that vary subjectively among artists—ranging from quick for simple models to intricate and time-consuming for complex ones with many parts, colors, and details. Following painting, the software exports texture maps, which are then re-imported into the modeling software to apply colors and effects. Commonly used maps include Albedo for pure colors, Metallic for reflections and shine, Roughness for surface texture, Normal map for high relief effects, and Ambient Occlusion for ambient shadow influences.

Another stage is Rigging, primarily for organic models such as characters and animals that require specific mesh deformation, like joints. In this stage, "bones"are added inside to create an "armature", with these bones affecting nearby polygons so that when the bones move, the polygons follow. This rig is crucial for animating organic models.

Finally, the last stage of the asset production process is Animation, which occurs before the asset is sent to the programming team. In this step, all keyframes of the model animation are mapped, similar to video editing software. This approach involves storing the scale, rotation, and displacement information of parts of the model at specific time points, and then recording the modifications of these parts at subsequent time points. The transition between recorded time intervals animates the modified parts of the object. This is a straightforward but widely used animation concept.

One of the strategies for developing the pipeline stages described above is the use of practices that help manage the team's work. Agile frameworks, such as Scrum, present a methodology that counters the rigidity of traditional models like the Waterfall cascade approach. Scrum, a widely-used agile process framework, manages work on complex projects through events like Sprint, Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective, involving roles such as Product Owner, Scrum Master, and the Development Team, along with artifacts like Product Backlog, Sprint Backlog, and the application increment. This iterative approach gradually adds new features, fostering collaborative development and enhancing the final product's value (Schwaber e Sutherland 2017).

Gomes et al. (Gomes Filho et al. 2018) present a case study on the application of Scrum and Lean Kanban to support the development of animations for a 3D animation production company, demonstrating how adaptation to the development process reduced animation development time, increased transparency, and enhanced team engagement.

McKenzie et al. (McKenzie et al. 2021) surveyed eight game development studios in New Zealand, discussing the applications and limitations of Agile frameworks like Scrum and Kanban in pre-production, production, and post-production stages. The authors emphasized that Agile frameworks, such as Scrum and Kanban, must often be adapted from their conventional use to meet the specific needs of different pipelines within game development phases.

Maxim and Guzdial (Maxim e Ridgway 2007) highlighted challenges between artists and programmers in game development courses, emphasizing software engineering principles. The study emphasizes the application of software engineering principles to game development. By treating the students in the course as an interdisciplinary team, the authors highlight, among the lessons learned, the challenges encountered between the artists and the team of programmers.

Petrillo and Jenkins (Petrillo et al. 2008) studied the primary challenges in the development of 710 electronic games, finding that these challenges mirror those of traditional software development. They highlighted issues such as unrealistic scope, difficulties with defining requirements, and common problems like crunch time (reported in 45% of cases) and budgetary excess (reported in 25% of cases).

Musil and Spielhofer (Musil et al. 2010) studied Austrian video game development practices, focusing on the challenges faced by interdisciplinary teams of artists and engineers. They suggest that integrating software engineering practices can enhance game development and improve the process for diverse teams. The research proposes a flexible process based on the Scrum framework for game development phases—pre-production, production, and project closure—to foster improved team

collaboration and enhance video game software development.

Game development, similar to software development, faces almost the same challenges, as evidenced by Petrillo and Jenkins' research (Petrillo et al. 2008). Strategies to tackle these challenges are explored by McKenzie et al. (McKenzie et al. 2021), who analyze the approaches used by game development companies in each stage of development, including pre-production, production, and post-production, highlighting difficulties in managing multidisciplinary teams as noted by Maxim and Guzdial (Maxim e Ridgway 2007). This study diverges by examining how the implementation of agile practices by 3D artists during production stages can alleviate issues in 3D asset development. Unlike Gomes et al. (Gomes Filho et al. 2018), who focus on Kanban to identify and mitigate bottlenecks in 3D animation, and Musil and Spielhofer (Musil et al. 2010), who advocate for Scrum's use throughout game development, this work introduces an alternative to Scrum tailored for the production process from the 3D artist's perspective. The goal is to aid the 3D artist in meeting stakeholder, and programming team needs through practices enhancing their role within the game development production process.

3. Methodology of Research

The lack of well-defined and focused practices for the 3D modeling process in the initial phases of the project revealed problems that hampered the development and maintenance of assets throughout the project. Therefore, we carried out an exploratory case study to define a set of agile practices aiming to align and facilitate the work of 3D artists according to the demands of developers and stakeholders.

Figure 2 shows the three phases employed to conduct the methodology of this research. The process begins with the Analysis phase, where data related to issues in the development of assets are collected and classified from commits in Gitlab, as well as problems reported during weekly meetings. Following the classification of problems in the analysis phase, a literature search is performed to identify best practices and propose suitable solutions for each identified issue. Upon implementing and adapting the proposed practices, this study consolidates the lessons learned from the 3D modeling process.

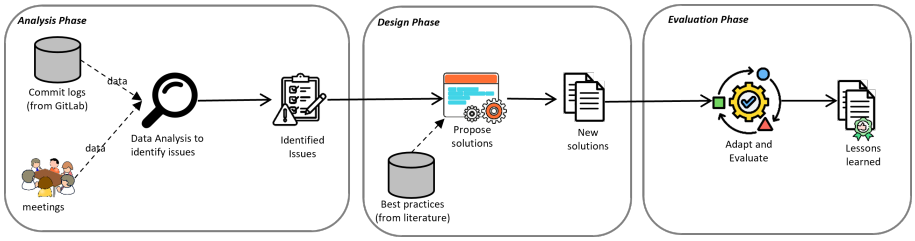

Figura 2. Phases of research methodology.

The project used as a case study of this work encompasses a 9-year R&D Game Development effort with a team comprising 7 PhDs and 27 developers. The 3D modeling team handled dual demands: firstly, focusing on low-level detail modeling to create models with minimal polygons, optimizing processing. These models, depicting terrain with vegetation, vehicles, buildings, people, and objects, met stakeholders' visual

Tabela 1. Description and identified issues.

ISSUE	DESCRIPTION	RQ1	RQ2
Mesh Modification	Modifying existing components or incorporating new elements into already completed models.	1. Insufficient information in the activity description. 2. New modifications emerged during the development of the asset.	1. 3D Artists help gather requirements in the Pre-Production phase. 2. Increase interaction between the 3D artist and the programming team and stakeholders.
Low Poly High Poly	Add or reduce the number of polygons in the mesh.	1. Lack of information on the quality and performance of the assets.	1. Increase interaction between the 3D artist and the programming team. 2. Send prototypes to the programming team throughout the asset development stages.
Texture Modification	Changing colors, tones, and details in the texture of assets.	1. Lack of information on the quality of the assets.	1. Increase interaction between the 3D artist and the programming team. 2. Send prototypes to the programming team throughout the asset development stages.
Rigging Issues	Issues with the behavior of the model rig when imported into the game development engine.	1. Lack of information about the behavior of assets in the development engine.	1. Increase interaction between the 3D artist and the programming team. 2. Send prototypes to the programming team throughout the asset development stages.
Animation Issues	Inserting new animations or making corrections to previously developed animations.	1. Insufficient information in the activity description. 2. New modifications emerged during the development of the asset.	1. 3D Artists help gather requirements in the Pre-Production phase. 2. There is no way to avoid adding new features. they must be accepted and welcomed in the development process.
Standards	Difficulty managing objects, materials, and textures for complex models.	1. Lack of standards in asset development.	1. Establish standards for asset development.

requirements. Secondly, the team delved into highly detailed and complex models tailored for instruction and simulation purposes, specifically for Computer-Based Training (CBT).

3.1. Methodology Phases

Starting with the Analysis phase, during the weekly meetings, we could pinpoint new requests or adaptations emerging towards the end of the asset's development, maintenance needs, or, in some instances, a complete reworking of the asset. As seen in Figure 1, the workflow for asset production is susceptible to numerous adjustments, with stakeholders requesting new functionalities, scale modifications, or texture changes. Furthermore, the programming team may request adjustments to the rig, texture, or alterations in polygon count, either reducing or increasing it. The lack of collaboration among all parties throughout the asset's development hinders the improvement of asset quality, occasionally leading to the rework of completed steps.

We have used issues assigned to the 3D artists in the GitLab tool to identify modeling problems. Additionally, problems reported during the weekly meetings with the development team contributed to the list of issues related to 3D modeling.

Afterward, during the Design phase, by analyzing the commit descriptions on the GitLab platform and the issues reported during weekly meetings throughout the project, the problems were classified into the following categories: Mesh Modification, Low Poly and High Poly, Texture Modifications, Rigging Issues, Animation Issues, and Standards. In order to identify problems and propose solutions, two research questions were asked: Why did this problem occur? (RQ1) and How could it be avoided? (RQ2).

Table 1 summarizes the problems classified in this analysis. In the Issue column, the problems are presented based on the analysis of issues and meetings conducted during the project. Additionally, the Description column provides a brief description of the problems. Finally, columns RQ1 and RQ2 present the answers to the research questions proposed to identify the reasons for each issue and propose possible solutions, respectively.

Finally, during the Evaluation phase, after identifying the main problems involved in the 3D modeling process, we conducted a bibliographic analysis to find suitable solutions for each type of problem. The practices we found were incorporated into the development routines of the 3D artists of the project and were subjectively evaluated throughout the development of their activities. The evaluation considers the following aspects: ease of maintenance of 3D models, reduction of asset development time, collaboration between the artistic team and programmers, reduction of rework, and the quality of assets delivered.

4. Lessons Learned

The fundamental solution to avoid many problems related to the creation of 3D assets was to discard the planned development flow, similar to the waterfall model. The 3D artist would receive the model requirements in this model and carry out the entire production process without interacting with other development teams. The final deliverable was then handed over to the programming team. The lack of interaction with other development teams throughout the production stage revealed problems only when the model was already completed, leading to rework in the production stages. Adopting an incremental development process, such as Scrum, facilitated incorporating various practices throughout the production stage and mitigated the problems listed in Table 1.

The 3D artist selects activities from a prioritized list known as the backlog to work on during a predefined period called a sprint. Daily meetings called daily scrums are conducted throughout the sprint to share progress and address any challenges encountered during the production stage. At the sprint's conclusion, a review and retrospective of the work are undertaken, guided by the Scrum master.

In Figure 3, the production stage executed by the 3D artist is illustrated. On the left, activities in the 3D Assets development pipeline are represented by colored spheres. The artist initiates the process by planning and defining the activities for the sprint, such as creating a character. Subsequently, the artist progresses through the 3D modeling pipeline as the asset production advances. Daily scrum meetings play a crucial role in identifying and mitigating problems that might have been previously reported only at the end of the production process. This approach ensures that the 3D artist's activities are incrementally performed, enhancing model quality and addressing issues as they arise. At the sprint's conclusion, a review and retrospective of the 3D artist's activities occur, involving perspectives from both the artist and the programming team. This collaborative effort, guided by the Scrum master, consolidates the accomplished activities.

In addition, the transitioning to an iterative and incremental development model, a set of practices was adopted to mitigate the problems listed in Table 1 throughout the flow of 3D artist activities.

4.1. User Story

The vague description of the 3D activities produced by the software developers to the 3D artist became evident in our study. Given the multidisciplinary nature of the development team, the diverse knowledge levels among teams posed challenges in requirements gathering. While the 3D artist focused on visual and artistic aspects, the programming team dealt with calculations, angles, polygons, and so forth.

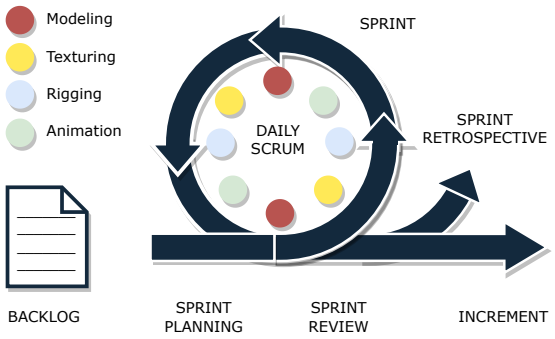

Figura 3. Scrum for assets development.

According to Keith in his book "Agile Game Development"(Keith 2020), each development team, along with stakeholders, has different ways of expressing themselves. Using user stories helps communication between development teams. User stories follow a template determined collaboratively by the team and stakeholders. (Cohn 2004; Cohn 2023) recommends the following: As a **user role**, I want **goal** so that **reason**.

User role: A customer of the game or a pipeline user who benefits from this story. Goal: The story's objective represents a feature or function in the game, tool, or pipeline. Reason: The benefit to the customer or user when this feature or function is used.

The utilization of user stories facilitated the identification of all the needs required by stakeholders, ensuring better detailing for parties with more significant interaction with the user role. Additionally, it met the needs of the programming team in describing all parts of the model that needed to be interactive, thereby resolving the main problems related to mesh modifications.

4.2. Prototype

With the adoption of an incremental model and daily interaction with other development teams, the use of prototypes proved instrumental in identifying issues early in the production stage. It prevented problems from surfacing only when the asset was finalized and sent to post-production. Using prototypes allowed the programming team to conduct tests on simplified models, assessing their functionality and preventing the need for rework on completed steps. This approach effectively reduced issues related to animation malfunctions, rigging, excessive polygon usage, low mesh quality, and even addressed concerns about texture behavior within the game engine.

The use of prototypes not only helped in averting development problems but also contributed to enhancing the quality of the models. As the programming team conducted tests on the 3D models, evaluating the model's performance during its initial development phases became possible. This assessment helped determine whether the model could be further detailed to improve quality or simplified to enhance performance, tailoring the level of detail (LODs) as needed.

4.3. Feedback

Bruce R. Maxim emphasizes that communication between programmers and artists is fundamental. Feedback between teams was essential to validate the approaches used

during the production process and aggregate knowledge. The ongoing communication between the programming team and the 3D artist contributed to aligning the different levels of knowledge among teams. This alignment ensured that all knowledge acquired in the development of an asset was propagated to subsequent models, thereby reducing development time and improving the quality of models. It also helped in avoiding previously recurring problems.

Once the 3D artist became familiar with the behavior of rigging within the game engine, the nuances of textures based on the lighting of the virtual environment, and the behavior of models with varying polygon counts for each asset, the development of assets began to require fewer modifications to the mesh, textures, animations, rigging, and the addition or reduction of polygons throughout production.

4.4. Standards for object naming

In many cases, the need arises to develop highly complex models, each consisting of over 100 functional parts, 80 materials, and, in some instances, each material associated with up to 5 texture maps. The complexity challenged not only the programming team in linking each texture to its respective material and each material to its corresponding part of the model but also posed a challenge for different artists tasked with maintaining models not initially developed by them. It underscored the importance of adhering to development standards.

The most criticized aspect concerning complex models was the naming of interacting parts. Some engines, such as Unreal Engine 5, provide Asset Naming Conventions to standardize model parts (5 2024). Similar principles were adopted to organize intricate models. Each part of the model featured a suffix with the name of the associated material, followed by its function, and finally, the LOD level. The basic structure followed the format: [**Material name**]_[**Part name**]_[**LOD Level**].

For example, the object RadioButton, textured by the buttons material in its most detailed model LOD0, is named **buttons_RadioButton_LOD0**. This nomenclature enables both programmers and artists to identify the material texturing as a part of the model, the function of the object, and the level of detail. Additionally, since some materials included four texture maps with varying resolutions depending on the application, a naming pattern was established. This pattern consisted of a prefix indicating the material name, followed by the resolution, and finally, the type of map. The basic structure for a texture map followed the format: [**Material name**]_[**Resolution**]_[**Texture map**]

Suppose texture maps were created to map all the vehicle's buttons. In this case, the standard name for the serial buttons material, the resolution of the maps (e.g., 4096 pixels), and the type of map (albedo, metallic, roughness, normal map, and ambient occlusion) were specified. For instance, the normal map would be represented as **buttons_4096_normal.png**, and for the ambient occlusion map, it would be **buttons_4096_AO.png**, simplifying the name slightly. For other map types, the full name was used.

The adoption of these development standards contributed to a consistent development process, simplified maintenance, and minimized errors associated with the complexity of the models.

5. Discussion

The most critical issues in the asset development process were associated with inserting new functionalities into existing models, involving the rework of two time-consuming steps in the creation process (modeling and texturing). This problem is strongly linked to the lack of specifications in the initial development stage due to insufficient communication between the 3D artist and stakeholders for describing requirements and the vague delineation of the 3D artist's tasks.

Including the 3D artist in the pre-production stage assisted in specifying requirements and assisted stakeholders in defining implementable aspects in the virtual environment, aligning model development with programmer needs. Additionally, constant feedback during model development stages allowed for identifying and mitigating problems throughout development rather than discovering them solely in the final stage, avoiding rework.

Many problems stemmed from an imprecise description of the 3D model's functionality. Identifying its functionality and interactions in the virtual environment was crucial for preventing issues related to prioritizing specific parts' detailing and distinguishing the object's moving parts. User stories proved to be the optimal solution, providing a concise description of activities from the perspective of the application user and aiding in understanding all the needs associated with each 3D model.

Standards for assets development were crucial due to constant team rotation. This standardization helped the artistic and programming teams align their activities irrespective of team turnover throughout the project, increasing team contribution.

The experiences we gained throughout the project enabled us to identify the main problems in the asset development process and propose and test solutions that facilitated the development of 3D models. By the conclusion of the research, the set of practices consolidated, contributing to the reduction of time in both the development process and the maintenance of 3D models and significantly minimizing rework between development phases.

6. Conclusion

This work summarizes the lessons learned over nine years of a collaborative R&D game development project. Agile practices were proposed to improve the alignment of the 3D artists' work, stakeholders, and developers to overcome the primary issues documented in the project's early stages. The developed set of practices proved comprehensive in resolving all identified problems. The methodology significantly reduced the time required to solve unmitigated problems and assisted in the development stages for the 3D artist. As limitations of the research, the lack of metrics to validate the results and the minimum turnover of the artistic team in the final stages stand out. Furthermore, future work could explore and quantitatively evaluate the performance of suggested practices in other asset development projects.

7. Acknowledgement

We thank the Brazilian Army and its Army Strategic Program ASTROS for the financial support through the SIS-ASTROS GMF project (TED 20-EME-003-00).

Referências

[5 2024] 5, U. E. (2024). Recommended asset naming conventions in unreal engine projects. https://docs.unrealengine.com/5.2/en-US/recommended-asset-naming-conventions-in-unreal-engine-projects/.

[Aleem et al. 2016] Aleem, S., Capretz, L. F., e Ahmed, F. (2016). Game development software engineering process life cycle: a systematic review. *Journal of Software Engineering Research and Development*, 4(1):1–30.

[Beck et al. 2001] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., e Thomas, D. (2001). Manifesto for agile software development.

[Clement] Clement, J. Global video game market value from 2020 to 2025.

[Cohn 2004] Cohn, M. (2004). *User stories applied: For agile software development*. Addison-Wesley Professional.

[Cohn 2023] Cohn, M. (2023). User story template: What it is and why it works so well. https://www.mountaingoatsoftware.com/blog/why-the-three-part-user-story-template-works-so-well. Accessed on 2023-04-23.

[Gomes Filho et al. 2018] Gomes Filho, A. F., Alencar, D., e de Toledo, R. (2018). Agile in 3d: Agility in the animation studio. In *Agile Methods: 8th Brazilian Workshop, WBMA 2017, Belém, Brazil, September 13–14, 2017, Revised Selected Papers 8*, pages 63–76. Springer.

[Hristov e Kinaneva 2021] Hristov, G. e Kinaneva, D. (2021). A workflow for developing game assets for video games. In *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–5. IEEE.

[Keith 2020] Keith, C. (2020). *Agile Game Development: Build, Play, Repeat*. Addison-Wesley Professional.

[Maxim e Ridgway 2007] Maxim, B. R. e Ridgway, B. (2007). Use of interdisciplinary teams in game development. In *2007 37th Annual Frontiers In Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports*, pages T2H–1. IEEE.

[McKenzie et al. 2021] McKenzie, T., Morales-Trujillo, M., Lukosch, S., e Hoermann, S. (2021). Is agile not agile enough? a study on how agile is applied and misapplied in the video game development industry. In *2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE)*, pages 94–105. IEEE.

[McKenzie et al. 2019] McKenzie, T., Trujillo, M. M., e Hoermann, S. (2019). Software engineering practices and methods in the game development industry. In *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, pages 181–193.

[Musil et al. 2010] Musil, J., Schweda, A., Winkler, D., e Biffl, S. (2010). Improving video game development: Facilitating heterogeneous team collaboration through flexible software processes. In *Systems, Software and Services Process Improvement: 17th European Conference, EuroSPI 2010, Grenoble, France, September 1-3, 2010. Proceedings 17*, pages 83–94. Springer.

[Petrillo et al. 2008] Petrillo, F., Pimenta, M., Trindade, F., e Dietrich, C. (2008). Houston, we have a problem... a survey of actual problems in computer games

development. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 707–711.

[Politowski et al. 2021] Politowski, C., Petrillo, F., Ullmann, G. C., e Guéhéneuc, Y.-G. (2021). Game industry problems: An extensive analysis of the gray literature. *Information and Software Technology*, 134:106538.

[Schwaber e Sutherland 2017] Schwaber, K. e Sutherland, J. (2017). The scrum guide-the definitive guide to scrum: The rules of the game. *SCRUM. org, Jul-2013*.