Leveraging Unity for 2D Pixel Game Development: Techniques and Best Practices

Ke Wang

Pittsburgh Institute, Sichuan University ,Chengdu, 610065, China

Abstract. The rapid evolution of game development technologies emphasizes the demand for high-quality graphics and immersive gameplay. Meanwhile, there has also been a pixel art revival in the game development industry. This research explores the process of creating a 2D pixel-art adventure game with the Unity engine, figuring out some common problems that developers face and efficient solutions to them. To address these problems, the research uses several powerful toolsets offered by Unity and some other efficient approaches. This paper makes a 2D pixel-art game as a case study, focusing on the challenges of animations, map design, and gameplay mechanics. This paper found an innovative 3D-to-2D animation workflow, allowing developers to create better animations with less time. By applying Unity's tool Tilemap Editor, this paper also found an efficient way to design levels. Additionally, this research offers some suggestions on how to improve game mechanics. The findings underscore the importance of consistent visual design and thorough playtesting. This research contributes to pixel-art games by providing practical solutions for game developers and demonstrates the potential of the Unity engine.

1 Introduction

As technologies develop, players' demands and trends for game development are evolving rapidly. Players expect higher graphics quality, more snappier experiences, and richer interactive content. However, Game development is a diverse and complex area, and developers need powerful tools and game engines to satisfy these demands. As one of the world's most powerful game engines, Unity serves an important role in the game development industry. As a multiple-platform game engine, Unity provides powerful functions, rich resource repositories, and user-friendly interfaces, which allow developers to create 2D and 3D games efficiently. So far, Unity has become the world's most widely used game engine. By using Unity as their game engine, both individual developers and large companies can turn their concepts into reality efficiently.

Researchers have achieved great progress in the Unity engine in different research directions. Through developing a simple shooting game, Koulaxidis and Xinogalos used basic optimization techniques to improve Unity's performance in mobile games [1]. Adam and Nikolaus focused on specific problems when developing Virtual Reality using Unity [2].

Corresponding author:2023141520112@stu.scu.edu.cn

They introduced a powerful toolkit and successfully simplified the process of getting experiments up and running quickly without scripting [2]. Hye and Won constructed a handwriting system using Unity [3]. They applied the Convolutional Neural Network and EMNIST data set to this system within the framework of unity, demonstrating the power of Unity [3]. In summary, researchers are still trying their best to extend the application of Unity in various fields to propose solutions in response to diverse and complex demands for game development.

Although there are many excellent 2D pixel-art adventure games in the market, many developers still feel unfamiliar with the developing processes. When inexperienced developers begin to create a 2D pixel-art adventure game, they may encounter plenty of problems that haven't been discussed widely and have trouble finding effective solutions. This paper intends to find some common problems most developers may encounter and propose solutions to these problems. This would also demonstrate Unity's power in making such 2D games and explain how Unity contributes to its development process. To achieve this, this research is trying to make a 2D pixel-art adventure game, exploring the completed process of making such games. Using Unity, this research plans to exhibit every important step of this process, including game design, animation production, storytelling, map design, and mechanism design and implementation. Finally, this paper will address common challenges and showcase effective solutions to demonstrate the engine's capabilities and contributions to game development.

2 Game Development Trends and Unity Engine

2.1 Trends in Game Development

The game development field has changed significantly as advanced technologies developed over the past decade. Players tend to play games with high-quality graphics, cross-platform support, and immersive Experiences. Meanwhile, players' enthusiasm for 2D pixel Art games never faded, which caused a pixel art revival. This trend also changed the way developers create their games. Developers also change the tools and methodologies they use.

Contrary to the general acknowledgment that high-quality graphics are just photorealism, high-quality graphics can also take many other forms, like pixel art style. Pixel art games were once the result of limited technologies. Hardware at that time only supported this kind of graphics. However, pixel art has already developed into a new game type, attracting many players even in today's game market, where high-definition graphics dominate the game industry. It enjoys a great reputation among players not only because it's nostalgic but also because it represents a unique aesthetic choice [4]. What's more, pixel art games are suitable for indie developers because indie developers can create great pixel art games without expensive resources or large teams. Until now, pixel art games are still the most common choice for indie developers. As a result, many indie games, such as Celeste and Stardew Valley, have achieved great success around the world.

As gaming platforms diversify, from consoles and PCs to mobile devices. The ability to smoothly transit games across multiple platforms has become crucial, especially for 2d game developers. Compared to those 3D AAA games, which have a high requirement for devices' performance and are impossible to run on devices like mobile phones, 2D games have a low requirement for performance. As a result, if 2D games can be published across platforms, they can satisfy players' demands to play games on any device, especially mobile phones. Also, this cross-platform ability can be a great advantage for pixel art games. It allows these games to have more audience without significant optimization.

Modern players not only love games with appealing visuals, but they also demand rich interactive experiences. Game mechanics and storytelling are the two most important parts of such experiences. For 2D pixel-art games, this trend goes beyond platforming or puzzle-solving. Undertale, a wonderful 2D pixel-art game, won its players through its powerful storytelling and player interaction, proving that immersive experiences are also important in small-scale 2D games.

2.2 Introduction to Unity Engine

Developments need powerful tools to satisfy players' demands for high-quality graphics, cross-platform support, and immersive Experiences. Unity engine is one of the most powerful game-development tools in the market.

To begin with, the core feature of Unity is its robust graphics engine, which could provide high-quality rendering capabilities. And this ability is not only for 3D games. For 2D pixel-art games, through the powerful tool universal render pipeline, which is known as URP, developers can create fantastic 2D light, providing players with more immersive visual experiences. What's more, Unity's animator component can help developers make vivid animations fairly quickly and control the transitions between animations more precisely. Additionally, Unity uses C# as its programming language and invents the Monobehaviour class, making it accessible to minimal programmers [5]. Another important strength of Unity is its ability to develop cross-platform. Using the Unity engine, developers can deploy their games across multiple platforms with a tiny number of adjustments. Unity also provides an asset store with rich resources. Developers can find all the resources they need, such as textures, animations, models, and scripts. In summary, Unity's features and tools facilitate developers significantly in every aspect of game development.

3 Case STUDY: DEVELOPING a 2d PIXEL-ART ADVENTURE GAME

3.1 Introduction to the Game

This research develops a 2D pixel-art adventure game as a case to explore how to use Unity to facilitate the development process. This game is a 2d pixel-art action game with powerful storytelling. It is set in a dark, mystical world. Players will face a series of challenges and go through different levels with unique enemies. This project uses Unity as its game engine and Visual Studio Code as its integrated development environment. Most of the development tools are offered by the Unity engine, such as Sprite Renderer, Animator, Rigidbody2D, and so on. Besides, this project uses Maya 2024 as the 3D modeling software and Aseprite as the art tool.

The core concept of the game is a dual-character system. The protagonist that the player controls has two distinct Souls: Black and White. The "one body, two souls" mechanic is crucial to both the gameplay and the storytelling. They have different fighting styles and opposite personalities. Thus, players can try various weapons. Also, players can choose which skills to enhance and upgrade the protagonist. Their relationship is the biggest mystery of the whole story. The narrative is dark and intricate, with themes of identity and duality. Players will finally uncover the mysteries around Black and White and then make a choice between them. Different choices will lead to different endings.

3.2 Common Problems and Solutions

In this section, this paper will introduce some common problems developers may face and offer some effective solutions.

3.2.1 Animation Generation

To begin with, the most common problem developers face is how to make great visual aesthetics in a very short time. Creating high-quality pixel art can be especially challenging for indie developers with limited experience in graphic design. The process of hand-drawing every frame of an animation is time-consuming and annoying. In this project, the solution is developing a visually compelling game with a 3D animation workflow. The first step of this workflow is to create a 3D model of the character. Second, use this 3D model to make 3D animations. Finally, 3D animations can be converted into 2D pixel-art animations using a program published in GitHub. This workflow has several advantages over traditional 2D animation workflows. Once developers create the characters' 3D models, they don't need to make animations alone. They can buy some excellent animations in the Unity asset store or other asset stores and apply these animations directly to their characters. It is convenient and easy, especially for developers with little experience in graphic design. This workflow also facilitates the modification process [6]. For example, if developers want to give the characters new costumes, they can add the models of costumes into the characters' models instead of drawing a new animation by hand [6]. In summary, this workflow can generate excellent animation efficiently and easily for any developer.

3.2.2 Map Design

Another problem developers may encounter is map design. Unity engine offers a tool called Tilemap, which can address this problem, but some junior developers know little about its powerful functions. This project designed five different levels using the Tilemap, demonstrating its great power to facilitate developers. Tilemap's core feature is dividing map assets into small squares, allowing developers to 'paint' the map using these squares. For example, this project has a map asset with grass and mud. Developers divide this asset into different squares, which include different types of grass squares and mud squares. Then, developers can choose the squares they want and use various Brush tools to add these squares to their scenes. This process is more like painting and really facilitates the map design process. This feature is not only innovative but also facilitates the following development processes. For example, the player interacts with the maps or environments through colliders. Colliders are important components that allow players to collide with something instead of going through them. In this project, if the developers want the players to collide with the mud and pass through the grass, they don't need to manually add colliders to the mud [7]. They can just find the mud squares and set colliders to the squares, and then all the mud in the scenes will have colliders. Also, if developers aren't satisfied with the shape of the collider, they can edit it manually to make it perform better at their own will. In conclusion, with the powerful functions Tilemap offers, developers can design and create their maps quickly and conveniently.

3.2.3 Implementing the Game Mechanism

Besides animation generation and map design, the most difficult problem for developers is implementing the game mechanism with code. Developers may find it hard to bring the concepts into reality, especially for some issues concerning the physics system. However, the

Unity engine's physics system is extremely powerful and accessible to developers. Unity's physics system has great advantages in collision detection, applications of force, and rigid body simulation [8]. Within the class Monobehaviour, which was invented to offer all kinds of tools and facilitate the developers, developers can easily implement game mechanisms with simple codes. This allows developers to implement some characters' behaviors in logic similar to the real world. For example, if developers want to make a game object jump, they can first give the object a mass. They can use tools offered by Unity engine, such as AddForce, which functions like adding a force to the object. Then, the object will respond to this force, obeying Newton's laws. It's such a natural and direct way to write codes that developers do not need to come up with other abstract methods to achieve the same function. Generally, Unity's powerful physics system helps developers implement game mechanisms in a natural and fast way.

4 Evaluation of this Project

4.1 Strengthes of this Project

This development case of the 2D pixel-art adventure game showcases several strengths, which contributed to offering some effective solutions to specific development problems. One of the most significant strengths of this project was the detailed explanations and applications of Unity's powerful toolset. By combining Unity's features, such as the Tilemap Editor, the physics system, and the Monobehaviour, developers are able to create a game with excellent graphics design and appealing mechanisms. Besides the utilization of Unity's core features, this project also finds an innovative animation workflow. This workflow allows the team to produce high-quality animations much more efficiently than traditional frame-by-frame animation workflow. This approach can significantly reduce the time and effort required for animations. Developers will also find it convenient to adjust their animation under this workflow. This innovative workflow is especially beneficial for developers without much graphics design experience. For those experienced developers, this workflow can also speed up their development process.

4.2 Weaknesses of this Project

While the project demonstrated numerous strengths, several weaknesses and areas could be improved. It's important to find these issues to refine the development processes. The 3D-to-2D animation workflow achieved overall success in improving efficiency. However, it also has some weaknesses. One of the main weaknesses is a lack of visual consistency when developers have little graphics design experience. The reason is that this animation workflow can only produce characters' animations. However, a consistent art style is significant. Game artists should use various visual styles and stylistic elements to create a cohesive whole [9]. Developers with little graphics design experience can use this workflow to generate animations, but they may be unable to produce other visual elements like backgrounds with a consistent style. As a result, all the visual elements don't align with the intended aesthetic. The dual-character mechanic also presented certain challenges. Some players reported that the transition between characters was clunky. This issue arises from the complexity of managing two distinct characters, which leads to transitions that are not as smooth as expected. Although Unity offers powerful tools, such as animation controllers and state machines, to help developers address these problems, developers still need to improve their programming skills when addressing complex problems. Another area that could be improved is the playtesting process. Developers thoroughly tested this game for technical

stability. However, when players begin to play this game, they still find many bugs that could affect their gaming experience. One of the most important issues among them is the difficulty balancing. Some players found certain levels to be difficult, while others found some bugs to cheat in games. All these bugs could negatively impact player engagement.

4.3 Suggestions for Future Projects

According to the strengths and addressing the weaknesses identified in this project, several recommendations can be made to improve the development process and enhance the quality of future 2D pixel-art games. First, to ensure a more cohesive visual experience, future projects should hire some graphics designers. If not, developers should establish clear guidelines for the creation of game assets to ensure that all elements adhere to the same pixel art style. What's more, Improving the fluidity of the character-switching mechanism should be a priority in future projects. Creating smoother transition animations should be the first step. Then, developers also need to create more intuitive user interfaces to guide players. Finally, increasing the depth of playtesting is essential for future projects. The developers could publish the test version of the game to players and collect players' feedback. This can help developers find some technical issues and major bugs that could impact players' gaming experience. Also, through players' feedback, developers can balance the difficulty of certain levels. In this way, the development team could ensure a polished final product.

In conclusion, following these recommendations, future projects can build on the success of this game while addressing these weaknesses. Focusing on visual consistency, fluid mechanics, and a deeper playtesting process can contribute to creating more successful 2D pixel-art games [10].

5 Conclusion

This study explored the development of 2D pixel-art games using the Unity engine, showing some common challenges developers face and how Unity's powerful tool can address these problems. By developing a 2D pixel-art game as a case study, this research explained many tools to facilitate the development process, including 3D-to-2D animation workflow, the Tilemap Editor, and Unity's physics system. These toolsets help developers create a 2D pixel-art game with high-quality graphics, immersive gameplay, and an efficient development process. enable developers to focus on creative design without worrying about the implementation of underlying technologies.

The findings of this study reveal several approaches to speed up the development processes and improve players' gaming experience using powerful tools offered by Unity. The innovative 3D-to-2D animation workflow effectively reduces the time and effort needed for character animations and improves the quality of graphics at the same time. This workflow proves junior developers with little experience can also make high-quality games. The dual-character mechanic posed some challenges in ensuring smooth transitions and needs further improvement.

This research emphasizes the importance of a cohesive visual style in future development projects. It also suggests that thorough playtesting and further exploration of efficient game development workflows could benefit both game developers and players in the field of 2D pixel-art games. The findings of this project offer valuable guidance for future research in this field.

References

1. G. Koulaxidis, S. Xinogalos, Improving Mobile Game Performance with Basic Optimization Techniques in Unity, Modelling, **3**, 201-223 (2022).
2. O. A. Bebko, F. N. Troje, Experimental Design with Unity Game Engine, J. Vis., **20**(11), 810-810 (2020).
3. M. H. Lee, H. W. Lee, Development of Handwriting Recognition Using Deep Learning in Unit3yD, Journal of the Korean Computer Game Society, **32**(1) (2019).
4. S. Wei, X. Jiarong, Z. Shan, et al., Development and Implementation of Pixel Art Game Based on Human-Computer Interaction, in Proc. Beijing Inst. Graphic Commun., China, (2023).
5. T. Bhosale, S. Kulkarni, S. N. Patankar, 2D Platformer Game in Unity Engine, Int. Res. J. Eng. Technol. (IRJET), **05**(04), 3021-3024 (2018).
6. M. Wiik, Integrating 3D into the 2D Game Character Workflow, Bachelor's Thesis, Kajaani Univ. Appl. Sci., (2013).
7. T. T. Nguyen, Introduction to 2D Game Development with Unity, Bachelor's Thesis, Metropolia Univ. Appl. Sci., (2021).
8. F. Hussain, H. Shakeel, F. Hussain, N. Uddin, T. L. Ghouri, Unity Game Development Engine: A Technical Survey, Univ. Sindh J. Inf. Commun. Technol., **4**(2), 73-81 (2020).
9. M. Lauriste, Impact of Visual Design and the Game Development Process in Small Teams, Bachelor's Thesis, Tallinn Univ. Technol., (2024).
10. M. Foxman, United We Stand: Platforms, Tools, and Innovation with the Unity Game Engine, Soc. Media + Soc., **5**(4), 1-10 (2019).