Pixelat3D: a Technique to Render 3D Scenes with Pixel Art Aesthetics

Naomi Celestino Ribes[1], **Anderson Maciel**[1,2], **Rafael Piccin Torchelsen**[1,3]

[1]AKCIT Lab - Instituto de Informática - UFRGS

[2]Instituto Superior Técnico, University of Lisbon / INESC-ID

[3]Centro de Desenvolvimento Tecnológico - UFPel

{ncribes,amaciel}@inf.ufrgs.br, rafael.torchelsen@inf.ufpel.edu.br

Abstract. Introduction: *Pixel art has been explored since the early development of 2D computer graphics and remains a popular aesthetic, particularly in video games. However, reproducing this style in real-time 3D environments remains challenging.* ***Objective****: To develop a shader that emulates pixel art aesthetics in real-time 3D while preserving visual detail.* ***Methodology****: The proposed approach uses priority maps to dynamically adjust the texture level of detail (LOD) based on object distance and viewing angle, combined with a post-processing stage for object outlining. The visual results are evaluated through user-perceived quality testing (n=62).* ***Results****: The proposed method, Pixelat3D, showed a significant improvement in user preference across most evaluation scenarios.*
Keywords *Pixel Art, Shader, Non-Photorealistic Rendering, Video Game, Art.*

1. Introduction

Pixel art is a form of digital art in which images are created and edited at the pixel level. It is characterized by a blocky, low-resolution aesthetic and the deliberate use of limited color palettes. Originally driven by hardware limitations, it remains widely used in modern games, particularly in 2D titles. A key challenge of this style lies in its low resolution, where each pixel must convey a large amount of visual information. Excessive detail, however, can introduce visual noise and hinder object readability. Achieving this balance requires considerable effort and experience from artists.

This problem is amplified in 3D scenes, where the angle between textures and the camera changes dynamically, altering the visible area of each texture. As the visible area decreases, decisions must be made about which information to preserve, balancing noise reduction with contextually relevant visual expression. Traditional texture filtering methods rely on interpolation to reduce noise, but this approach smooths out the pixelation that defines the style. Furthermore, the importance of visual features is often context-dependent (sometimes even driven by artistic or emotional intent) which standard filtering techniques do not account for. In 2D, this issue is more constrained, as artists typically need to produce only a single version of each texture. In contrast, extending this approach to 3D would require an impractical number of manually crafted variations. This may explain the relative scarcity of 3D games with pixel art quality comparable to 2D titles.

This work proposes a method for rendering scenes in real time in a pixel art style, extending previous approaches with focus on keeping the readability of features at greater

Figure 1. A scene rendered with Pixelat3D. Note how closer objects (such as the right side bricks) have a higher level of detail and visible shading on their texture. In contrast, farther blocks lack this shading, keeping only primary visual elements to maintain a clearer image.

distances and oblique angles. The method incorporates user-defined priority maps for each texture, specifying features to be preserved or merged in accord to a pixel art style, which often involves complex and subjective decisions. Fig. 1 illustrates the effect in a scene with varying textures and distances. We also present a qualitative user evaluation to assess the perceived aesthetic quality of the proposed technique.

2. Related Works

Pixel art is a popular style among game developers and requires specialized training to create [Silber 2015, Serpa e Rodrigues 2019]. Works by [Coutinho e Chaimowicz 2022, Coutinho e Chaimowicz 2024, Saravanan e Guzdial 2022, Han et al. 2018, Wu et al. 2022] approach generation of 2D sprites using AI, while [Moreira et al. 2022] employs normal maps to improve asset quality, and [Kuo et al. 2016] proposes a method for generating animation frames. These studies highlight the need to reduce the manual effort involved in creating such assets, even in 2D. However, for 3D environments, we did not find comparable initiatives in the literature. Only a few informal attempts were identified online [King 2018, Recordá Illas 2024]. Our work focuses on improving the quality of texture resampling in dynamic 3D scenes, where 2D approaches are not directly applicable, as they are not designed for varying sampling conditions such as zoom and perspective transformations.

Cel-shading is related in that it aims to provide artists with manual control over lighting and filtering. However, unlike pixel art, it does not impose constraints on resolution. [Barla et al. 2006] explores the use of 2D gradients (rather than traditional 1D) enabling more artistic flexibility and control compared to physically accurate approaches.

Both [Pope 2017] and [Johansen 2025] present 3D dithering techniques. [Pope 2017] proposes mapping a dither matrix onto a sphere fixed in the scene to stabilize the image and reduce frame-to-frame inconsistencies during camera movement, improving rotational consistency but still exhibiting artifacts during translation. [Johansen 2025][1] addresses this limitation by stabilizing the texture directly on surfaces and adapting the matrix based on the camera, maintaining a consistent apparent size. Dithering is relevant to shading, as it can increase the perceived color range in gradients. Additionally, the use of surface-based texture manipulation is conceptually similar to the approach adopted in Pixelat3D.

[1] https://youtu.be/HPqGaIMVuLs

The current state of the art does not address the same objectives as this work. Research on real-time pixel art stylization remains limited, and no approaches specifically focused on texture detail preservation were identified during our review. This gap may be explained by the wide range of artistic styles within pixel art, which makes parametrization and implementation challenging. As a result, such techniques are often explored through topology design or image generation rather than real-time rendering.

3. Pixelat3D

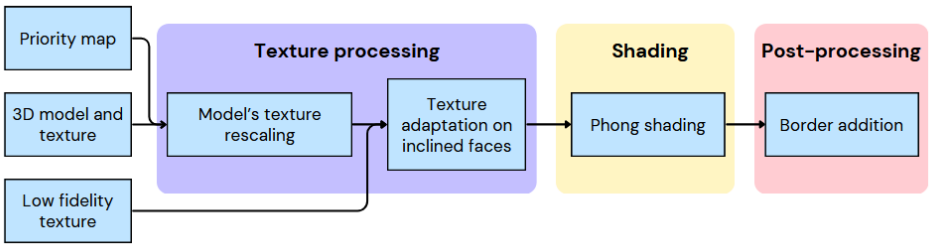


Figure 2. Overview of the rendering pipeline.

Our method consists of two main steps. First, each object's texture is processed in screen space to account for viewing angle and distance. Then, a post-processing stage adds stylized outlines to the objects (see Figure 2). The scene is rendered at high resolution and subsequently downscaled to achieve a pixelated visual effect.

3.1. Texture processing

As the distance from the camera increases, texture detail is progressively reduced, similarly to a mipmapping approach, while also attenuating features based on a priority map. This priority map encodes a ranking, represented as a grayscale range (Figure 3b), that determines the relative importance of features for object readability. It is provided by the user alongside a low-fidelity version of the original texture (Figure 3c), which is used later in the pipeline. These two textures must be created by the artist, a manual process that should consider which features are of greater overall importance for the recognition of the surfaces. This might not be an initially intuitive process, seeing as in some cases shadows and highlights can produce undesired noise if given more priority than the "bulk" of the texture; at other times, this highlight could be of higher significance i.e. when the illusion of depth itself is a defining feature for the shape. These are precisely why the manual artist intervention is still a necessity, seeing as the automation of such arbitrary choices would be a significantly complex, if not entirely impossible task.

Depth is computed independently for each object, as different texture and geometry configurations can vary drastically. Two user-defined thresholds, $nearLim$ and $farLim$ (Figure 4a), define the distance range over which texture rescaling occurs. This value is then normalized (Eq. 1) to the interval shown in Figure 4b:

$$ScaledDepth = \frac{Distance}{farLim} - \frac{nearLim}{farLim} \tag{1}$$

This provides greater control over the algorithm's behavior, as different objects and texture LODs may introduce noise at varying distances, with no improvement in

(a) (b) (c)

Figure 3. Example of textures provided to an object, with its texture (a), priority map (b) and low fidelity texture (c).

readability beyond a point. Texture regions closer than $nearLim$ retain their original resolution, while those beyond $farLim$ are clamped to 1/16 of their original size.

To prevent distortion, the scaling intensity I (Fig. 2, Texture rescaling) is rounded down and kept within a range between the texture's original dimensions $Dim(x, y)$ and 1/16 of its size (Eq. 2). It is also exponentially scaled (2^I, Eq. 3), which aims to prevent pixel misalignment (mixels) within the texture, and limits reduction to four layers:

$$I = \lfloor ScaledDepth(1 - Distance) \rfloor \mapsto \begin{cases} 0 & : I < 0 \\ I & : 0 \leq I \leq 4 \\ 4 & : I > 4 \end{cases} \tag{2}$$

$$Scale(x, y) = \frac{Dim(x, y)}{2^I} \tag{3}$$

This scale value essentially allows the reduction of the model's texture resolution without altering its dimensions. The texture is subdivided in quadrants (Eq. 4), with size determined by the previously calculated scale which, along with the pixel size $pSize(x, y)$ (Eq. 5), finally culminates in the scaled UV positions (uv_s, Eq. 6), ensuring the UVs stay properly aligned within the pixel grid. Each quadrant then takes the color of the highest priority pixel within its area, which is determined by the priority map (Figure 5).

$$Offset(x, y) = \frac{Dim(x, y)}{Scale(x, y)} \tag{4}$$

$$pSize(x, y) = \frac{(1, 1)}{Dim(x, y)} \tag{5}$$

$$(u_s, v_s) = \frac{\lfloor (u, v)Scale(x, y) \rfloor}{Scale(x, y)} + Offset(x, y)pSize(x, y). \tag{6}$$

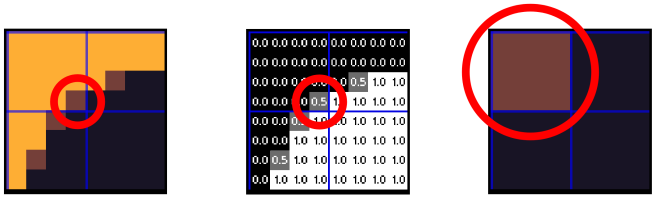
Finally, the cosine of the angle between the surface normal and the eye direction is computed, yielding a value between -1 and 1 that represents face orientation (Figure 6). Faces observed at steep angles can then be replaced with low-fidelity versions to improve readability under a severely reduced screen area. As the available screen space decreases, conveying meaningful detail becomes impractical; beyond a given threshold, textures are replaced by solid colors to avoid noise or blur, as it would hurt the pixel art aesthetic.

3.2. Post-processing

The addition of contour requires a border detection algorithm, done through the depth difference of each pixel to its neighbors [King 2018] (Eq. 7). Neighbors are identified for

(a) (b)

Figure 4. In the bird's eye view (a) the user-defined limits for each object ($nearLim$ (n$_i$) and $farLim$ (f$_i$) where i is the object's index) are represented by the colored-coded areas. Points located inside the interval are kept between 0 (black) and 1 (white), as can be visualized by the gradients in (b); this value ($ScaledDepth$, see Eq. 1) is later used to determine the proper texture scale.



(a) (b) (c)

Figure 5. A texture going through the rescaling process. The 8x8 image (a) is intended, in this example, to be rescaled to 1/4 of its original size. Each pixel in the priority map (b) is scanned and returns the highest priority value (circled in red) for each quadrant. The color of the relative position in the original texture is then applied to all the pixels within the quadrant in screen space, resulting in a single color for the entire quadrant (c).

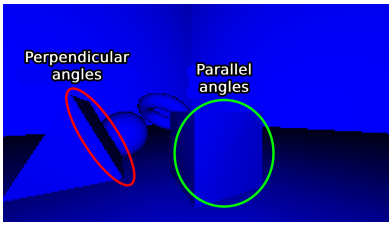


Figure 6. Visualization of the faces' relative angle to the screen plane, where black means perpendicular angles in relation to the camera's view vector ($view(x, y, z) \cdot normal(x, y, z) \approx 0$) and blue means parallel angles ($view(x, y, z) \cdot normal(x, y, z) \approx 1$).

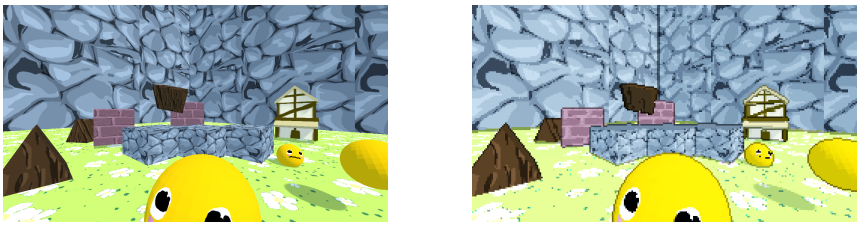
a fragment, allowing the scanning of either four or eight pixels based on user preference (see Fig. 7); if the depth of any neighbor is closer to the screen (within the threshold), a value of 1 is returned (Eq. 8), marking the pixel as an outline. These are all combined into a mask and interpolated with the initial image, using a darker shade of the closest color.

$$Outline = \sum_{i=1}^{n} Distance - Distance_i \tag{7}$$

$$\forall Outline > Threshold : Outline = 1 \tag{8}$$

(a) (b)

Figure 7. Comparison between different neighbor counts. With four neighbors (a) the highlighted pixel isn't considered a border, since none of its neighbors (highlighted in blue) are closer to the screen; with eight neighbors (b) the top right neighbor is closer, thus making this a border pixel.



(a) (b)

Figure 8. Scene rendered at full resolution (a) and with Pixelat3D (b). Notice how smaller details, such as the bricks' texture, are exaggerated to account for the reduced screen space at longer distances; also, both the wood and stone textures omit smaller lines, overtaken by the surrounding colors.

4. Visual Results

To demonstrate the effectiveness of the proposed method, we compare results with and without the Pixelat3D shader (Figure 8). Figures 9 and 10 illustrate the effects of priority mapping at varying distances, while Figure 11 presents texture behavior at acute viewing angles, and Figure 12 shows the resulting silhouettes. An example video is also available[2].

The main challenges in the stylization process stem from the need to convey large amounts of information in a limited space, without relying on color interpolation. Additionally, the method relies on user input to define artistic intent, which limits full automation, particularly because it requires manually created priority maps. The effectiveness of the shader also depends on texture characteristics: high-contrast textures tend to produce clearer results, whereas highly detailed textures may lead to noisy outputs.

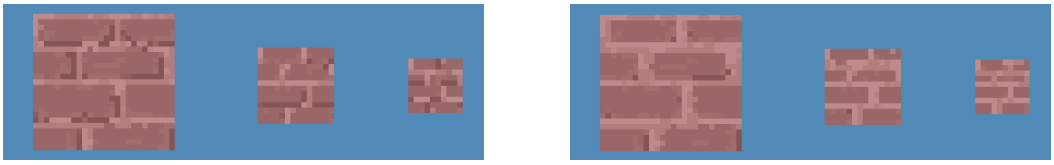
Tests without priority mapping show inconsistent readability across distances, mainly due to sub-pixel behavior during rescaling, which becomes clearer at lower resolutions. Incorporating Pixelat3D improves feature clarity at greater distances (see Fig. 9). Fig. 10 also shows the influence of different priority maps on texture rescaling.

Fig. 11 presents a comparison at different viewing angles, both with and without Pixelat3D. The method improves texture definition at acute angles and, when necessary, replaces fine details with solid colors, reducing visual clutter. Finally, the post-processing further stylizes by outlining objects, reinforcing the pixel art aesthetic (see Fig. 12).

5. User Evaluation

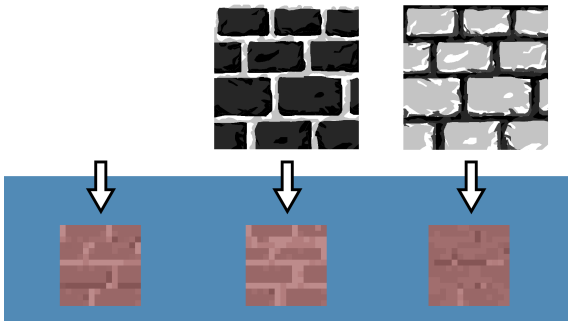
We evaluated the methods using a questionnaire that assessed users' opinions on aesthetics, similarity to pixel art, and readability at different distances. Seeing as this was

[2]https://www.inf.ufrgs.br/vislab/sbgames26-pixelat3D



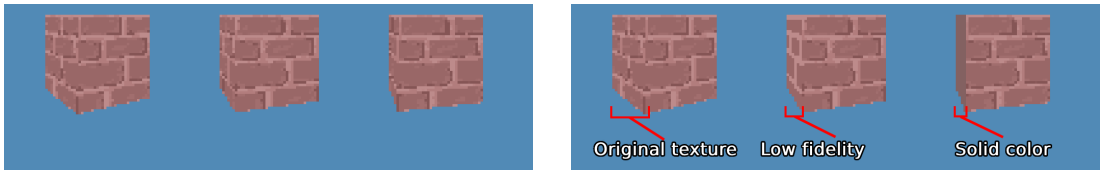
(a) (b)

Figure 9. Comparison between an object's texture at different distances. (a) uses nearest-neighbor sampling, preventing the interpolation of colors which would result in a reduction of the pixelation and lead away from the art style. Note how the bricks in (b) are more clearly defined, as they are deemed the most important visual feature by the priority map used.



(a) (b) (c)

Figure 10. Comparison between nearest-neighbor sampling (a), and textures using different priorities, with bias towards joints (b) or bricks (c), showing how higher priority areas (in white) stay visible at higher distances.



(a) (b)

Figure 11. Comparison between different angles. Notice better detail definition in the middle column of (b) when compared to texture change in (a).

(a) (b) (c)

Figure 12. Comparison between different outline styles, without any post-processing (a), four neighbors (b) or eight neighbors border detection (c).

still an early proof of concept, performance was still not accounted in our evaluations.

5.1. Questions and Stimuli

We formulated a series of questions based on validated instruments such as the System Usability Scale (SUS), Player Experience Inventory (PLEX), and visual perception studies in games, though disregarding questions about game feel and immersion. These were sent to computing and arts faculty members and students at UFPel by e-mail. The questionnaire featured videos and images of rendered scenes in different configurations to be compared, both with and without Pixelat3D, and was divided into five sections. Before being presented with the scenes, participants were also asked demographic questions.

Section (S1) showed a side-by-side video of a scene rendered both with the default shader (A), and with priority mapping and acute angle texture modifications enabled (B). The participants were then asked to answer within a Likert scale of 1 (“More easily on version A”) to 5 (“More easily on version B”) the questions: 1) “The scene elements (e.g.: objects, surfaces) are distinguishable at a distance”; 2) “The objects are distinguishable between themselves”; 3) “The surfaces at steep angles are perceptible and coherent”.

Next (S2), the same recorded scene was presented, now comparing Pixelat3D rendering with and without outlines (labeled A and B, respectively). Again, they answered on a scale of 1 (“Version A”) to 5 (“Version B”) the questions: 1) “Which version shows better distinction between objects?”; 2) “Which version shows aesthetics closer to pixel art?”; 3) “Which version is visually more pleasant?”. Section (S3) showed a video with three versions of a rendered image with Pixelat3D having no outlines (A), four neighbors (B) and eight neighbors (C): “About the outline style, you prefer:”, with options “Version A (None)”, “Version B (Subtle)”, “Version C (Strong)” and “Depends on the object”.

The next section (S4) showcased two videos, one without priority mapping and one with (labeled A and B, respectively); each of the videos featured four spinning cubes at a distance with four different textures: a brick wall (little noise and contrast), cobbled stones (little noise, high contrast), hedge leaves (high noise and contrast), and a tree log (high noise, little contrast). These were followed by the questions: 1) “Which version do you consider more readable?”, with answer options “Version A” and “Version B”; 2) “Which version has more pleasant aesthetics?”, with answer options “Version A” and “Version B”; 3) “Briefly justify your answer” (with a text field); 4) “Did you notice any visual inconsistencies (e.g.: objects or surfaces ’flickering’, details disappearing)?”, with answer options “Yes, on version A”, “Yes, on version B”, “Yes, on both versions” and “No, in neither version”; 5) “If yes, was there a difference in the inconsistency degree between versions?”, with answer options “Yes, the inconsistency was greater on version A”, “Yes, the inconsistency was greater on version B” and “No, they are equally inconsistent”. Another section (S5) followed with the same questions and video format, now featuring the cubes significantly closer to the screen. Finally, a free text field was presented for participants to provide feedback and suggestions regarding readability and aesthetics.

5.2. Experimental results

The conducted questionnaire gathered responses from **62 volunteers**, from which: 32.8% answered “High”, 16.4% “Moderate”, 23% “Little” and 27.9% “None” for the question of experience with pixel art games. Furthermore, 41% were female, 54.1% male, and 4.9%

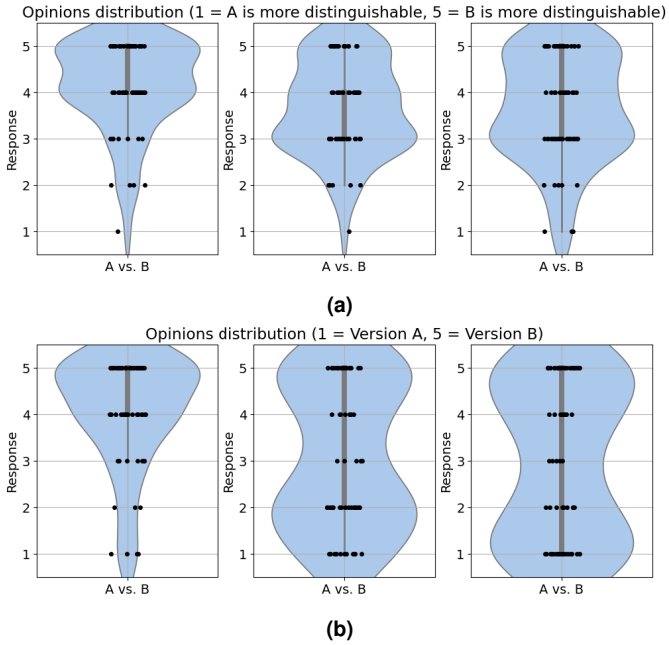


Figure 13. Opinion distributions for S1 (a) and S2 (b). S1 shows responses more concentrated towards version B (Likert scale of 5), suggesting it is more distinguishable at a distance. S2 shows more subjective responses regarding post-processing, being evenly split between versions A (no outlines) and B (with outlines) in terms of aesthetics, except for question 1b, which still showed a preference for version B in terms of readability.

either other or did not inform their gender; 86.9% were between 18 and 28 years old, and 13.1% over 28 years old. Of these, 43 identified as specialists, from which: 70.5% answered "Computing" and 31.9% "Artist", 3 participants having answered true to both.

A Wilcoxon signed-rank test was conducted to compare responses between conditions A and B on a 5-point ordinal scale. The results for section S1, Figure 13a, indicated a statistically significant difference ($W = 100.0$, 88.0, 140.0 $p < 0.000001$, 0.0001, 0.001, for questions 1, 2 and 3 respectively), **suggesting that elements in the scene are more distinguishable at a distance in version B** (Pixelat3D with priority mapping and acute angle texture modifications) than in version A (default shader).

The test was also conducted on section S2, Figure 13b, with results indicating a statistically significant difference only for question 1 ($W = 190.5$ $p < 0.000001$), suggesting that while elements in the scene are more distinguishable in version B (Pixelat3D with priority mapping, acute angle texture modifications and post-processed outlines) than in version A (Pixelat3D with only priority mapping and acute angle texture modifications), the aesthetic improvement of the outlines is entirely subjective.

For section S3, a chi-square goodness-of-fit test was conducted to determine whether responses were equally distributed. Although most participants reported a preference for a subtle contrast, the difference was not significant ($p > 0.2$); this further cements the previous test's notion that the preference for outlines is entirely subjective.

Section S4 compares the effects of the engine's default rendering (A) with Pixelat3D's (B) on different textures at a distance. In Questions 1 and 2: 90.2% and

91.8% chose B, respectively. For question 3, when asked to justify, some participants complained about flickering pixels on A and some stated greater legibility, stability, or consistency on B. For question 4, a chi-square goodness-of-fit test was performed to assess whether responses were evenly distributed across the four categories. The results were statistically significant χ^2 (3) = 11.55, p = 0.0091, indicating that the responses were not evenly distributed. Although most of the participants (23) believe that there are visual inconsistencies in both versions, 19 said that they occur only in version A, while 5 mentioned that they occur only in version B. The results of Question 5 also indicated that the responses were not evenly distributed, $\chi^2(3) = 21.48$, $p < 0.00001$, with most of the participants (18) finding the inconsistencies greater in version A, 4 believing them equal in both versions, and 1 finding version B more inconsistent. **With this we can say that, at a distance, the use of Pixelat3D significantly improves the readability and definition of textures and, although there is flickering, it gets noticeably reduced with its use**.

Section S5 keeps the same objective as section S4, now at a closer distance. In questions 1 and 2: 72.1% and 70.5% chose B, respectively. In question 3, when asked to justify, some participants stated smoother and more consistent movement on B in comparison to A, as well as reduced flickering, especially on the higher noise textures, though still mentioning some degree of it being present. Questions 4 and 5 also conducted chi-square goodness-of-fit tests; the results showed that, although there is still a preference for version B, the effects are less pronounced and/or noticeable at closer distances. With this, it can be said that the effect of Pixelat3D on both readability and aesthetics is slightly decreased at closer ranges when compared to longer distances. This was expected, as the objects get closer to the camera, the priority map use is decreased. When asked for feedback, most answers suggested changes related to post-processing, reducing or removing the outlines, as well as the noticeable difference in color brightness.

6. Conclusion

Recreating an effect similar to the style of pixel art is challenging for 3D scenes. The use of the shader showed how prioritized features manage to stay prominent in different distances, angles, and textures, improving on existing techniques that focus solely on detailing through topology and showing mostly satisfactory results; issues such as the flickering would require a custom rescaling algorithm that treats sub-pixels. In future works, we plan to add shading effects through dithering and better explore the benefits of topology through displacement mapping, as well as a performance analysis.

7. Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and partially by CNPq–Brazil (grant 421962/2023-2), and by the Advanced Knowledge Center in Immersive Technologies (AKCIT), with financial resources from the PPI IoT/Manufatura 4.0 / PPI HardwareBR of the MCTI grant number 057/2023, signed with EMBRAPII, and partially supported by Fundação para a Ciência e a Tecnologia (FCT) through the project “HIITS: Haptics for Immersive Interactive Training Simulators” (reference 2024.16976.PEX).

Paper webpage

`https://www.inf.ufrgs.br/vislab/sbgames26-pixelat3D`

References

Barla, P., Thollot, J., e Markosian, L. (2006). X-toon: An extended toon shader. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 127–132.

Coutinho, F. e Chaimowicz, L. (2022). Generating pixel art character sprites using gans. In *2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 1–6. IEEE.

Coutinho, F. e Chaimowicz, L. (2024). Pixel art character generation as an image-to-image translation problem using gans. *Graphical Models*, 132:101213.

Han, C., Wen, Q., He, S., Zhu, Q., Tan, Y., Han, G., e Wong, T.-T. (2018). Deep unsupervised pixelization. In *SIGGRAPH Asia 2018 Technical Papers on - SIGGRAPH Asia '18*, volume 37, pages 1–11, New York, New York, USA. ACM Press.

Johansen, R. S. (2025). Dither3d: Surface-stable fractal dithering.

King, K. (2018). three.js webgl - post processing - pixelation.

Kuo, M.-H., Yang, Y.-L., e Chu, H.-K. (2016). Feature-aware pixel art animation. *Comput. Graph. Forum*, 35(7):411–420.

Moreira, R. D., Coutinho, F., e Chaimowicz, L. (2022). Analysis and compilation of normal map generation techniques for pixel art. In *2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 1–6. IEEE.

Pope, L. (2017). Return of the obra dinn [releasing oct 18].

Recordá Illas, R. (2024). Developing 3d pixel art rendering. B.S. thesis, Universitat Politècnica de Catalunya.

Saravanan, A. e Guzdial, M. (2022). Pixel vq-vaes for improved pixel art representation.

Serpa, Y. R. e Rodrigues, M. A. F. (2019). Towards machine-learning assisted asset generation for games: a study on pixel art sprite sheets. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 182–191. IEEE.

Silber, D. (2015). *Pixel art for game developers*. CRC Press.

Wu, Z., Chai, L., Zhao, N., Deng, B., Liu, Y., Wen, Q., Wang, J., e He, S. (2022). Make your own sprites. *ACM Trans. Graph.*, 41(6):1–16.