RETRÔ: Simulando Pixel Art Através de Técnicas de Renderização Não-Fotorrealista a Partir de Modelos 3D

RETRO: Simulating Pixel Art Through Non-Photorealistic Rendering Techniques From 3D Models

Wesley Santos[1], Rafael F. Ivo[1], Jacilane de H. Rabelo[1]

[1]Universidade Federal do Ceará (UFC) – Russas, CE – Brasil

wesleysantos@alu.ufc.br, {rafaelivo,jacilane.rabelo}@ufc.br

Abstract. *Pixel art was widely used in games from the early 1970s to the 1990s, not just as an art style, but rather the only way to represent images in electronic games due to technological limitations. Even with the evolution of computer graphics, pixel art transcends its origins and establishes itself as an art style. However, creating pixel art is a slow, laborious and almost entirely manual process. Based on this context, this work aims to simulate pixel art from a 3D model, using non-photorealistic rendering techniques. The work identifies that the automatic method does not replace the artist's work, but has the purpose of accelerating the pixel art construction process.*
Keywords: *Non-Photorealistic Rendering, 3D Models, Pixel Art, Digital Games.*

Resumo. *O pixel art foi muito utilizado em jogos no início de 1970 à 1990, não apenas como um estilo de arte, mas sim a única maneira de representar imagens em jogos eletrônicos devido às limitações tecnológicas. Mesmo com a evolução da computação gráfica, o pixel art transcende suas origens e se estabelece como um estilo de arte. Entretanto, construir pixel art é um processo lento, trabalhoso e quase integralmente manual. Partindo desse contexto, este trabalho tem o propósito de simular um pixel art a partir de um modelo 3D, utilizando técnicas de renderização não-fotorrealistas. O trabalho identifica que o método automático não substitui o trabalho do artista, mas tem o propósito de acelerar o processo de construção do pixel art.*
Palavras-chave: *Renderização Não-Fotorrealista, Modelos 3D, Pixel Art, Jogos Digitais.*

1. Introdução

Pixel art é uma forma de expressão artística que utiliza *pixels* individuais para criar imagens ou animações com uma estética digital retrô. Há uma quantidade limitada de trabalhos acadêmicos que definem *pixel art*, pois o termo varia até mesmo entre aqueles que a desenvolvem [Samuelson 2020]. *Pixel art* trata-se de uma forma de arte que é feita a partir de vários pontos na tela denominados *pixels* [Vahl e de Albuquerque 2022].

No passado, com os recursos tecnológicos limitados, o uso dos *pixels* exigia cuidado e criatividade para buscar soluções otimizadas [Vahl e de Albuquerque 2022]. Essa técnica começou a ser utilizada em jogos eletrônicos durante o início dos anos 1970 e 1980, devido à limitação de processamento dos consoles daquela época, pois não tinham tecnologia o suficiente para processar imagens feitas a mão ou fotografia. Isso fez com que o *pixel art* se tornasse a estética dominante durante a produção de jogos até os anos 1990, quando a evolução da tecnologia permitiu o uso de modelos 3D e fotografias [Lima 2023]. Vale ressaltar que, até então, o *pixel art* não era um estilo de

arte, mas sim a única maneira de representar imagens nos jogos eletrônicos devido às limitações tecnológicas dos *hardwares* [Alencar 2017].

Hoje, o *pixel art* é mais comumente encontrado no mundo dos jogos independentes. Como criadores independentes trabalham com orçamentos muito menores, o *pixel art* é tido como uma alternativa viável e barata para trabalhar [Samuelson 2020], pelos seguintes aspectos: (a) estilo minimalista que reduz a quantidade de detalhes necessários para criar uma imagem; (b) requisitos técnicos mínimos, pois o *pixel art* pode ser criado até mesmo em dispositivos básicos, como computadores antigos ou telefones celulares; e, (c) tamanho de arquivo pequeno, devido à sua natureza baseada em *pixels*, as imagens de *pixel art* geralmente têm um tamanho de arquivo pequeno em comparação com outros formatos de imagem digital.

Por outro lado, é possível observar que o *pixel art* é um processo que demanda tempo e esforço considerável. A natureza dos detalhes da criação *pixel* a *pixel* resulta em um trabalho que pode ser descrito como lento e trabalhoso. A maior parte do processo é realizada manualmente, o que significa que cada *pixel* deve ser cuidadosamente posicionado para alcançar o efeito desejado [Silber 2015].

A quantidade de estilos artísticos presentes em jogos que fazem uso de *pixel art* é vasta. Entretanto, algumas características são transversais à maioria dos estilos. Sendo uma das mais marcantes, a limitação de quantidades de cores, uma limitação decorrente do *hardware* gráfico dos anos 80-90. Essa limitação, tornou necessária a criação de jogos com cores fortes e saturadas, assim como variações abruptas de tonalidades, destacando regiões iluminadas e sombreadas. Apesar dos atuais jogos independentes não possuírem tais limitações, grande parte dos estilos existentes seguem parcialmente esta limitação buscando uma correlação estética com estes jogos mais antigos.

Apesar do *pixel art* ser um estilo marcado, em sua criação, pelas limitações técnicas, ele continua a ser aplicado em diversos estilos de jogos de diferentes gêneros, desde os mais realistas aos mais cartunescos (Figura 1). Este trabalho é focado na criação de *sprites* mais próximos do estilo deste último grupo (cartunescos), ao utilizar a técnica de *toon shading* [Gooch 2001], onde variações de iluminação suaves são substituídas por linhas destacadas de regiões claras e escuras, e ao adicionar linhas de silhueta e contornos aos *sprites* automaticamente, similar ao jogo da esquerda ilustrado na Figura 1.

Figura 1. Jogos de diferentes gêneros e estilos. Da esquerda para direita: Megaman X[1] (1993) e Dandara: Trials of Fear Edition[2] (2018).

[1] https://abrir.link/TbKEb
[2] https://store.steampowered.com/app/612390/Dandara_Trials_of_Fear_Edition/

O algoritmo proposto seguirá os passos apresentados na Figura 2. Da esquerda para a direita: (i) um objeto 3D é utilizado como entrada; (ii) em seguida é aplicado iluminação local (Phong e *toon shading*) para simular a limitação de cores; (iv) dando seguimento é aplicado pós-processamento de imagem para detecção de silhuetas e aplicação do arredondamento de coordenadas da textura, para simular uma imagem de baixa resolução;e, (v) por fim, o resultado final, uma aproximação de um *sprite 2D*. Este algoritmo não tem a intenção de substituir o papel do artista, mas pode ser empregado para otimizar o processo de criação artística.

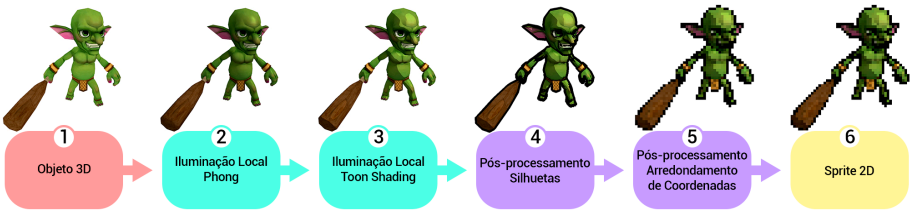

Figura 2. Processo de construção de *sprite* a partir do modelo 3D Goblin01[3]

Nas demais seções, o artigo apresenta a seguinte estrutura: seção 2 apresenta os trabalhos relacionados; seção 3 detalha a metodologia adotada, com o processo de geração de *sprite*; seção 4 expõe os resultados obtidos e sua análise; e a seção 5 é dedicada à conclusão e trabalhos futuros.

2. Trabalhos Relacionados

Para este trabalho foram selecionados alguns estudos e análises voltados para o estilo *pixel art*. Há um grande acervo de trabalhos e autores que abordam técnicas para pixelizar imagens assim como temas que categorizam padrões de cores e estilo de bordas. Nesta seção estão os trabalhos que mais se destacaram para esta pesquisa.

Em seu trabalho, Lei *et al.* (2024) busca mapear as possibilidades de criação de *pixel art*. Os artistas podem criar *pixel art* manualmente ou, alternativamente, utilizar softwares específicos, como o Photoshop. Além disso, o estudo destaca ferramentas como o *PixelME*, que permite aos usuários gerar *pixel art* fazendo upload de imagens. Outra maneira é utilizando algoritmo de geração automática, destacando que abordagens baseadas em Rede Generativa Adversarial (GAN) é não controlada, as imagens de *pixels* geradas por redes neurais não atendem aos requisitos estéticos básicos de *pixel art*, como uma largura de *pixel* única e uma disposição regular.

Vale destacar que, em seu método, Lei *et al.* (2024), com referências de Shang e Wong (2021), busca otimizar a técnica de pixelização de imagens por meio do algoritmo AOP (*Automatical Optimization Pixelization*). Esse algoritmo, um processo iterativo, começa com a divisão da imagem original em blocos para segmentar áreas distintas de contorno e não contorno. Posteriormente, essas áreas são tratadas individualmente em múltiplas iterações que incluem extração de cores primárias, preenchimento de blocos de *pixels* e ajustes finais de posicionamento. O método não só melhora a eficiência do processo de pixelização, mas também a qualidade visual do resultado final, garantindo que as imagens pixelizadas mantenham a fidelidade estética do *pixel art*.

[3] https://assetstore.unity.com/packages/3d/characters/goblin01-188119

Shang e Wong (2021) propõem um algoritmo automático de pixelização para imagens de retrato, fundamentado nos algoritmos *Simple Linear Iterative Clustering* (SLIC) e *Fuzzy Iterative Self-Organizing Data Analysis* (FISODATA). Diversas melhorias foram implementadas para aprimorar as imagens, incluindo a reordenação de superpixels, a detecção de objetos em cascata, a filtragem bilateral gaussiana e o realce de bordas de características. O algoritmo demonstra ser especialmente eficaz na pixelização de imagens de retrato naturais com transições de cor suaves. Além disso, o algoritmo não apenas atende aos requisitos de bordas nítidas e cores limitadas das imagens de *pixels*, mas também determina automaticamente a largura e altura da imagem de saída, assim como o tamanho da paleta, através da análise da imagem.

Vahl e de Albuquerque (2022) realizam uma pesquisa documental que tem o objetivo de mapear os diferentes estilos de *pixel art* e suas aplicações em jogos digitais com visão aérea. Em suas considerações, os autores destacam características importantes quanto a esse estilo de arte, como a paleta de cores utilizada pelos jogos variam a depender da atmosfera do jogo. Além disso, por vezes é utilizado contornos monocromáticos em objetos e personagens para destacá-los do cenário, de forma que o jogador identifique os personagens e os objetos que são possíveis de interagir.

Tarasconi e Santos (2021) fazem uma análise documental de cinco jogos com estilo *pixel art*, com o intuito de identificar padrões de produção de ícones dentre esses jogos. O artigo obtém como resultado, padrões na confecção de bordas, ícones e saturação de elementos artísticos em comum dos jogos. Todavia, alguns jogos não seguem o padrão em certos tópicos da análise. Os autores enfatizam que há um certo padrão de produção nos jogos analisados em seu trabalho. Nele pode destacar os aspectos de padrões de bordas de ícones, saturação e tonalidades usadas para representá-los. Porém, devido ao pequeno acervo de jogos utilizado em sua pesquisa, não foram identificados padrões nos aspectos e técnicas únicas desse estilo.

Os estudos elaborados por Shang e Wong (2021) e Lei *et al*. (2024) oferecem abordagens e soluções para a criação e otimização de *pixel art*, mas a partir de uma imagem estática, sem a possibilidade de alteração de posicionamento ou rotação dos elementos em cena (Figura 3). O método proposto neste trabalho se diferencia ao utilizar técnicas de renderização não-fotorrealista combinadas com pós-processamento de imagens para criar uma representação em *pixel art* de objetos tridimensionais utilizando a *game engine Unity*[4]. Isso possibilita que o objeto 3D seja ajustado para atingir o resultado que mais se adequa às necessidades do projeto, permitindo ajustes finos, como posição, rotação e iluminação.

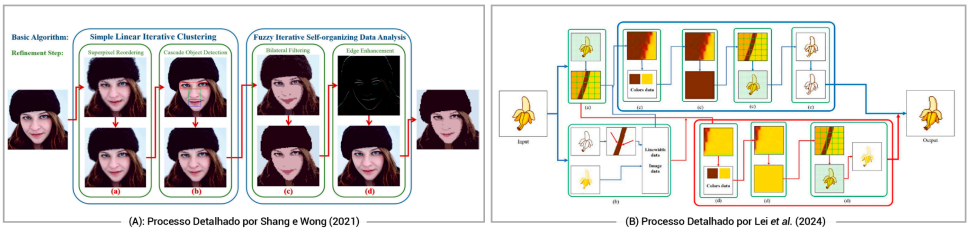

Figura 3. Processos detalhados dos algoritmos de pixelização estudados.

[4] https://unity.com/pt

3. Processo de Geração de *Sprites*

A princípio, os autores optaram por escolher a *game engine Unity* pelas suas ferramentas para programar *shaders* de forma visual. Descrevendo detalhadamente o processo, o modelo 3D serve como objeto de entrada do algoritmo e é importado para o ambiente de desenvolvimento, como o *Unity*, onde será manipulado e renderizado de acordo com as técnicas propostas neste estudo. É importante ressaltar que o modelo 3D deve consistir em uma malha poligonal (ou triangular) descrita em algum formato específico, como *.obj* ou *.fbx,* e incluir não apenas a malha, mas também a textura e o mapeamento da textura. Já no ambiente de desenvolvimento, ajustes do posicionamento e características de fontes de luz, bem como posicionamento da câmera.

Após o modelo 3D ser o objeto de entrada, o próximo passo envolve a aplicação da iluminação de Phong é um modelo de iluminação local e embora não seja fisicamente correto ele produz resultados suficientemente adequados para a percepção de objetos iluminados. A cor final obtida com este modelo é composta pela combinação de três componentes: ambiente, difusa e especular. A componente ambiente simula a dispersão global da luz no ambiente e garante que o objeto seja minimamente visível mesmo em área de sombra. A componente difusa simula a reflexão da luz em todas as direções, capturando a cor do objeto sob luz direta, enquanto a componente especular simula o brilho ou o reflexo da luz, conferindo às superfícies um aspecto brilhante e contribuindo para a percepção de sua textura e forma [Bui-Tuong 1975]. A implementação de *toon shading* presente neste trabalho realiza uma limiarização das componentes difusa e especular do modelo de iluminação de Phong. Na Figura 4, são apresentados os shaders implementados na Unity. Nas Figuras 4 (A) e (C), destaca-se a implementação das componentes difusa e especular do modelo de Phong, respectivamente. Nas Figuras 4 (B) e (D) são apresentadas as modificações para a obtenção do *toon shading*.

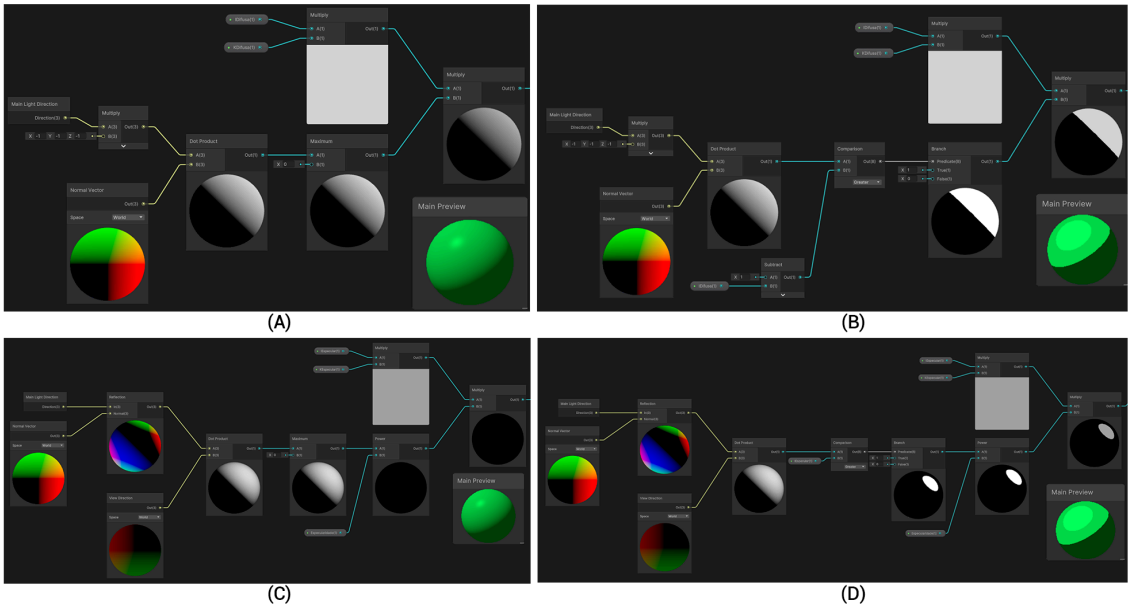

Figura 4. *Toon shading* a partir da iluminação de Phong (*shader graph*).

Na Figura 4, são apresentados os *shaders* implementados na *Unity*. À esquerda, um trecho da implementação de Phong destaca a iluminação difusa *(A)* e especular *(C)*, enquanto à direita é apresentada a modificação, a partir de Phong, para a obtenção do *toon shading*. Em *(B)* a área iluminada é limitada pela intensidade de iluminação difusa da luz e, de forma semelhante em *(D),* é calculada a iluminação especular.

Dando seguimento, será utilizada uma técnica baseada no filtro de Sobel aplicado ao *depth buffer* e ao *normal buffer* do modelo 3D. Esta abordagem, estudada por Pinheiro (2010), teve como base o trabalho de Decaudin (1996). O filtro de Sobel é uma técnica comumente empregada em processamento de imagem para detecção de bordas, que calcula gradientes de intensidade em uma imagem para identificar transições abruptas de cor ou intensidade. Para encontrar esse tipo de variação, um filtro de Sobel geralmente é utilizado, porém qualquer outro filtro de detecção de arestas é adequado para essa aplicação [Pinheiro 2010].

Ao aplicar o filtro de Sobel aos *buffers* de profundidade e normais do modelo 3D, é possível realçar as bordas do objeto, destacando suas formas e contornos. Essa é uma solução para a obtenção de silhuetas nítidas e contornos definidos, características do *pixel art*. Na Figura 5 é representado o processo de detecção de silhuetas em um modelo 3D disponibilizado gratuitamente pela *Unity Asset Store*[5].

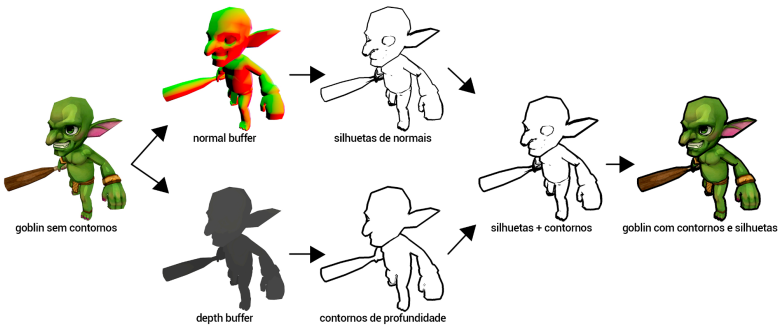

Figura 5. Processo de detecção de silhuetas e contornos.

Na etapa que será simulado o *pixel art*, busca-se emular a estética do *pixel art* através da redução da resolução da imagem. Para isso, as coordenadas de textura *(u, v)* são multiplicadas pela escala de *pixel*, e então é aplicada a função de arredondamento para baixo *(floor)* para obter apenas a parte inteira dessas coordenadas. Em seguida, as coordenadas são divididas novamente pela escala de *pixel*. A Figura 6 apresenta essa implementação através do *shader graph* (programação visual) na *Unity*.

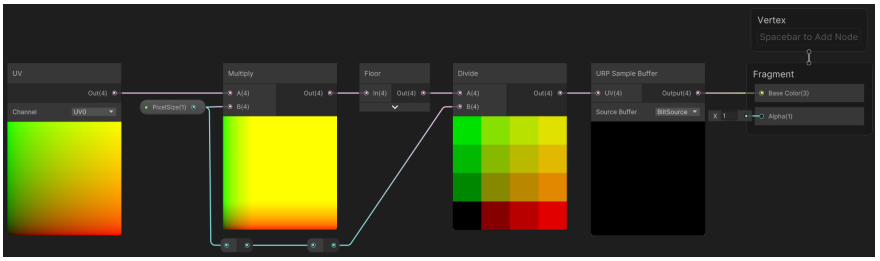

Figura 6. Redução de resolução (implementado no *Shader Graph* da *Unity*).

[5] https://assetstore.unity.com/

Este método de redução de resolução se alinha com os princípios descritos por Gonzalez e Woods (2010) em "Processamento digital de imagens" , onde técnicas de redimensionamento são utilizadas para modificar a escala das imagens enquanto se preserva ou se adapta a qualidade visual original. O arredondamento de coordenada para simular *pixel art*, embora não diretamente discutido por Gonzalez e Woods, pode ser entendido como uma forma de redimensionamento onde a resolução é deliberadamente reduzida para enfatizar cada *pixel* individualmente.

Com esse processo de arredondamento de coordenadas da textura, é possível renderizar uma imagem que se aproxima da estética do *pixel art*. Vale ressaltar que o simples ato de renderizar uma imagem em baixa resolução não a caracteriza como *pixel art*. Por isso, é interessante extrair as características desse estilo e estudá-las antes de simplesmente diminuir a resolução de uma imagem.

4. Testes e Resultados

Nesta seção, testes foram realizados utilizando modelos 3D gratuitos disponíveis na plataforma *Unity Asset Store*. Foram realizados testes para avaliar a eficácia e a qualidade do método de pixelização proposto. Os testes foram conduzidos utilizando uma variedade de modelos 3D de diferentes formas e quantidade de polígonos, em que cada modelo foi submetido ao processo de pixelização proposto.

Na Figura 7, é apresentado uma série de renderizações feitas pelo algoritmo, divididas em 5 colunas, em que: (*A*) utilizando o método de Phong; (*B*) utilizando o *toon shading*; (*C*) *pixel art* em uma grade 32x32; (*D*) *pixel art* em uma grade 64x64; (*E*) *pixel art* em uma grade 128x128. As colunas (*A*) e (*B*) são renderizações para obter os detalhes do objetos 3D e definir a quantidade de cores, enquanto as colunas (*C*), (*D*) e (*E*) são resultantes do algoritmo com diferentes taxas de *pixel*.

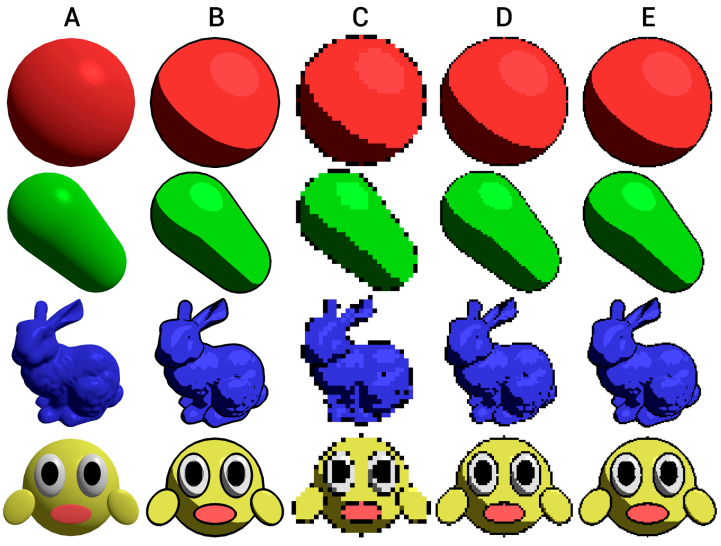

Figura 7. Testes do algoritmo em diferentes objetos 3D.

É possível observar que, a partir dos testes feitos apresentados na Figura 7, dependendo da complexidade do objeto 3D, informações detalhadas e curvas sutis podem ser perdidas durante o processo de pixelização. Enquanto objetos simples, como a esfera na cor vermelha, mantêm sua forma básica, modelos mais complexos, como o

coelho na cor azul, podem sofrer uma perda significativa de detalhes. Essa variação destaca a necessidade de considerar a complexidade dos objetos 3D ao aplicar a técnica.

Uma característica importante do *pixel art* é que todos os *pixels* percebidos em uma imagem foram colocados ali propositalmente. Não quer dizer, por outro lado, que cada *pixel* é posto um por um na imagem, mas sim que há um pensamento por trás do arranjo desses elementos [Silber 2015]. Seguindo essa interpretação, na Figura 8, à esquerda, é apresentado um *pixel art* renderizado pelo algoritmo, enquanto à direita é exibido o mesmo *pixel art* ajustado manualmente pelo artista. Dessa forma, é possível utilizar o *pixel art* como base e realizar ajustes manuais mais refinados, procurando resgatar informações ou detalhes que possam ter sido perdidos durante o processo.

Figura 8. Ajuste manual em um *pixel art* renderizado pelo algoritmo.

Durante o ajuste manual, foi observado que, embora a imagem tenha sido dividida em blocos, a distribuição dos *pixels* não se mostrou uniforme a ponto de não apresentar silhuetas bem definidas, conforme os resultados obtidos por Lei *et al*. (2024) e Shang e Wong (2021). Isso é evidente nos testes realizados na Figura 7, em que a qualidade das silhuetas diminui à medida que o tamanho da matriz de *pixels* diminui, como ilustrado na coluna (*C*) representando uma grade de 32x32. Esse resultado era esperado, uma vez que a imagem não foi verdadeiramente reduzida em escala de pixel, mas sim teve suas coordenadas de textura arredondadas para simular a baixa resolução.

Na Figura 9, é possível observar que, ao reduzir o tamanho da grade de *pixels*, os detalhes das silhuetas podem se tornar mais ou menos visíveis. Silhuetas mais finas podem perder alguns detalhes após a pixelização em uma grade 32x32, mas podem apresentar mais detalhes em uma grade 64x64. Por outro lado, silhuetas mais espessas tendem a reter mais detalhes quando apresentada em uma grade 32x32, embora possam não apresentar tanta clareza quanto as silhuetas mais finas. Essa análise destaca a complexidade da interação entre a espessura da silhueta e o tamanho da grade de *pixels* na pixelização e ressalta a importância de considerar esses fatores.

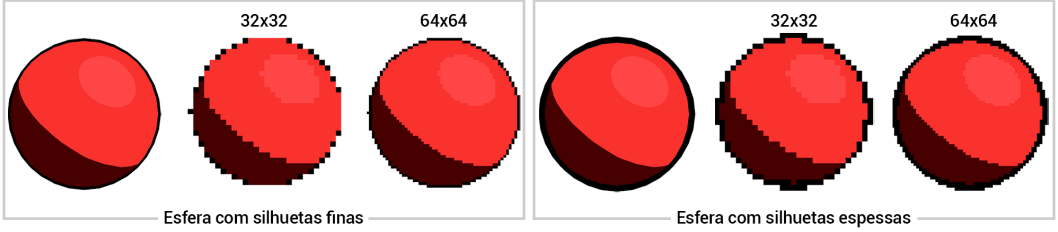

Figura 9. Influência da espessura da silhueta no resultado final.

Diferente dos métodos utilizados por Lei *et al*. (2024) e Shang e Wong (2021), que segmentam a imagem de entrada em blocos (ou uma grade NxN) e executam uma

série de processos iterativos para otimizar a pixelização (Figura 10). O método presente neste trabalho não busca uma abordagem predominantemente detalhada. Em vez disso, o objetivo principal é renderizar uma imagem que o artista possa utilizar como ponto de partida e acelerar a produção de *pixel art*. Isso significa que o processo de pixelização é simplificado, focando na preservação das características essenciais do objeto 3D original e uma aproximação visual do estilo *pixel art*.

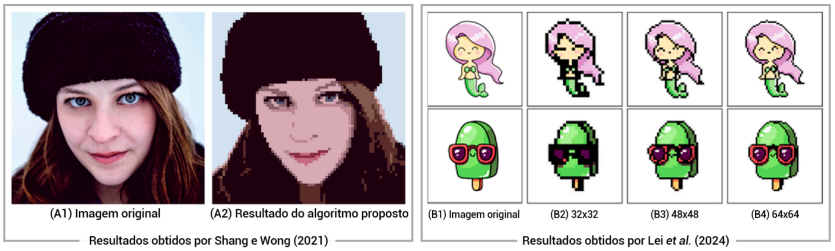

Figura 10. Resultados obtidos pelos trabalhos anteriores.

Mesmo diante dessa limitação, o algoritmo pode ser empregado na geração de *sprite sheets*. Ao contar com um modelo 3D animado, é possível selecionar sua posição, um ponto específico na animação (*Cycle Offset*, destacado em vermelho), e aplicar o algoritmo para produzir uma sequência de imagens (Figura 11). Essas imagens são geradas a partir de *frames* específicos da animação do modelo 3D. As imagens podem então ser compiladas em um *sprite sheet*, diminuindo consideravelmente a carga de trabalho do artista. Na Figura 12, é exemplificado um *sprite sheet* elaborado a partir de uma sequência de imagens renderizadas pelo algoritmo a partir de um modelo 3D.

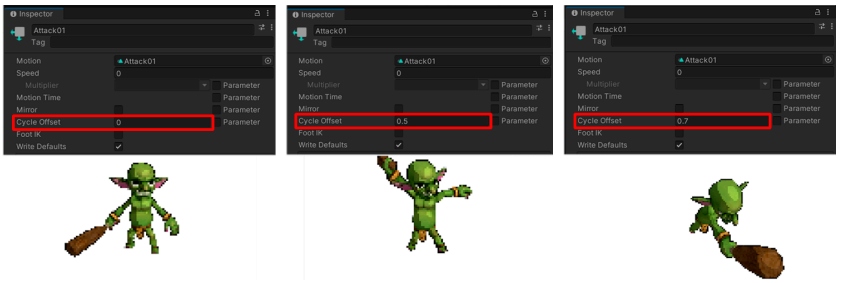

Figura 11. Selecionando pontos específicos na animação pelo *Cycle Offset (entre 0 e 1)*.

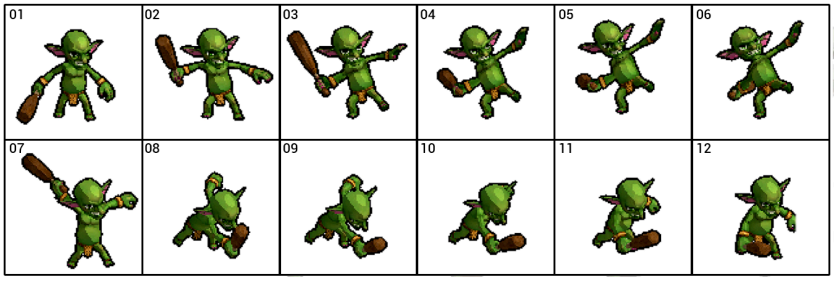

Figura 12. *Sprite sheet* de imagens geradas com base no modelo 3D.

Em resumo, os testes realizados demonstraram a eficácia do algoritmo proposto na geração de *pixel art* a partir de modelos 3D. Apesar das limitações observadas, como a não uniformidade na distribuição dos *pixels* e a necessidade ajuste manual nas silhuetas e detalhes internos da imagem, o algoritmo mostrou-se capaz de preservar as

características essenciais dos modelos originais, proporcionando uma representação pixelizada estéticamente agradável. Além disso, a possibilidade de utilizar o algoritmo para gerar *sprite sheets* representa uma alternativa eficaz para simplificar o processo de criação de *sprites* para jogos digitais.

5. Conclusão e Trabalhos Futuros

Neste trabalho, foram exploradas uma série de técnicas de renderização não-fotorrealista com o objetivo de criar uma representação em *pixel art* de objetos tridimensionais. A base para as técnicas foi o modelo de iluminação de Phong, sobre o qual foram aplicadas renderização em *cartoon* (*toon shading*), e pós-processamento de imagens, utilizado para detecção de contornos e silhuetas, além de arredondamento de coordenadas da textura para simular uma imagem de baixa resolução. Essas abordagens permitiram a criação de visuais que remetem ao estilo artístico de *pixel art*, preservando as dimensões dos objetos.

Apesar de alguns artefatos presentes na imagem final, nos testes apresentados na seção 5, o *pixel art* gerado pode ser utilizado como um ponto de partida para artistas e desenvolvedores de jogos. Este algoritmo pode se mostrar útil para aqueles que possuem menor habilidade ou pouca experiência em criar arte 2D no estilo *pixel art*, oferecendo uma solução semi-automatizada para gerar assets visuais retrô. Além disso, é possível agilizar o processo de criação de gráficos para jogos (como *sprite sheets*), permitindo que os criadores foquem mais em aspectos de *design* e *gameplay*.

Durante a análise dos resultados, foi possível perceber que o modelo 3D e seu posicionamento relativo à câmera afetam diretamente o *pixel art* gerado. Para trabalhos futuros seria promissor identificar que características um modelo 3D, como nível de detalhe da malha e pose com relação a câmera, deve possuir e como alterá-lo objetivando alcançar melhores resultados de *sprite sheets*. Outra observação realizada durante a análise foi a falha nas linhas das silhuetas ao diminuir a grade de *pixel*. Em pesquisas subsequentes, o problema poderia ser mitigado ao realizar a detecção de silhuetas em espaço de imagem, após a imagem gerada no *toon shading*.

Estas abordagens podem ajudar a manter as linhas de silhueta mais coesas e fechadas. No entanto, essas ideias não foram implementadas neste trabalho devido à necessidade de compreender completamente as implicações de calcular uma silhueta em uma imagem de baixa resolução. Embora a linha do contorno possa ser facilmente manipulada, as linhas internas podem escurecer o objeto, removendo as nuances de cores obtidas através do *toon shading*.

Referências

Alencar, F. H. B. M. (2017). “Pixel art & Low Poly art: catalisação criativa e a poética da nostalgia”.

Bui-Tuong, P. (1975). “Illumination for computer generated pictures”. In: CACM.

Decaudin, P. (1996). “Cartoon-looking rendering of 3D-scenes”. In: Syntim Project Inria, v. 6, n. 4.

Gonzalez, R. C. and Woods, R. E. (2010). “Processamento digital de imagens”. In: São Paulo: Pearson Prentice Hall, 3. ed.

Gooch, B. and Gooch, A. (2001). “Non-Photorealistic Rendering”. In: A. K. Peters, Ltd., USA.

Lei, P. Xu, S. and Zhang, S. (2024). “An art-oriented pixelation method for cartoon images”. In: The Visual Computer, v. 40, n. 1, p. 27-39.

Lima, E. C. L. (2023). “Pixel Art: Identidade visual e elementos de cenário para um jogo digital baseados em símbolos manauara”.

Pinheiro, D. F. (2010). “Processo de detecção e síntese de contornos no cel shading utilizando filtros digitais”. CIn - UFPE.

Samuelson, G. (2020). “Pixel art-The Medium of Limitation: A qualitative study on how experienced artists perceive the relationship between restrictions and creativity”.

Shang, Y. and Wong, H. (2021). “Automatic portrait image pixelization”. Computers & Graphics, v. 95, p. 47-59.

Silber, D. (2015). “Pixel art for game developers”. In: CRC Press.

Tarasconi, D. M. and Dos Santos, M. A. S. (2021). “Padrões de Produção de Ícones e Símbolos em Jogos com Estilo Pixelart”. In: Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital. SBC. p. 297-300.

Vahl, P. H. M. and De Albuquerque, R. M. (2022). “Mapeamento de estilos de pixel art aplicados a jogos digitais com visão aérea”. In: Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital. SBC. p. 228-237.