Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Monografia em Sistemas de Informação

Editor de *Pixel Art* Integrado a Redes Adversárias Generativas: Requisitos, Implementação e Otimização de Modelos

Lucas Gabriel Silveira Chaves

Belo Horizonte, MG
2025

Lucas Gabriel Silveira Chaves

Editor de *Pixel Art* Integrado a Redes Adversárias Generativas: Requisitos, Implementação e Otimização de Modelos

Relatório de Monografia baseada em pesquisa científica, apresentada como requisito para a conclusão do curso de graduação em Sistemas de Informação da Universidade Federal de Minas Gerais.

Orientador: Luiz Chaimowicz

Coorientador: Flávio Coutinho

Belo Horizonte, MG
2025

Agradecimentos

Agradeço aos meus pais, Edna e Mauro, por todos os esforços que dedicaram à criação e à educação de seus filhos, e por todo o apoio ao longo dos anos; aos meus irmãos, Michelle e Maicon; à minha avó Lazara, por todo carinho, e também a toda a minha família; à minha companheira Carolina, por todo o suporte e companheirismo que também me permitiram chegar aqui; aos meus orientadores, Luiz e Flávio, por todo o aprendizado durante o trabalho; e, por fim, a todos os meus amigos que me acompanharam nessa jornada: os nomes são muitos, mas a importância de cada um é única.

Belo Horizonte, MG
2025

Resumo

Muitas vezes, artistas de jogos se deparam com tarefas mais repetitivas e mecânicas, que ocupam esforços que poderiam estar sendo usados em atividades criativas. Sendo assim, este trabalho se propõe a investigar um editor *web* de *Pixel Art* integrado a modelos de aprendizado profundo utilizados para geração de sugestões de imagens de personagens. Nossa metodologia envolve a aplicação de pesquisa de usuário para levantamento de requisitos e validação de hipóteses, da implementação de novas ferramentas básicas de edição, e uma investigação acerca de possíveis otimizações nos modelos geradores utilizados pela interface, baseados nas arquiteturas StarGAN e CollaGAN. Os resultados da nossa pesquisa de usuário demonstram um conjunto mínimo de ferramentas consideradas de alta importância para a edição de *Pixel Art* e sugerem que artistas de jogos reconhecem, em suas atividades diárias, espaços para integrações com modelos de redes profundas, com ressalvas acerca da natureza da interação e com foco em manter o protagonismo do usuário nas decisões e nas artes geradas. Por sua vez, as técnicas de otimização investigadas nos modelos integrados ao editor possibilitaram a obtenção de melhores resultados em relação aos modelos de base nos cenários avaliados. Concluímos que o uso de editores como o proposto, integrado a modelos generativos via interação mista com usuário, pode apoiar artistas de jogos em suas tarefas. Mais ainda, as investigações realizadas nos modelos mostram que técnicas de quantização diferenciável e de perda de cobertura de paleta de cores favorecem a convergência e a sintetização de imagens em modelos StarGAN, enquanto a ampliação de imagens como pré-processamento possibilita resultados análogos em modelos da arquitetura CollaGAN, considerando os hiperparâmetros e as configurações utilizadas.

Palavras-chave: *Pixel Art*; geração de *sprites*; redes adversárias generativas; inteligência artificial; interação humano-computador.

Sumário

1 Introdução

Durante o processo de desenvolvimento de jogos é comum que artistas se deparem com tarefas repetitivas e mecânicas, que acabam por consumir o tempo que poderia ser dedicado a tarefas intelectuais, complexas e criativas. No contexto de jogos em *Pixel Art*, por exemplo, a movimentação dos personagens é realizada por meio de animação em célula, em que cada quadro de movimento é um *sprite* (i.e. uma imagem do personagem em questão em uma certa pose). Devido a esta característica e ao nível de detalhamento das texturas de jogos 2D atuais, por vezes torna-se trabalhoso e repetitivo corrigir, alterar ou gerar os vários *sprites* de um mesmo personagem (Silber 2015).

Nesses cenários, a aplicação de técnicas de aprendizado profundo em tarefas relacionadas a processos criativos e artísticos, como a criação de *sprites* de personagens, de cenários ou de objetos no geral pode ser uma potencial aliada de artistas de jogos. Isso pode ser alcançado por meio do desenvolvimento de ferramentas baseadas em modelos generativos que auxiliem e facilitem o trabalho artístico sem que haja interferências ou perdas no processo criativo dos artistas envolvidos no trabalho (Liu 2024), como por meio da realização de sugestões, por parte dos modelos, baseadas somente a partir de iniciativas do usuário.

Figura 1: Direções de orientação de personagem em visão *top-down* — frente, esquerda, direita e trás. Fonte: autores.

Assim, Coutinho e Chaimowicz (2024a) propuseram a criação de uma interface de iniciativa mista (i.e. em que ambos usuário humano e agente computacional atuam de forma conjunta) para edição de *sprites* integrada a modelos de redes adversárias generativas (GAN). Em sua pesquisa, os autores se propuseram a abordar a geração de imagens de personagens em poses alvo com base em uma ou mais imagens dos mesmos em poses de origem, por meio de modelos GAN responsáveis por realizar a tradução de imagem para imagem. A Figura 1 mostra as poses contempladas pelos autores, normalmente utilizadas em jogos que permitem a movimentação em quatro direções e que possuem uma visão de cima para baixo (i.e. *top-down*).

Sendo assim, o presente trabalho se propôs a investigar a interface de edição de *Pixel Art* inicialmente proposta por Coutinho e Chaimowicz (2024a), além dos modelos generativos utilizados pelos autores, a fim de se garantir uma experiência enriquecedora para potenciais usuários da ferramenta.

1.1 Objetivos Gerais

Na primeira etapa do nosso trabalho, nos propusemos a investigar a interface *web* para edição de *Pixel Art* integrada a modelos generativos inicialmente proposta por Coutinho e Chaimowicz (2024a). Nossa investigação, realizada mediante pesquisa de usuário, teve a finalidade de validar a hipótese acerca do benefício da utilização complementar de tais modelos por *designers* e artistas de jogos em seus trabalhos. Além disso, buscamos levantar requisitos e implementar ferramentas para a plataforma em questão.

Nossos resultados demonstram um conjunto mínimo de ferramentas consideradas de alta importância para a edição de *Pixel Art* e sugerem que a aplicação de modelos generativos como proposto e para o problema em questão pode auxiliar artistas de jogos em suas tarefas criativas.

Em um segundo momento, os esforços da pesquisa foram direcionados à investigação acerca de possíveis otimizações nas arquiteturas GAN desenvolvidas e treinadas anteriormente por Coutinho, Chaves e Chaimowicz (2025), com a finalidade de se gerar sugestões de imagens com uma menor presença de ruídos de alta frequência, com bordas e formatos mais bem definidos e, de forma geral, mais fidedignas e próximas das imagens alvo. As tentativas de otimização foram implementadas com o objetivo de diminuir a presença de ruídos de alta frequência nas imagens geradas e obter melhores valores para as métricas de avaliação definidas pelos autores originais. Nossa hipótese para esta etapa do trabalho é de que a técnica de quantização diferenciável de paleta (Coutinho, Chaves e Chaimowicz 2025) e o redimensionamento de imagens de entrada por meio de interpolação por vizinho mais próximo, como método de pré-processamento ou técnica de aumento de dados (i.e. *data augmentation*), se configuram como potenciais otimizações aos modelos generativos analisados.

As técnicas de pré-processamento e aumento de dados por redimensionamento de imagens foram investigadas em ambas as arquiteturas integradas ao editor de *Pixel Art* analisado — CollaGAN e StarGAN, enquanto a quantização diferenciável de paleta foi adicionada somente à StarGAN, uma vez que os autores originais já a integraram à arquitetura da CollaGAN em seu trabalho anterior. A motivação de incluir a quantização por paleta é reduzir a presença de pequenas variações de uma mesma cor nas imagens geradas, que ocorre em função do processo de inferência ou geração das imagens em modelos generativos profundos, que produz imagens no espaço contínuo RGBA. O estilo *Pixel Art*, contudo, adota uma paleta

de cores limitadas, devido à utilização de informações discretizadas (i.e. *pixels*) para composição das obras.

1.2 Objetivos Específicos

1. Levantar requisitos para o editor sob a perspectiva do usuário final;
2. Avaliar a interface por meio de pesquisa com potenciais usuários;
3. Investigar formas de apresentação dos modelos generativos ao público-alvo;
4. Validação de hipótese de que a aplicação de modelos generativos no contexto de desenvolvimento de jogos tem a capacidade de apoiar o trabalho criativo de artistas;
5. Validação de hipótese de que a técnica de quantização diferenciável de paleta e o redimensionamento de imagens como pré-processamento ou aumento de dados se configuram como potenciais otimizações aos modelos generativos analisados;
6. Aplicar a técnica de interpolação por vizinho mais próximo nas imagens durante treinamento dos modelos, avaliando sua utilização nas etapas de pré-processamento e de aumento de dados.
7. Aplicar técnicas de quantização de paleta de cores, para garantir que as imagens geradas sigam a paleta de cores das imagens originais;
8. Realizar experimentos para avaliar a utilização das técnicas empregadas nos modelos estudados e analisá-las comparativamente.

2 Referencial Teórico

Neste capítulo, serão descritos os conceitos relacionados ao trabalho, como *Softwares* de desenho, geração de conteúdo procedural, redes adversárias generativas e demais conceitos associados a aprendizado profundo e processamento digital de imagens.

2.1 *Softwares* de Desenho

Coutinho e Chaimowicz (2024a) apresentaram a interface de edição investigada pelo presente trabalho. A interface se trata de uma aplicação *web* voltada para o desenho de *sprites* de personagens em *Pixel Art*, implementada em JavaScript, HTML e CSS. Além disso, a ferramenta é integrada com modelos generativos responsáveis por gerar sugestões de imagens de determinado personagem em outra direção, dada como entrada uma imagem original do mesmo personagem, desenhada pelo usuário. A Figura 2 mostra a interface do editor implementado pelos autores.

Figura 2: Componentes do editor. Fonte: autores.

2.2 Algoritmo de Bresenham para Desenho de Linhas

A API da ferramenta *canvas* do JavaScript utiliza gráficos vetoriais. Porém, no contexto de *Pixel Art*, isto resultaria em desenhos com efeito *anti-aliased*, com variações indesejadas das cores utilizadas. Sendo assim, utilizamos o algoritmo de Bresenham (Bresenham 1965) para desenhos de linhas e de elipses. Abaixo, explicaremos o algoritmo e sua utilização.

Dado um mapa de *pixels* presente em um monitor, os *pixels* se movimentam da esquerda para direita no eixo x e de cima para baixo no eixo y. Neste caso, x será incrementado ao se movimentar da esquerda para a direita e que o y será incrementado ao se movimentar de cima para baixo. O algoritmo em questão possibilita o desenho de um gráfico em *raster* que aproxima a seguinte linha diagonal vetorial r, representada pela Figura 3, tal que $r : y = mx + b$.

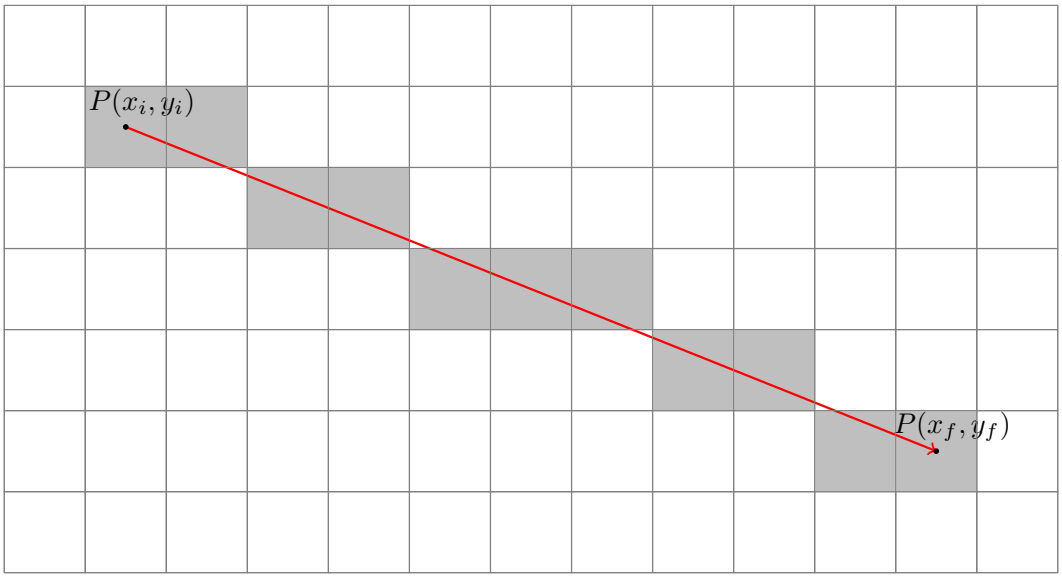


Figura 3: Representação visual do algoritmo de Bresenham. Fonte: autores.

Como o algoritmo trabalha com variações dos movimentos dos *pixels*,a equação da reta pode ser reescrita como $y = \frac{dy}{dx}x + b$, ou em sua forma padronizada em termos de x e y:

$$f(x, y) := (dx)x + (-dx)y + (dx)b = 0 \tag{1}$$

Suponha um ponto $P(x_t, y_t)$ como parte da linha se $f(x_t, y_t) = 0$. Dada a movimentação de um *pixel* como definido anteriormente, ao desenharmos a linha r, o valor da coordenada x será incrementado por um valor à direita. Frente a

esta variação de movimento no eixo x, o algoritmo irá analisar a variação no eixo y para definir se o valor de y frente a esta alteração será incrementado ou se manterá o mesmo.

Para isso, consideramos o ponto médio entre as duas possibilidades de coordenadas do eixo y, (x_t+1, y_t+1) ou (x_t+1, y_t), dado por $f(x_t+1, y_t+1/2)$. A escolha do próximo ponto, entre (x_t+1, y_t+1) e (x_t+1, y_t), será feita de forma que o erro seja minimizado. Se o valor da função para este ponto for zero, o ponto estará na linha com um erro nulo. Se o valor for maior que zero, a linha está abaixo do ponto — neste caso, y será incrementado. Se o valor for menor que zero, a linha está acima do ponto — neste caso, y se mantém.

Porém, o cálculo de um ponto flutuante é custoso. Sendo assim, o algoritmo de Bresenham propõe a utilização exclusiva de números inteiros para realizar esta operação, obtendo a distância D entre o ponto inicial e o ponto final:

$$D = f(x_t + 1, y_t + 1/2) - f(x_i, yi) \tag{2}$$

Convertendo esta expressão para a forma padronizada da equação, encontra-se $D = dy - dx\frac{1}{2}$. Para utilizar cálculos com números inteiros, é possível reescrever a equação acima como:

$$D = 2dy - dx \tag{3}$$

Sendo assim, se $D > 0$, o y será incrementado e o próximo ponto será $(x_t + 1, y_t+1)$ e $\delta D = 2dy-2dx$, enquanto se $D \leq 0$, y se manterá constante, o próximo ponto será $(x_t + 1, y_t)$ e $\delta D = 2dy$.

Para generalizar o algoritmo para aceitar casos de retas com inclinação negativa, com movimentação do eixo x da direita para esquerda e do eixo y de baixo para cima, podemos checar se o ambos x e y serão incrementados ou decrementados a cada iteração do algoritmo.

2.3 Interpolação de Imagens

Interpolação é uma técnica de Processamento Digital de Imagens utilizada para tarefas como ampliação (*zooming*), encolhimento (*shrinking*), rotação e correção digital, na qual se utiliza dados conhecidos para estimar valores em pontos desconhecidos. (Gonzalez e Woods 2017). Isto pode ser realizado criando-se uma malha de dimensões maiores em relação à imagem original e que contenha o mesmo espaçamento de *pixels* desta, seguida do encolhimento da malha para as dimensões originais da imagem, de forma que a malha encolhida tenha um espaçamento menor que a imagem original.

A técnica de interpolação por vizinho mais próximo em específico se baseia em estimar um valor de intensidade para cada ponto desconhecido na malha através

da obtenção do *pixel* presente na imagem original que seja conhecido e que esteja mais próximo (vizinho) ao *pixel* cujo valor de intensidade seja desconhecido. Esta técnica pode produzir artefatos indesejados, como a distorção de bordas. Devido a isso, utiliza-se outros métodos de interpolação, como os métodos bilinear e bicúbico. Contudo, tais métodos utilizam médias de valores de intensidades dos pontos vizinhos, que, por sua vez, impede a utilização de tais métodos no presente trabalho, visto que buscamos a ampliação de imagens em *Pixel Art* sem que haja alteração em sua paleta de cores. A Figura 4 exemplifica o funcionamento desta técnica.

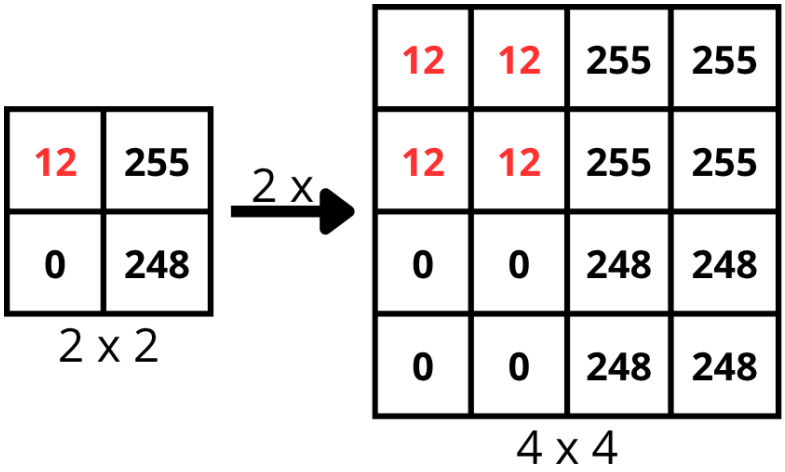


Figura 4: Exemplo de interpolação por vizinho mais próximo, com redimensionamento de uma matriz 2×2 para uma matriz 4×4. Fonte: autores.

2.4 Geração de Conteúdo Procedural

O conceito de geração de conteúdo procedural (PCG) pode ser entendido como a criação algorítmica de conteúdo de jogos com entrada limitada ou indireta do usuário. Quando utilizada em ferramentas de *design* inteligentes, a PCG pode permitir que *designers* e artistas de jogos possam focar em tarefas criativas sem se preocuparem com detalhes e trabalhos manuais repetitivos (Togelius, Shaker e Nelson 2016).

A PCG utilizada em conjunto com a criatividade humana é chamada de PCG de iniciativa mista, na qual ambos agentes (humano e computacional) participam da criação de conteúdo. Esses sistemas necessitam que haja entradas a partir do agente humano para que o agente computacional tome ações, de forma que apenas parte do processo de criação seja automatizado. À título de exemplo, Baldwin et al. (2017) criou um sistema de PCG de iniciativa mista para assistir *designers*

na tarefa de criação de mapas, por meio da utilização de algoritmos genéticos, para possibilitar a criação de heurísticas para padrões de *game design* que permitem a geração de níveis complexos baseados no tipo de mecânica de jogo utilizada.

Além disso, a utilização de aprendizado profundo — através de PCGML (Summerville et al. 2018) e de aprendizado por reforço — através de PCGRL (Khalifa et al. 2020) tem possibilitado avanços na geração de conteúdo procedural. Em especial, Coutinho e Chaimowicz (2024a) investigaram a utilização de PCGML de iniciativa-mista em relação à criação de personagens em *Pixel Art* por meio do desenvolvimento de um editor de *sprites* baseado em agentes computacionais que auxiliem e facilitem o trabalho de artistas de jogos sem que haja interferências ou perdas no processo criativo (Liu 2024).

2.5 Redes Adversárias Generativas

Redes adversárias generativas, ou GANs, podem ser compreendidas como um *framework* para geração de conteúdo, sobretudo visual, através do processo de treinamento adversário. Nesta abordagem, treina-se duas redes neurais convolucionais que competem entre si em um jogo *minmax*. O gerador, G, evolui para ser capaz de gerar novo conteúdo similar aos utilizados durante seu treinamento, enquanto o discriminador, D, aprende a discriminar dados reais de dados gerados por G. Ambas as redes, no artigo original, são *perceptrons* multicamadas (Goodfellow et al. 2014).

Nos últimos anos, diversas arquiteturas de GANs foram alvo de pesquisa para os mais diversos problemas relacionados à geração de imagens. Isola et al. (2017) pesquisaram a utilização de GANs condicionais em problemas de tradução de imagem-para-imagem, através da arquitetura Pix2Pix, na qual utiliza-se uma imagem em um determinado domínio para gerar uma outra imagem em um domínio alvo (Pang et al. 2021). Ainda no contexto de GANs condicionais, Choi et al. (2018) desenvolveram a StarGAN, proposta para realizar a tradução de imagem para imagem em múltiplos domínios, a partir de um único domínio de origem. Por sua vez, Lee et al. (2019) propuseram a arquitetura *Collaborative* GAN ou CollaGAN, aplicada ao problema de imputação de dados ausentes.

Nessa linha de pesquisa, trabalhos recentes buscaram aplicar arquiteturas de GANs condicionais, como Pix2Pix, CollaGAN e StarGAN ao problema de geração de *sprites* de personagens em *Pixel Art* em uma determinada posição, dada como entrada imagens destes em posições originais (Coutinho e Chaimowicz 2022a; Coutinho e Chaimowicz 2022b; Coutinho e Chaimowicz 2024a; Coutinho e Chaimowicz 2024b; Coutinho, Chaves e Chaimowicz 2025). Neste trabalho, estamos interessados, sobretudo, em dar continuidade e expandir as pesquisas iniciadas por tais trabalhos.

Inicialmente, Coutinho e Chaimowicz (2022a) investigaram o uso de *GANs*

na geração de *sprites* de personagens de *Pixel Art* em uma direção alvo (e.g. olhando para a direita) a partir de uma imagem do personagem em outra direção (e.g. olhando para frente), em um problema de tradução de imagem-para-imagem. Para isso, utilizaram a arquitetura Pix2Pix. Em um segundo momento, os autores continuaram a investigação acerca do uso dessa mesma arquitetura para o problema proposto, explorando o uso de diferentes conjuntos de dados (Coutinho e Chaimowicz 2022b).

Posteriormente, os autores investigaram também as arquiteturas CollaGAN e StarGAN, comparando-as com a Pix2Pix (Coutinho e Chaimowicz 2024a). Além disso, realizaram-se modificações na CollaGAN, como o aumento de capacidade do modelo por meio do aumento do número de canais em cada camada da arquitetura e a aplicação de uma nova política de *dropout* de dados de entrada. As arquiteturas foram integradas à interface *web* de interação mista (i.e. em que ambos usuário humano e agente computacional atuem de forma conjunta) para edição de *sprites* de personagens em *Pixel Art* supracitada, na qual os modelos generativos são aplicados ao problema de tradução de imagem para imagem (no caso da StarGAN) e de imputação de dados ausentes (no caso da CollaGAN), por meio da geração de imagens de determinado personagem em diferentes poses.

Originalmente, Lee et al. (2019) observaram que imagens produzidas pela CollaGAN perdiam a qualidade ao se diminuir a quantidade de imagens de diferentes domínios utilizadas como entrada. Porém, como é comum a ausência de mais de um domínio em situações reais, os autores investigaram o *dropout* de entradas como estratégia de seleção de lotes. Por exemplo: para o problema proposto de geração de *sprites*, no qual temos quatro domínios, ao se selecionar um lote, pode-se utilizar três, duas ou apenas uma imagem dos domínios de origem. Na arquitetura original, o número de imagens a serem excluídas do lote é escolhido uniformemente, com 33% de chance de se utilizar todas as imagens dos domínios de origem.

Contudo, Coutinho e Chaimowicz (2024b) posteriormente observaram que uma abordagem mais conservadora de *dropout* favorece a obtenção de melhores resultados. A abordagem conservadora sugerida pelos autores consiste em treinar o modelo com maior probabilidade de não haver exclusão de imagens dos lotes, e com uma diminuição gradativa na probabilidade de se ocorrer o *dropout* conforme se aumenta a quantidade de imagens excluídas do lote. Mais especificamente, há 60% de probabilidade de não haver exclusão de dados, 30% de chance de se excluir apenas uma imagem do lote e 10% de se excluir duas imagens do lote — situação na qual a entrada será somente uma única imagem de um dos três domínios de origem. Além disso, também foram realizadas alterações relacionadas ao processo de substituição de imagens alvo originais pelas imagens cíclicas geradas durante a etapa reversa de treinamento do modelo.

Em seu trabalho mais recente, Coutinho, Chaves e Chaimowicz (2025) propuseram novas modificações à CollaGAN para garantir que todas as imagens geradas sigam estritamente a paleta de cor das imagens de origem, por meio de um processo de Quantização Diferenciável de valores de *pixels* — obtendo, assim, resultados que se igualam ou superam o estado da arte. A busca por métodos diferenciáveis de quantização da paleta de cores das imagens se deu devido ao fato de que a aproximação estatística produzida pelo gerador gera muitas variações de uma mesma cor — o que pode resultar em retrabalho para artistas de *Pixel Art*. Sendo assim, o espaço latente contínuo das redes não garante que as imagens geradas sigam, a priori, uma paleta de cores. Em nosso trabalho, nós buscamos reproduzir e adaptar o método de quantização desenvolvido pelos autores, agora para o modelo baseado na arquitetura StarGAN. Este estudo é detalhado no Capítulo 3 (Metodologia).

As subseções seguintes irão descrever brevemente as arquiteturas CollaGAN e StarGAN utilizadas por Coutinho e Chaimowicz (2024b) e estudadas também no presente trabalho.

2.5.1 CollaGAN

Modelo de Base (Lee et al. 2019). O problema investigado pela arquitetura CollaGAN é o da imputação de dados ausentes, situação na qual o modelo espera receber múltiplas entradas de múltiplos domínios para gerar o dado ausente de um domínio específico. A sua arquitetura original possui um único par de gerador e discriminador. O gerador segue o padrão *U-net*, consistindo de codificadores e decodificadores conectados entre si. Os autores utilizaram normalização de instância (Ulyanov, Vedaldi e Lempitsky 2017) e Leaky ReLU (K. He et al. 2015), em detrimento de normalização por lote e ReLU, respectivamente. O discriminador é composto por uma série de camadas de convolução e Leaky ReLU. Para discriminar as imagens locais, PatchGAN (Isola et al. 2017; Zhu et al. 2017) foi utilizada. O modelo treina para otimizar uma função objetivo com os seguintes termos: perda adversarial, perda de consistência cíclica e índice de similaridade estrutural (SSIM). O treinamento é supervisionado e pareado (Lee et al. 2019).

Coutinho, Chaves e Chaimowicz (2025). A arquitetura da CollaGAN apresentada pelos autores conta com modificações na topologia do gerador, para se adequar ao problema proposto de geração de *sprites*. O gerador possui quatro ramificações de codificador, cada uma responsável por processar a imagem de origem de cada um dos quatro domínios possíveis, além do rótulo *one-hot encoded* do domínio alvo e um *v*etor que representa a paleta requisitada para a imagem do domínio alvo. Cada ramificação possui quatro blocos de *downsampling*, seguidos por uma camada *bottleneck* para concatenar os mapas de ativação das ramificações. Em seguida, as informações são passadas para um decodificador compartilhado

com quatro blocos de *upsampling*. Os detalhes espaciais são preservados através de conexões com salto (i.e. *skip connections*) entre as camadas de ativação dos codificadores e suas respectivas camadas de *upsampling* no decodificador. Por conseguinte, os autores aumentaram o número de canais para cada uma das camadas a fim de prover maior capacidade para a rede.

Por sua vez, o discriminador recebe um lote de imagens e retorna valores binários que determinam, a partir do ponto de seu ponto de vista, a procedência de cada uma das imagens vistas — isto é, se são imagens reais ou criadas pelo gerador. Ademais, o discriminador classifica o domínio de cada imagem, através da probabilidade D_{dmn} desta ter cada um dos domínios possíveis. A topologia do discriminador não foi alterada, com este possuindo seis blocos de *downsampling*. Cada um dos blocos consiste de uma convolução que reduz a resolução pela metade e incrementa o número de canais com ativação Leaky ReLU. O último bloco do discriminador contém uma camada de *dropout* e duas convoluções paralelas que representam os valores D_{adv} e D_{dmn} gerados através de ativação pelas funções `linear` e `softmax`, respectivamente.

A função objetivo do gerador utiliza as seguintes perdas: reconstrução, adversarial, classificação de domínio, consistência cíclica múltipla e índice de similaridade estrutural (SSIM), presentes também na função de custo do modelo de base — além dae paleta de cores como novo componente. Em contrapartida, a função objetivo do discriminador utiliza apenas a perda adversarial e a perda de classificação de domínio, também presentes no modelo de base. A seguir, descreveremos cada uma das funções de perda citadas.

Cada passo de treinamento consiste em uma etapa direta e outra reversa. Durante a direta, o gerador G recebe um minilote de imagens pareadas com um domínio ausente aleatório t, para em seguida sintetizar uma imagem correspondente a este domínio.

Supondo um problema no qual haja os seguintes domínios $N = \{a, b, c, d\}$. Caso $t = d$, os domínios de origem serão $S = \{a, b, c\}$, com suas respectivas imagens de entrada x_a, x_b e x_c. Seja x_{zero} o tensor preenchido com zeros e com dimensão equivalente aos outros tensores. Nessa situação, o modelo gerará a imagem $\hat{x}_d$ que se aproxima da imagem x_d real tal que:

$$\hat{x}_d = G(\langle x_a, x_b, x_c, x_{zero} \rangle, d) \tag{4}$$

Para habilitar a consistência cíclica no gerador, isso é, a reconstrução das imagens de origem, a etapa reversa do passo de treinamento irá sintetizar as imagens $\tilde{x}_{a|d}$, $\tilde{x}_{b|d}$ e $\tilde{x}_{c|d}$ para cada um dos domínios em S, utilizando $\hat{x}_d$ para reconstruir as imagens originais:

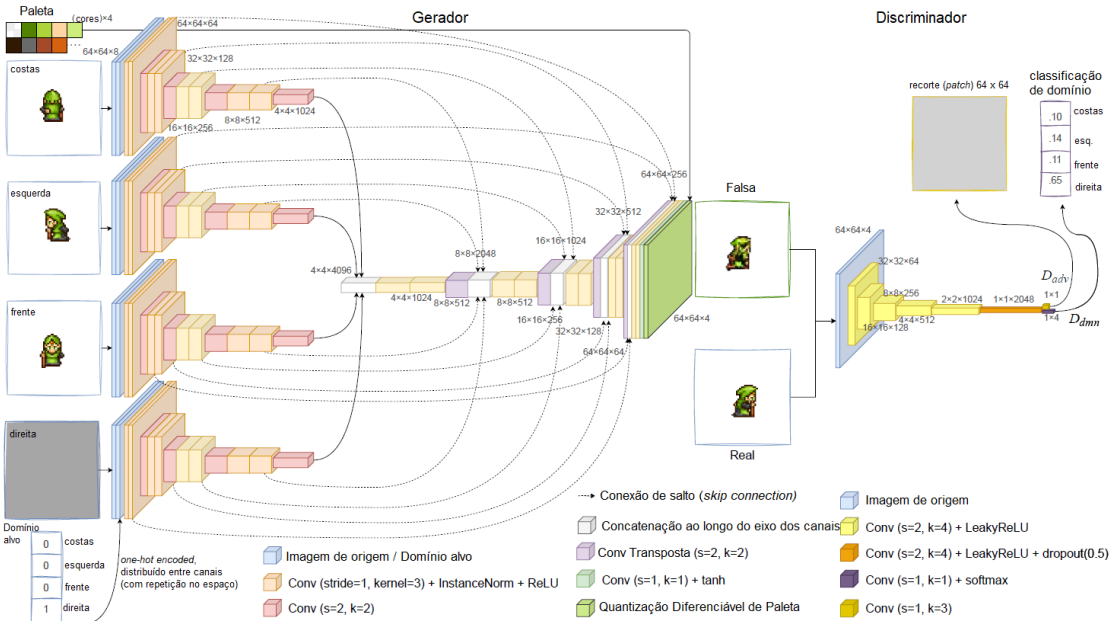


Figura 5: Arquitetura CollaGAN proposta por Coutinho, Chaves e Chaimowicz (2025). Esquerda: gerador recebe um personagem nos domínios de origem, a paleta alvo e um rótulo *one-hot encoded* replicado espacialmente indicando o domínio alvo. Ou seja, cada uma das posições $H \times W$ da imagem recebe os mesmos valores, para então serem concatenados a cada um dos canais. Os dados de entrada seguem as ramificações de codificadores e são concatenados na camada de *bottleneck* para, em seguida, serem propagados para o decodificador. Direita: o discriminador recebe a imagem e discrimina se esta é real ou falsa, gerando os valores D_{adv} com *logit* real/falsa e D_{dmn} com probabilidades da imagem possuir cada um dos quatro domínios possíveis. Fonte: Coutinho, Chaves e Chaimowicz (2025) (traduzido).

$$\begin{aligned} \tilde{x}_{a|d} &= G(\langle x_{zero}, x_b, x_c, \hat{x}_d \rangle, a) \\ \tilde{x}_{b|d} &= G(\langle x_a, x_{zero}, x_c, \hat{x}_d \rangle, b) \\ \tilde{x}_{b|c} &= G(\langle x_a, x_b, x_{zero}, \hat{x}_d \rangle, c) \end{aligned} \tag{5}$$

A perda regressiva irá orientar o gerador a utilizar informações dos domínios de origem para traduzir a imagem gerada corretamente:

$$\mathcal{L}_{rec} = \mathbb{E}_{x_t, \hat{x}_t}[\|x_t - \hat{x}_t\|_{L_1}] \tag{6}$$

Enquanto isso, a perda de consistência cíclica múltipla o conduz a codificar informações de $\hat{x}_t$ para permitir reconstrução cíclica das entradas:

$$\mathcal{L}_{mcyc} = \mathbb{E}_{x_s,\tilde{x}_{s|t}}\left[\sum_{s\in S} \|x_s - \tilde{x}_{s|t}\|_{L_1}\right] \tag{7}$$

A perda SSIM é utilizada para otimizar a qualidade das imagens geradas durante a etapa reversa, buscando aproximar as imagens $\tilde{x}_s$ das imagens reais x_s pertencentes a S:

$$\mathcal{L}_{ssim} = \mathbb{E}_{x_s,\tilde{x}_{s|t}}\left[-\log\left(\frac{1}{2|P|}\sum_{p\in P(x_s,\tilde{x}_{s|t}),s\in S}(1+SSIM(p))\right)\right] \tag{8}$$

A perda adversarial, empregada em ambos gerador e discriminador, otimiza o Quadrado dos Erros das classificações feitas pelo discriminador. As perdas adversariais do gerador, $\mathcal{L}^G_{adv}$, e do discriminador, $\mathcal{L}^D_{adv}$, são, respectivamente:

$$\mathcal{L}^G_{adv} = \mathbb{E}_{\hat{x}_t}[(D_{adv}(\hat{x}_t)-1)^2] + \mathbb{E}_{\tilde{x}_{s|t}}\left[\sum_{s\in S}(D_{adv}(\tilde{x}_{s|t})-1)^2\right] \tag{9}$$

$$\mathcal{L}^D_{adv} = \mathbb{E}_{x_t}[(D_{adv}(x_t)-1)^2] + \mathbb{E}_{\hat{x}_t}[(D_{adv}(\hat{x}_t))^2] + \mathbb{E}_{\tilde{x}_{s|t}}\left[\sum_{s\in S}(D_{adv}(\tilde{x}_{s|t})^2)\right] \tag{10}$$

A perda de classificação de domínio, calculada por meio de entropia cruzada, orienta o gerador a sintetizar imagens que são classificadas com o domínio correto. A perda $\mathcal{L}^{fake}_{dmn}$, do gerador, utiliza apenas imagens geradas, enquanto a perda $\mathcal{L}^{real}_{dmn}$, do discriminador, considera apenas imagens reais:

$$\mathcal{L}^{fake}_{dmn} = \mathbb{E}_{\hat{x}_t}[-\log(D_{dmn}(\hat{x}_t))] + \mathbb{E}_{\tilde{x}_{s|t}}\left[\sum_{s\in S} -\log(D_{dmn}(\tilde{x}_{s|t}))\right] \tag{11}$$

$$\mathcal{L}^{real}_{dmn} = \mathbb{E}_{x_t}[-\log(D_{dmn}(x_t))] \tag{12}$$

Por fim, os autores incluíram também a perda de cobertura de paleta para o gerador, $\mathcal{L}_{pcov}$, para garantir que as imagens geradas contenham a paleta esperada em sua integridade. Esta perda será abordada com mais detalhes no Capítulo 3(Metodologia). Sendo assim, as funções objetivo do gerador, $\mathcal{L}_G$, e do discriminador, $\mathcal{L}_D$, são as seguintes:

$$\mathcal{L}_G = \mathcal{L}^G_{adv} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{mcyc}\mathcal{L}_{mcyc} + \lambda_{ssim}\mathcal{L}_{ssim} + \lambda_{dmn}\mathcal{L}^{fake}_{dmn} \tag{13}$$

$$\mathcal{L}_D = \mathcal{L}^D_{adv} + \lambda_{dmn}\mathcal{L}^{real}_{dmn} \tag{14}$$

2.5.2 StarGAN

Modelo de Base (Choi et al. 2018). Enquanto a CollaGAN investiga o problema da imputação de dados ausentes, o problema investigado pela arquitetura original da StarGAN é a tradução de imagem para imagem. A sua arquitetura também possui um único par de gerador e discriminador. O modelo espera receber uma imagem em um domínio de origem em conjunto com um rótulo do domínio alvo, para então gerar uma imagem no domínio desejado. O treinamento é não-supervisionado e, por padrão, não-pareado.
Nesse contexto, o gerador G é treinado para traduzir uma imagem de entrada x_s, com um domínio s, em uma imagem de saída $\hat{x}_t$ condicionada somente ao rótulo do domínio alvo t, gerado aleatoriamente para permitir flexibilidade durante a tradução, tal que $G(x_s, t) \rightarrow \hat{x}_t$. O discriminador D utiliza um classificador auxiliar para controlar múltiplos domínios por meio de distribuição de probabilidades sobre a fonte da imagem (i.e. real ou falso) e sobre o seu rótulo de domínio, tal que $D : x \rightarrow D_{src}(x), D_{dmn}(x)$, onde x pode ser uma imagem real ou uma imagem falsa, sintetizada pelo gerador.
A arquitetura do modelo de base é adaptada a partir da CycleGAN (Zhu et al. 2017). O gerador é composto por duas camadas convolucionais com um $stride = 2$ para *downsampling*, seis blocos residuais e duas camadas de convoluções transpostas com $stride = 2$ para *upsampling*. Além disso, utiliza-se também Normalização de Instância. Enquanto isso, o discriminador utiliza PatchGAN (Isola et al. 2017; Zhu et al. 2017) para classificar se as imagens locais são reais ou falsas (i.e. produzidas pelo gerador).

A função objetivo do modelo é constituída pelos seguintes componentes:

Perda adversarial. Seja G o gerador, D o discriminador, x_s a imagem de entrada, t o rótulo do domínio alvo e $D_{src}(x_s)$ a distribuição de probabilidade sobre os domínios de origem. Sabendo-se que $G(x_s, t) = \hat{x}_t$, a perda adversarial pode ser definida da seguinte forma:

$$\mathcal{L}_{adv} = \mathbb{E}_{x_s}[\log D_{src}(x_s)] + \mathbb{E}_{\hat{x}_t}[\log(1 - D_{src}(\hat{x}_t))] \tag{15}$$

a fim de conduzir o gerador a produzir imagens próximas das reais. Esta função é utilizada pelo gerador e também pelo discriminador.

Perda de classificação de domínio. Utilizada para se garantir que, a partir de x_s e t, gera-se uma imagem $\hat{x}_t$ que seja corretamente classificada como pertencente ao domínio t. Para isso, é adicionado um classificador ao discriminador D, responsável por realizar a classificação de domínio durante a otimização de ambos

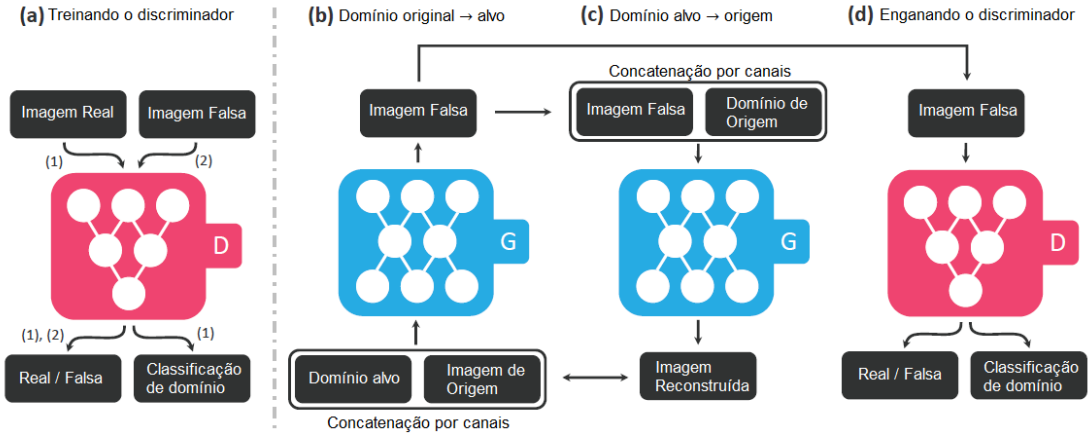


Figura 6: Visão geral da arquitetura original da StarGAN, com discriminador D e gerador G. (a) D aprende a distinguir entre imagens reais e falsas e classificar imagens reais ao seu domínio correspondente. (b) G recebe como entrada a imagem de origem, o rótulo contendo o domínio alvo e gera a imagem falsa. O rótulo do domínio alvo é espacialmente replicado e concatenado à imagem de entrada. (c) G tenta reconstruir a imagem original de entrada a partir da imagem falsa gerada, dada o rótulo de domínio de origem. (d) G tenta gerar imagens indistinguíveis das reais, para serem classificadas como o domínio alvo por D. Fonte: Choi et al. (2018) (traduzido).

gerador e discriminador. A função é, então, dividida entre dois termos: a classificação de domínio de imagens reais utilizada para otimizar D; e a classificação de domínio de imagens falsas utilizadas para otimizar G, respectivamente:

$$\mathcal{L}^{r}_{dmn} = \mathbb{E}_{x_s,s}[-\log D_{dmn}(s|x_s)] \tag{16}$$

$$\mathcal{L}^{f}_{dmn} = \mathbb{E}_{\hat{x}_t,t}[-\log D_{dmn}(t|\hat{x}_t)] \tag{17}$$

Em (16), a distribuição de rótulos dos domínios computados por D é representada por $D_{dmn}(s|x_s)$. O objetivo do discriminador é minimizar esta função, para então classificar x_s, a imagem real de entrada, a seu domínio correspondente s. Enquanto isso, (17) deve ser minimizada por G, para que este possa gerar imagens cujo domínio possa ser classificado como t.

Perda de consistência cíclica. Segundo os autores, a minimização da perda adversarial e de classificação de domínio não são suficientes para garantir que as imagens traduzidas preservem o conteúdo das imagens de entrada. Sendo assim,

os autores aplicam a perda de consistência cíclica ao gerador:

$$\mathcal{L}_{cyc} = \mathbb{E}_{x_s,\tilde{x}_{s|t}}[\|x_s - \tilde{x}_{s|t}\|_{L_1}] \tag{18}$$

onde $\tilde{x}_{s|t} = G(G(x_s, t), s)$ representa a imagem de origem x_s gerada a partir de t. Portanto, o gerador é utilizado não apenas para gerar a imagem traduzida, mas também para reconstruir a imagem original a partir da imagem traduzida.
Sendo assim, a função objetivo completa do gerador e do discriminador podem, respectivamente, ser definidas da seguinte forma:

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{dmn}\mathcal{L}^f_{dmn} + \lambda_{cyc}\mathcal{L}_{cyc} \tag{19}$$

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{dmn}\mathcal{L}^r_{dmn} \tag{20}$$

Coutinho, Chaves e Chaimowicz (2025). Os autores utilizaram a arquitetura do modelo de base para o problema de tradução multi-domínio. Porém, enquanto o modelo de base recebe como entrada apenas a imagem no domínio de origem e o rótulo do domínio alvo, esta nova versão do modelo recebe como entrada também o rótulo do domínio de origem. Abaixo definimos a nova perda introduzida pelos autores.

Perda de reconstrução. Para tornar o treinamento dos modelos StarGAN supervisionado e pareado, os autores adicionaram a perda de reconstrução à função objetivo do gerador, descrita em 21, por meio do cálculo da norma L1 entre a imagem alvo real, x_t, e a imagem alvo gerada, $\hat{x}_t = G(x_s, t)$, considerando o domínio alvo t.

$$\mathcal{L}_{rec} = \mathbb{E}_{x_t,\hat{x}_t}[\|x_t - \hat{x}_t\|_{L_1}] \tag{21}$$

Além disso, o discriminador recebe não apenas a imagem a ser avaliada, mas também a imagem de origem — realizando, assim, uma discriminação condicional. Com estas alterações, os autores encontraram melhores resultados para o problema proposto.

3 Metodologia

Este capítulo descreve os métodos utilizados na investigação acerca do editor de *Pixel Art* referenciado, e também na pesquisa por possíveis otimizações dos modelos generativos integrados à plataforma. Inicialmente, será discutida a pesquisa de usuário conduzida. Depois, o foco será dado aos métodos aplicados durante a implementação de novas funcionalidades básicas de edição.

Em seguida, serão definidos o problema da geração de *sprites* aplicado ao nosso contexto e a otimização em arquiteturas de redes adversárias generativas. Por fim, também serão discutidas as estratégias de otimização adotadas com o objetivo de se obter imagens mais limpas e com uma menor presença de ruídos de alta frequência.

3.1 Repositórios

Os repositórios da interface de edição de *Pixel Art*[1] proposta por Coutinho, Chaves e Chaimowicz (2025) e dos modelos generativos utilizados[2] com as modificações propostas pelo presente trabalho encontram-se disponíveis no GitHub. Além disso, os autores também disponibilizaram um protótipo *web* do editor que foi utilizado durante a pesquisa de usuário[3].

3.2 Pesquisa de Usuário

Nesta seção, será definida a hipótese que desejou-se validar com a pesquisa conduzida, além de sua estrutura e dos métodos relacionados à sua aplicação. Adiante, serão abordados os métodos relacionados à análise quantitativa e qualitativa dos nossos resultados.

3.2.1 Hipótese

A pesquisa busca responder a seguinte pergunta: "A utilização de modelos generativos a uma interface de interação de iniciativa mista para gerar sugestões de imagens pode apoiar o processo de criação e modificação de personagens *Pixel Art* realizado por artistas de jogos?"

3.2.2 Métodos de Aplicação de Pesquisa de Usuário

A pesquisa teve caráter qualitativo e foi realizada de forma assíncrona e remota, através de um questionário via *Google Forms*. Segundo Labes (1998), um ques-

[1] *Sprite Editor*: `https://github.com/fegemo/sprite-editor/`
[2] *Multi-domain*: `https://github.com/fegemo/multi-domain`
[3] Protótipo: `https://fegemo.github.io/sprite-editor/`

tionário pode ser utilizado frente à necessidade do registro de informações (e.g. casos de validação de hipóteses), sendo estruturado pelas seguintes etapas, que serão exploradas adiante:

I. Pesquisa, análise dos objetivos e delimitação do problema;

II. Elaboração do questionário;

III. Pré-teste;

IV. Aplicação;

V. Tabulação dos dados;

VI. Análise e interpretação.

Pesquisa, análise dos objetivos e delimitação do problema. A delimitação do problema se baseou na necessidade de se validar a relevância, para artistas de jogos, de uma interface de edição como a proposta, integrada a modelos generativos de aprendizado profundo, aplicada ao processo de criação de *Pixel Art*, a fim de evitar tarefas repetitivas e mecânicas.
Além disso, compreender qual seria um conjunto mínimo de ferramentas consideradas importantes ou essenciais para o editor é de importância para que este seja útil para o público-alvo. Portanto, foi necessário conhecer potenciais usuários finais, suas dores e suas perspectivas sobre a interface.

Elaboração do questionário e pré-teste. A elaboração do questionário foi realizada tendo-se em vista os objetivos e a delimitação supracitada. O pré-teste foi realizado pelos próprios autores, sobretudo para estipular o tempo de duração da participação na pesquisa.

Estrutura do questionário. A estrutura do questionário pode ser dividida em:

I. Texto introdutório, que busca apresentar o trabalho e informar os participantes sobre a pesquisa, sua participação e nossos interesses;

II. Termo de Consentimento Livre e Esclarecido;

III. Interação com editor simplificado e uma pergunta aberta sobre a percepção de uso do participante sobre o editor;

IV. Uma pergunta fechada sobre sua interação anterior com o editor. Uma pergunta fechada e duas abertas sobre a experiência pessoal do participante em relação à criação de *Pixel Art* e a tarefas repetitivas ou mecânicas encontradas durante este processo artístico.

V. Novo texto sobre a versão completa do editor, integrado com os modelos GAN e sobre o uso ético de inteligência artificial (IA) para auxiliar o trabalho artístico sem que haja perda de criatividade ou de protagonismo do artista. Nesta etapa, mostramos também um exemplo de uso dos modelos de IA no protótipo, e solicitamos que o participante realize uma nova interação, agora com a versão completa do editor.

VI. Uma pergunta aberta sobre a percepção do participante sobre a utilização de modelos de IA como os propostos no contexto analisado. Dois espaços opcionais para o participante realizar comentários adicionais ou reportar erros durante suas interações com o editor e, por fim, uma pergunta fechada acerca do desejo do participante de receber atualizações sobre o trabalho.

VII. Caso o participante deseje receber atualizações, uma última parte do formulário é apresentada, para que o email do participante seja inserido.

Sendo assim, o questionário possui um total de oito perguntas, sendo duas fechadas e seis abertas. Dentre estas, seis são optativas — sendo obrigatórias somente as relacionadas a comentários adicionais e relatos de erros.

Aplicação do questionário. Para a aplicação do questionário, foram utilizadas as seguintes técnicas:

I. Pesquisa de percepção, visto que buscou-se analisar a percepção subjetiva de potenciais usuários finais do editor, como artistas, acerca de sua experiência individual com criação de *Pixel Art* e da interface, com base em interações com esta;

II. Pesquisa exploratória, visto que se trata de uma pesquisa qualitativa com perguntas abertas e fechadas, buscando explorar ideias, percepções e opiniões acerca do contexto abordado, a fim de validar hipóteses;

III. Teste de usabilidade informal, visto que as interações propostas com o editor visam analisar como o usuário percebe a interface e seus componentes visuais e funcionais, em um ambiente não controlado e sem técnicas formais de testes de usabilidade.

A amostragem da pesquisa teve as seguintes características:

I. Amostragem por conveniência, pois o questionário foi apresentado a pessoas conhecidas que fazem parte do público-alvo constituído por potenciais usuários finais da interface de edição;

II. Amostragem em bola de neve, pois o questionário foi enviado para pessoas próximas que trabalham como artistas de jogos — que, por sua vez, poderiam (ou não) indicar o questionário para outros artistas conhecidos;

III. Amostragem não-probabilística aleatória, pois a pesquisa foi compartilhada em grupos virtuais de artistas e desenvolvedores de jogos, em plataformas como *Reddit*, *Facebook*, *Discord* e *WhatsApp*, com artistas aleatórios.

A pesquisa foi parcialmente anônima, pois permitiu, de forma opcional, que o participante informasse seu email caso quisesse receber atualizações do trabalho. Contudo, cabe salientar que, o preenchimento do questionário via *Google Forms*, assim como a natureza de algumas perguntas presentes, relacionadas à percepção de uso do protótipo pelo participante, o individualizam.
Por se tratar de um público-alvo nichado (artistas de jogos especializados em *Pixel Art*), optou-se por aplicar o questionário em língua portuguesa e inglesa, como tentativa de se aumentar o alcance da pesquisa e o número de participantes.
Para as interações com o editor, foi aplicado o conceito de cenário de uso, amplamente utilizado no campo da Interação Humano-Computador para pesquisas relacionadas à usabilidade e à experiência de usuário. Segundo Preece, Sharp e Rogers (2002), um cenário pode ser compreendido como uma narrativa informal, que descreve uma tarefa em uma história que permite a exploração e a discussão de determinados contextos, necessidades e requisitos.

3.2.3 Métodos de Avaliação de Pesquisa de Usuário

A análise dos resultados da pesquisa foi feita utilizando uma abordagem qualitativa para as perguntas abertas e quantitativa para as duas perguntas fechadas presentes no questionário.
Segundo Labes (1998), a estatística na análise e na interpretação de dados pode ser dividida entre estatística e indutiva — esta última apoiada em inferências e conclusões interpretativas. Sendo assim, inicialmente será discutida a tabulação dos dados e, em seguida, serão abordados os métodos de análise estatística e indutiva.

Tabulação dos dados. A tabulação dos dados foi realizada através da exportação do questionário para a ferramenta *Google Sheets* — funcionalidade nativa

do *Google Forms*. Como o questionário foi aplicado em língua portuguesa e inglesa, os resultados foram mesclados em apenas uma tabela, na qual o único resultado do questionário em inglês foi traduzido para português, com devido cuidado para evitar possíveis enviesamentos na tradução.
Em seguida, foi criada uma tabela para avaliar as respostas da segunda pergunta, referente à importância de cada uma das funcionalidades de edição citadas, para possibilitar a visualização da frequência de cada uma das respostas para cada um dos itens.
Por conseguinte, foi gerada outra tabela para armazenar as estatísticas descritivas, como média, mediana, moda e desvio-padrão, acerca da terceira pergunta, sobre a frequência de tarefas repetitivas e mecânicas, além de uma tabela similar à primeira, relativa à frequência de cada uma das respostas.

Análise descritiva e indutiva de dados quantitativos. Na análise de dados quantitativos das perguntas fechadas, cujas alternativas são ordinais, foi calculada a frequência absoluta para cada opção de resposta. A visualização dos dados foi realizada através de gráficos de barra.
Além disso, foram calculadas as estatísticas descritivas das alternativas, utilizando-se mediana, moda, média e desvio-padrão. Com base nesta abordagem, foi possível, na pergunta relacionada à importância de cada ferramenta de edição proposta, por exemplo, classificar quais são aquelas que são mais ou menos essenciais para uma interface de edição como a proposta.
Por sua vez, a pergunta sobre a frequência de tarefas repetitivas, possibilita a visualização com o contexto de criação de *Pixel Art* e oferece um dimensionamento de possíveis dores de artistas de jogos — que pode potencialmente ser sanada através de automatizações parciais no processo em questão.

Análise descritiva e indutiva de dados qualitativos. Para análise descritiva de dados qualitativos, foi gerada uma nuvem de palavras, a fim de se obter uma sumarização e descrição visual dos padrões de resposta dos participantes. Na análise inferencial, inicialmente foi realizada uma familiarização com os dados, por meio da leitura íntegra das respostas.
Em seguida, foram identificados padrões, temas e categorias nas respostas dos participantes, buscando identificar trechos de respostas que representam ideias e conceitos relevantes, frequentes ou semelhantes entre si. Posteriormente, os padrões encontrados foram interpretados e relacionados aos objetivos da pesquisa.
Por fim, foi realizada uma avaliação mista dos dados qualitativos e quantitativos, na qual buscou-se correlacionar ambos para responder perguntas como *"participantes que se deparam com muitas tarefas repetitivas tendem a achar que a utilização proposta de modelos generativos pode auxiliar o processo de criação de Pixel Art?"*.

3.3 Métodos de Desenvolvimento de Funcionalidades Básicas de Edição

As funcionalidades desenvolvidas durante o trabalho foram escolhidas levando-se em consideração a pesquisa de usuário realizada para levantamento de requisitos. A implementação foi realizada inteiramente com HTML, CSS e JavaScript. Devido a este fato, foi utilizado o Modelo de Objeto de Documentos — DOM para a manipulação e modificação de elementos do documento *web* — isto é, da interface.

Para a construção de uma tela de desenho dinâmica e interativa, foi utilizada a `canvas` de HTML, baseada em gráficos vetoriais, tendo sido necessário, assim, a criação de técnicas de rasterização dos desenhos feitos pela interface.

Esta transformação pode ser observada, sobretudo, no algoritmo implementado para geração de desenhos de elipses, que se apoia no algoritmo de Bresenham. Este algoritmo, formulado por Elton Bresenham, implementa o desenho de linhas em dispositivos matriciais por meio de gráficos *raster*, ou seja, imagens em forma de matriz que contêm descrições de *pixel* de forma discretizada (Bresenham 1965).

As linhas desenhadas através do algoritmo desenvolvido por Bresenham são aproximações de suas equivalentes em gráficos vetoriais, geradas através da seleção de *pixels* em um *raster* n-dimensional. Esta seleção é feita de forma a minimizar o erro na construção da linha e é amplamente utilizada em Computação Gráfica devido à sua otimização por utilizar somente operações incrementais e decrementais.

3.3.1 Desenho de Elipse

A função desenvolvida para desenho de elipses se apoia no algoritmo de Bresenham por utilizar a abordagem de utilizar operações de soma e subtração para minimizar os erros nas construções das linhas e assim obter formas geométricas aproximadas em *raster*.

Conforme explicado por Kennedy (s.d.), dada uma elipse, representada na Figura 7, com centro na origem de um plano cartesiano, com eixo maior a e eixo menor$b$, sua equação geral pode ser dada por:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{22}$$

Suponha que desejamos plotar pontos $P(x_i, y_i)$ em uma matriz discreta de *pixels*, para desenhar a referida elipse. Podemos, para isso, comparar erros associados às coordenadas x e y para tais pontos. Para isso, podemos reescrever a equação geral como:

$$x^2 \cdot b^2 + a^2 \cdot y^2 - b^2 \cdot a^2 = 0 \tag{23}$$

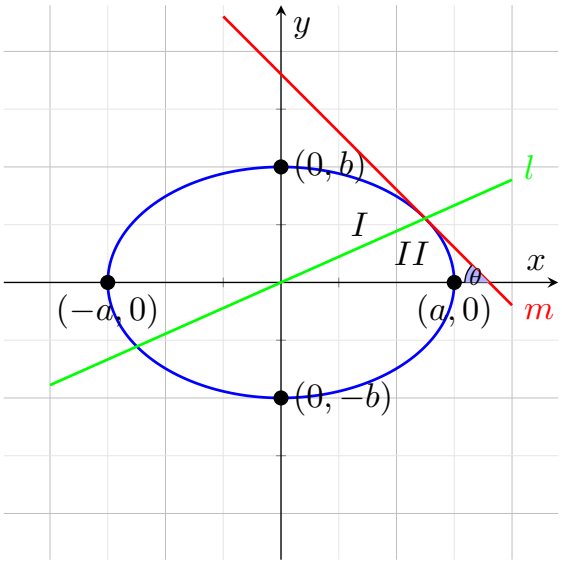


Figura 7: Representação visual de elipse centralizada nas origens do plano cartesiano. Fonte: autores.

De modo que $|x^2 \cdot b^2 + a^2 \cdot y^2 - b^2 \cdot a^2|$ passe a representar a medida do erro para um ponto $P(x_i, y_i)$.

Devido à simetria da elipse, podemos calcular a aproximação do desenho da elipse somente em relação ao primeiro quadrante do plano cartesiano em questão. Para isso, é necessário encontrar a reta tangente m à elipse no primeiro quadrante que possui inclinação $\theta = -1$, pois esta reta demarca os limites das duas regiões com variações de movimentação distintas.

Sendo dy a variação do movimento dos pontos ao longo do eixo y e dx a variação do movimento dos pontos ao longo do eixo x, a Região I é caracterizada por uma maior variação de y em relação a x, pois $\frac{dy}{dx} = \theta > -1$. Neste caso, a região se inicia no topo da elipse, ou seja, $P(0, b)$, e se prolongará até o ponto que tocará a reta tangente com inclinação $\theta = -1$. Neste intervalo, o valor de x sempre irá ser incrementado, enquanto o valor de y poderá ser decrementado ou não, a depender do erro de cada ponto da elipse.

Na região I, a condição de decisão do próximo ponto está associado ao ponto médio p_I: como as possibilidades de próximos pontos são (x_i+1, y_i) e (x_i+1, y_i-1), utilizamos o ponto médio $(x_i + 1, y_i - 0.5)$ na equação da elipse reescrita e, caso esse valor seja maior ou igual a zero, decrementamos y.

Ao alcançar a Região II, a variação em x será superior à variação em y, pois $\frac{dy}{dx} \leq -1$. Neste caso, a região continuará até o Ponto $P(a, 0)$. Neste intervalo, o valor de y sempre diminuirá, enquanto o valor de x poderá ser incrementado ou não, a depender do erro de cada ponto da elipse.

Na região *II*, a condição de decisão do próximo ponto está associado ao ponto médio p_{II}: como as possibilidades de próximos pontos são (x_i, y_i-1) e (x_i+1, y_i-1), utilizamos o ponto médio $(x_i + 0.5, y_i - 1)$ na equação da elipse reescrita e, caso esse valor seja menor ou igual a zero, incrementamos x.

Durante este processo iterativo, as variações de movimento nos dois eixos é calculada através das derivadas parciais da Equação 22, da qual obtemos:

$$\begin{aligned} \frac{\partial}{\partial x} x^2 \cdot b^2 + a^2 \cdot y^2 - b^2 \cdot a^2 &= 2xb^2 \\ \frac{\partial}{\partial y} &= 2ya^2 \end{aligned} \tag{24}$$

Assim, temos que o incremento de dx será $2xb^2$ e o decremento de y será $2ya^2$. Por fim, devido à simetria da elipse, podemos, para cada ponto do primeiro calculado do primeiro quadrante, encontrar facilmente seus simétricos e, assim, desenharmos a elipse corretamente.

3.4 Definição do Problema de Geração de *Sprites*

Nosso trabalho busca investigar os modelos treinados por Coutinho, Chaves e Chaimowicz (2025) para atacar o problema de geração de poses de *sprites* de personagens em *Pixel Art*. A seguir, formalizamos o problema. Seja $N = \{a, b, c, d\}$ os domínios que representam as possíveis poses de um personagem, o conjunto de dados consiste em entradas $x = (x_a, x_b, x_c, x_d)$ que representam imagens de um personagem em cada uma de suas poses. Um gerador GAN $G_{s \to t}$ é responsável por criar *sprites* em um domínio-alvo $t \in N$ utilizando um ou mais domínios de origem $S \subset N$. Seja x_s um conjunto de imagens do domínio de origem tal que $x_s \subset x$, a imagem de saída do gerador $\hat{x_t}$ pode ser definida como $\hat{x_t} = G_{s \to t}(x_s, t)$. Por sua vez, o discriminador D é responsável por classificar se determinada imagem é real ou falsa — isto é, criada pelo gerador.

3.5 Conjunto de Dados

Os dados de *sprites Pixel Art* de personagens são provenientes do *Pixel Art Characters Dataset* (PAC), disponibilizado por Coutinho, Chaves e Chaimowicz (2025)[4]. O conjunto agrega dados de múltiplas fontes, incluindo *assets* obtidos por meio de *Web Scraping* de bancos de dados públicos, e conta com 14.202 imagens pareadas de personagens em quatro direções: frente, costas, de lado esquerdo e de lado direito. A coleção possui, em sua maioria, personagens humanoides de tamanhos distintos. Porém, há também a presença de um pequeno número de animais, veículos

[4]Conjunto de dados PAC: `https://huggingface.co/datasets/plucksquire/pac`

e monstros. Além disso, os personagens possuem também variedade artística, o que confere ao conjunto uma grande diversidade. Como os dados de cada uma das diferentes fontes utilizadas possuíam dimensões distintas, optou-se por padronizá-los à maior dimensão presente no conjunto — 64×64 *pixels* —, por meio do preenchimento das imagens de menores dimensões com *pixels* transparentes. O conjunto foi dividido entre 12.074 (85%) exemplos para treinamento e 2.128 (15%) exemplos para teste.

3.6 Otimizações

Nesta seção, será abordada a metodologia empregada para investigar possíveis otimizações nos modelos CollaGAN e StarGAN treinados por Coutinho, Chaves e Chaimowicz (2025). Primeiro, o foco será dado ao redimensionamento de imagens através de interpolação por vizinho mais próximo durante as etapas de pré-processamento e de aumento de dados. Por fim, será discutida a quantização diferenciável de paleta.

3.6.1 Pré-processamento e Aumento de Dados

De acordo com Kumar (2025), o pré-processamento de Dados pode ser compreendido como o processo de conversão de dados brutos em um formato que algoritmos consigam processar e analisar, sendo um estágio crucial em aprendizado de máquina. O processo envolve a limpeza e a organização de dados para treinamento destes, e pode ser realizado a partir de diferentes estratégias (Z. He 2015).

O aumento de dados, por sua vez, é um conjunto de técnicas que geram artificialmente amostras de dados novas ou transformadas a partir de dados existentes, com o objetivo de aumentar diversidade e quantidade do conjunto de treino, melhorar generalização e reduzir *overfitting* em modelos de aprendizado profundo (Shorten e Khoshgoftaar 2019).

Em continuidade ao trabalho anterior de Coutinho, Chaves e Chaimowicz (2025) e a fim de se otimizar os modelos desenvolvidos pelos autores — e assim melhorar seus resultados, essa pesquisa buscou utilizar a interpolação por vizinho mais próximo para redimensionar as imagens de nosso conjunto de treinamento, originalmente com dimensões de 64×64 *pixels*. A técnica em questão foi aplicada como pré-processamento e também como forma de aumento de dados.

Nesse contexto, utilizamos um fator de redimensionamento para ampliar as imagens. Os valores testados para este parâmetro foram 2 e 3. Como pré-processamento, buscamos ampliar as imagens de 64×64 *pixels* para imagens de 128×128 ou 192×192 *pixels* antes que fossem processadas durante o treinamento dos modelos.

Por sua vez, a utilização do redimensionamento como técnica de aumento de dados foi feita de modo a redimensionar as imagens, tal como feito no primeiro caso, para, em seguida, realizar um recorte na imagem, para que o resultado final continuasse com o tamanho original. Esta operação de recorte é pseudo-aleatória, pois busca obter uma área que tenha *pixels* com opacidade maior que zero, isto é, que não sejam vazios.

Conforme será discutido adiante, nossos experimentos envolveram testar ambas as abordagens separadamente e em conjunto.

3.6.2 Quantização Diferenciável de Paleta

Segundo Coutinho, Chaves e Chaimowicz (2025), as imagens geradas pela CollaGAN original e também pelo modelo otimizado inicialmente proposto pelos autores exibiam uma ampla distribuição de cores não compatível com as imagens alvo, visto que a paleta de cor de imagens em *Pixel Art* apresenta uma distribuição limitada e discretizada de cores. Porém, no contexto abordado de geração de sugestões de *sprites* para apoiar o trabalho de artistas de jogos, é substancial que as imagens sugeridas pelo modelo sigam estritamente a paleta definida pelo usuário para evitar retrabalho. Sendo assim, os autores definiram a técnica de quantização diferenciável de paleta e a adicionaram à arquitetura da CollaGAN. Esta técnica força o modelo a gerar imagens que sigam a paleta definida como entrada. Normalmente, técnicas similares utilizam operações não-diferenciáveis para encontrar a cor da paleta de entrada menos distante do *pixel* analisado. Porém, nesses casos, as operações são incompatíveis com o cálculo de gradientes e otimizações relacionadas.

Para contornar o problema exposto e forçar a aderência das imagens sintetizadas pelo gerador à paleta de cor alvo por meio de operações diferenciáveis, os autores aplicaram a função `softmin` com mecanismo de temperatura em conjunto com um método de substituição de cor por média ponderada. Por fim, empregou-se também a perda de cobertura de paleta, para garantir que o gerador sintetize imagens que possuam a paleta alvo integralmente e, assim, evitar cenários em que são geradas imagens com ausência de uma ou mais cores desejadas.

Nesse trabalho, buscou-se seguir a linha definida pelos autores e implementamos o método da quantização diferenciável de paleta na arquitetura da StarGAN. Em seguida, serão descritas as operações que permitem o fluxo de gradientes com informações relativas à paleta alvo e a função de perda mencionada.

***Softmin* com Temperatura.** Para garantir a seleção discreta de cores durante a construção das imagens sintetizadas pelo gerador, bem como otimizações baseadas em gradientes, os autores utilizaram a técnica de recozimento simulado (i.e.

simulated annealing) aplicada à função `softmin`. Nesse método estocástico, inspirado em Termodinâmica, é utilizado o conceito de temperatura, definido por τ, responsável por controlar a suavidade da distribuição. No contexto em questão, altos valores de τ indicam uma distribuição suave, com pesos bem distribuídos. Enquanto isso, valores baixos indicam a concentração de pesos em uma cor específica, com menor distância — ideal para garantir a adesão à paleta desejada.
A função `softmin` foi escolhida pois se configura como uma opção *soft* (i.e. diferenciável) de se obter valores mínimos, em relação à sua contraparte *hard* (i.e. diferenciável), `argmin`, visto que estamos interessados em operações que permitam o fluxo de gradientes. Aqui, estamos interessados em valores mínimos pois buscamos obter a distância entre cada *pixel* e a paleta alvo, por meio da transformação dos valores destas distâncias em uma distribuição de probabilidades (e.g. valores entre [0,1], que somam 1). Dado um vetor de distâncias d, a `softmin` é calculada da seguinte forma:

$$\text{softmin}(d, \tau)_i = \frac{e^{\frac{-d_i}{\tau}}}{\sum_j e^{\frac{-d_j}{\tau}}} \tag{25}$$

A fórmula é obtida a partir da `softmax` aplicada a $-d$. Neste caso, o valor de τ é inicialmente alto (e.g. $\tau = 0.1$) e decai linearmente durante o treinamento, até alcançar valores próximos de zero — situação na qual a função `softmin` se aproxima da `argmin`.

Subtituição de Cor por Média Ponderada. Seja $I \in R^{H \times W \times C}$ uma imagem e $P \in R^{K \times C}$ a paleta alvo com K cores, representadas no espaço RGBA. A distância $L2$ entre o *pixel* I_{hw} e a cor P_k é dada por $D_{hw,k} = \|I_{hw} - P_k\|_2^2$.
Após se obter a distribuição de probabilidades via `softmin`, substitui-se a cor de cada um dos *pixels* da imagem pela média ponderada de todas as cores da paleta, utilizando como pesos as distâncias calculadas ente cada *pixel* e a paleta alvo. A imagem resultante $I_{hw}^{P\text{-quantized}}$ é obtida da seguinte forma:

$$I_{hw}^{P\text{-quantized}} = \sum_{k=1}^{K} \text{softmin}(D_{hw,k}, \tau)_k \cdot P_k \tag{26}$$

Segundo os autores, as imagens geradas no início do treinamento apresentam uma interpolação suave de cores devido aos altos valores iniciais de τ. Porém, com o decaimento de τ, eventualmente as imagens geradas convergem com uma aderência maior à paleta. Dessa forma, o `softmin` com temperatura garante que todas as cores presentes nas imagens geradas façam parte da paleta alvo. Porém, os autores constataram que, ainda assim, as imagens geradas podem não se conformar totalmente à paleta devido à discrepância entre as paletas da imagem de origem

e da imagem alvo, além da sub-utilização de cores da paleta alvo — dado que a quantização diferenciável por si só não força o gerador a utilizar toda a paleta. A fim de se mitigar o problema de sub-utilização de cores, os autores propuseram a perda de cobertura de paleta, a qual também foi utilizada.

Perda de Cobertura de Paleta. A perda de cobertura de paleta, $\mathcal{L}_{pcov}$, orienta o gerador a utilizar a paleta alvo em sua integridade. Inicialmente, é calculado para cada *pixel* I_{hw} a sua proximidade ou sua probabilidade associada a cor P_k, definida por $\alpha_{hw,k}$ e calculada via `softmin` com temperatura:

$$\alpha_{hw,k} = \frac{e^{\frac{-D_{hw,k}}{\tau}}}{\sum_{j=1}^{K} e^{\frac{-D_{hw,j}}{\tau}}} \tag{27}$$

Em seguida, calcula-se a utilização total da cor P_k através de todos os *pixels*, definida por β_k:

$$\beta_k = \sum_{h=1}^{H}\sum_{w=1}^{W}\alpha_{hw,k} \tag{28}$$

A perda de cobertura de paleta para a cor P_k, dada por $\mathcal{L}_{pcov,k}$, penaliza cores sub-utilizadas ($\beta_k < 1$):

$$\mathcal{L}_{pcov,k} = \max(1 - \beta_k, 0) \tag{29}$$

A perda de cobertura $\mathcal{L}_{pcov,k}$ para todas as K cores é, portanto, $\mathcal{L}_{pcov}$, pode ser definida como:

$$\mathcal{L}_{pcov} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{pcov,k}, \tag{30}$$

A perda de cobertura de paleta é adicionada à função objetivo original do gerador da StarGAN base (Choi et al. 2018), multiplicado a um fator $\lambda_{pcov} = 1$. Portanto, a função objetivo do gerador de nosso modelo modificado pode ser definida da seguinte forma:

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{dmn}\mathcal{L}^{f}_{dmn} + \lambda_{cyc}\mathcal{L}_{cyc} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{pcov}\mathcal{L}_{pcov} \tag{31}$$

3.7 Métricas de Avaliação

Para avaliação quantitativa, adotamos as métricas seguidas por Coutinho, Chaves e Chaimowicz (2025), a saber:

Erro Médio Absoluto (MAE). Para medir a diferença de cor, *pixel* a *pixel*, entre as imagens geradas e as imagens alvo. O MAE penaliza discrepâncias locais de posicionamento de *pixels* e inconsistências no formato da imagem. É importante observar que, no caso de testes relacionados a quantização diferenciável de paleta, um alto valor de MAE não necessariamente significa a geração de uma imagem pior, uma vez que, em decorrência da quantização, a imagem gerada pode apresentar uma paleta mais assertiva e próxima da real em detrimento de um maior distanciamento com *pixels* da imagem real.

***Fréchet Inception Distance* (FID).** Para medir a distância entre as distribuições de *features* extraídas por Inception v3 (Heusel et al. 2017) entre imagens geradas e reais. Esta métrica captura similaridade perceptual e global entre as imagens, sendo menos sensível a pequenos deslocamentos de *pixels* quando comparada ao MAE, e também à ausência de bordas nítidas nas imagens geradas.

Ambas as métricas são baseadas em distância, com valores pequenos indicando um melhor alinhamento às imagens alvo. Contudo, assim como os autores, nós também conduzimos inspeções visuais para analisar a fidelidade das imagens construídas, buscando identificar se as imagens apresentam bordas nítidas, cores e formatos bem definidos, ou ruídos, *blur* e paletas com grandes variações de cores.

4 Experimentos

Por meio de nossos experimentos, buscamos validar a nossa hipótese de que a utilização do redimensionamento de imagens, como técnica de pré-processamento e aumento de dados, e a aplicação da quantização diferenciável de paleta se configuram como potenciais otimizações para os modelos CollaGAN e StarGAN para o problema proposto por Coutinho, Chaves e Chaimowicz (2025) de geração de *sprites*.

Os modelos foram treinados com o conjunto de dados PAC com 100.000 passos de atualização no gerador e minilotes de 4 exemplos, utilizando uma GPU GeForce GTX Titan RTX 24GB 280W com 24 GB de VRAM. A taxa de aprendizado para a StarGAN foi de 0.0002, enquanto para a CollaGAN foi de $0.0001, 0.00005$ e 0.00001, a depender do teste. Utilizamos um decaimento linear da taxa de aprendizado a partir da segunda metade do treinamento (i.e. a partir de 50.000 passos). Para o Algoritmo de Otimização de Adam, utilizamos $\beta_1 = 0.5$ e $\beta_2 = 0.999$. Foi utilizado um mecanismo de *early stopping* para selecionar o modelo que obteve o melhor MAE no conjunto de teste por meio de um mecanismo de paciência, para previnir *overfitting*. Com este mecanismo de *early stopping*, os experimentos executaram, em média, por 60.000 passos ($\approx$ 17 épocas). Além

disso, como técnica de aumento de dados com probabilidade de 80% para cada personagem, foi aplicada a rotação de matiz (*hue*) em todos os testes, além de redimensionamento das imagens seguidas de um *crop* pseudo-aleatório em uma parcela dos testes. Utilizamos também a estratégia conservadora de *dropout* de entradas durante seleção de lotes (Coutinho e Chaimowicz 2024b).

Os valores de λ utilizados nas funções objetivo foram:

- CollaGAN: $\lambda_{L1} = 100$, $\lambda_{mcyc} = 10$, $\lambda_{dmn} = 10$, $\lambda_{ssim} = 1$, $\lambda_{pcov} = 1$
- StarGAN: $\lambda_{L1} = 100$, $\lambda_{dmn} = 1$, $\lambda_{rec} = 10$, $\lambda_{pcov} = 1$

Foram criados os seguintes experimentos para cada um dos modelos:

1. Modelo de base: teste único;
2. Modelo com quantização diferencial de paleta: teste único;
3. Modelo com redimensionamento de imagens: seis testes no total, utilizando 2 e 3 como fatores de redimensionamento e testando-se o redimensionamento como pré-processamento e como aumento de dados, isoladamente ou em conjunto;
4. Modelo com quantização diferencial de paleta e com o melhor caso do experimento 3: teste único.

Durante os experimentos, os modelos CollaGAN treinados com redimensionamento de imagens não convergiram com a taxa de aprendizado de 0.0001. Além disso, nesses cenários, o uso de minilotes com 4 amostras culminou em erros relacionados a esgotamento de recursos (*out of memory*). Sendo assim, reduzimos a taxa de aprendizado para 0.00001 e utilizamos minilotes com apenas uma amostra. No caso dessa arquitetura, consideramos apenas os modelos treinados com estas configurações para fins de padronização e análise justa. Os resultados serão detalhados no Capítulo 5 (Resultados).

5 Resultados

Neste capítulo, serão apresentados os resultados do presente trabalho. Inicialmente, o foco será dado aos resultados da pesquisa de usuário realizada e, em seguida, aos resultados da implementação de novas funcionalidades para o protótipo de interface de edição de *Pixel Art*. Finalmente, iremos discutir os resultados relacionados aos experimentos conduzidos para se validar as estratégias de otimização dos modelos GAN integrados ao editor.

5.1 Resultados da Pesquisa

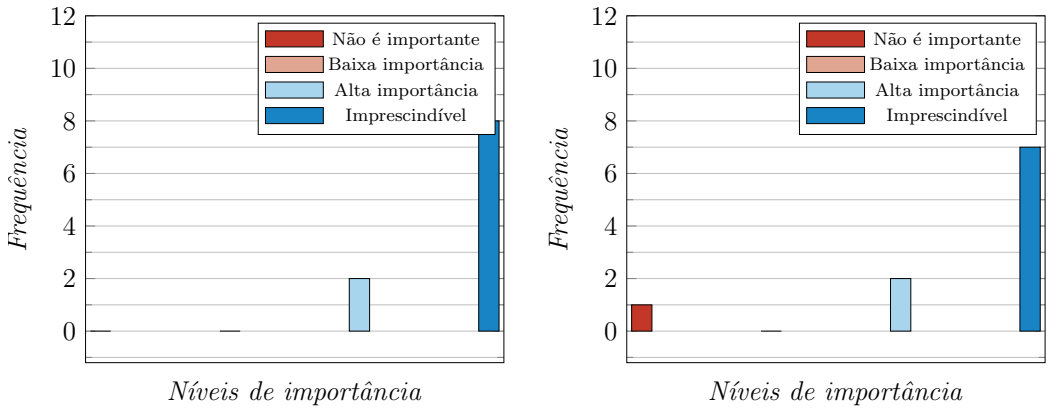
A pesquisa contou com a participação de 10 artistas de jogos, profissionais e amadores, como participantes anônimos. 9 das 10 respostas foram respondidas na versão em português do questionário, enquanto somente uma das respostas foi respondida na versão traduzida para língua inglesa. As respostas foram coletadas entre 03/01/2025 e 08/01/2025. Com base nas respostas dos participantes do questionário, foi possível organizar e interpretar os dados de forma a compreender melhor as percepções em relação às ferramentas de edição de *Pixel Art* e à utilização de modelos generativos para otimizar o processo criativo realizado por artistas de jogos.

5.1.1 Ferramentas de Edição

Os resultados da pergunta sobre ferramentas necessárias para uma interface de edição de *Pixel Art* podem ser observados na Figura 8, que agrupa gráficos de barras para a frequência de cada um dos valores possíveis ("Não é importante", "Baixa importância", "Alta importância" e "Imprescindível") para cada uma das ferramentas abordadas. Os participantes destacaram a necessidade de diversas ferramentas como espessura de pincel e borracha, seleção por área com operações associadas (rotacionar, escalar, recortar), recorte, paleta de cores utilizadas, desenhos de formas geométricas adicionais (em relação ao já implementado desenho de retângulo/quadrado) e, sobretudo, sistemas de camadas. Outras sugestões relevantes incluíram suporte a mesas digitalizadoras, movimentação e *zoom* no *canvas*, e funções específicas para animação com uso de *frames*. Os resultados refletem uma demanda por ferramentas que facilitem não apenas a criação, mas também a edição e o refinamento de *sprites*.

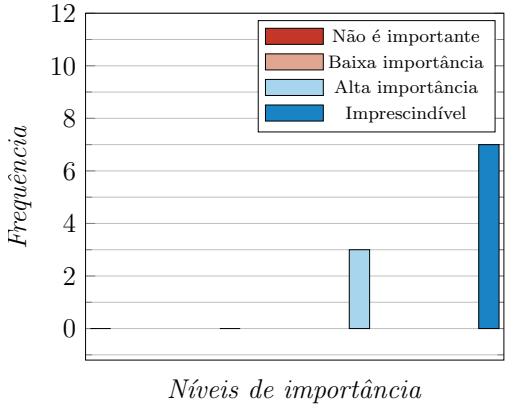
5.1.2 Automatização de Tarefas Repetitivas e Mecânicas

A maior parte dos participantes relatou frequência significativa de tarefas manuais e repetitivas durante o processo artístico, com média, mediana e moda centradas

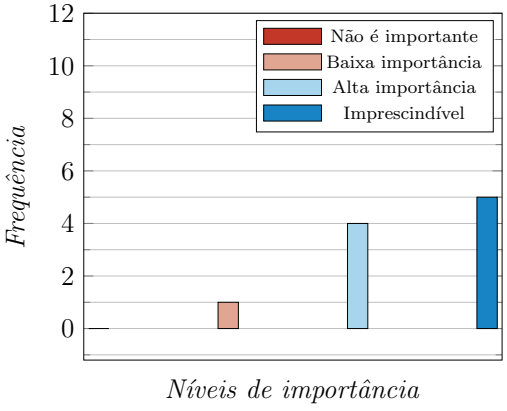


(a) Lápis

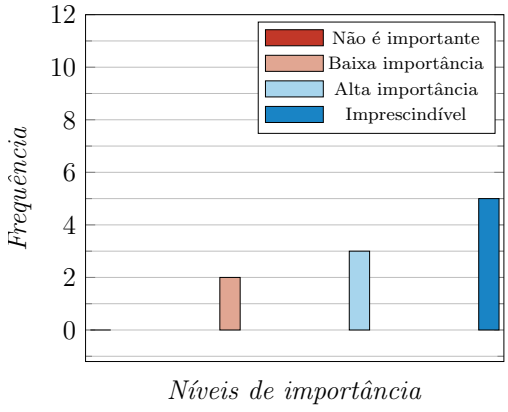
(b) Pincel



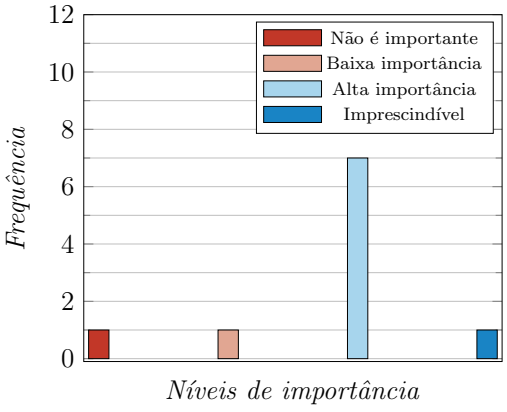
(c) Borracha



(d) Balde de tinta

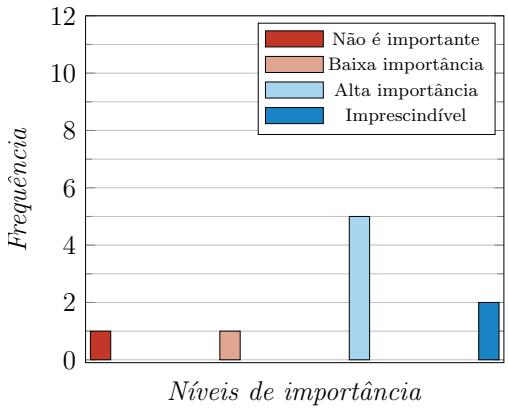


(e) Linha

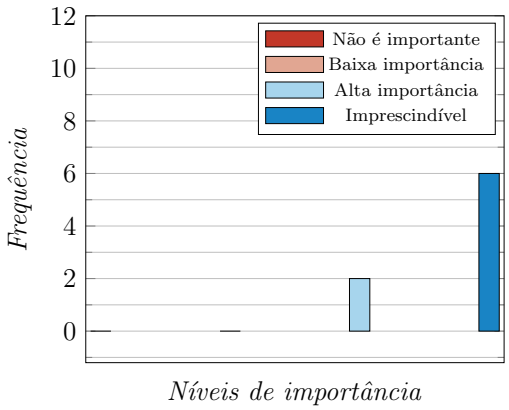


(f) Retângulo

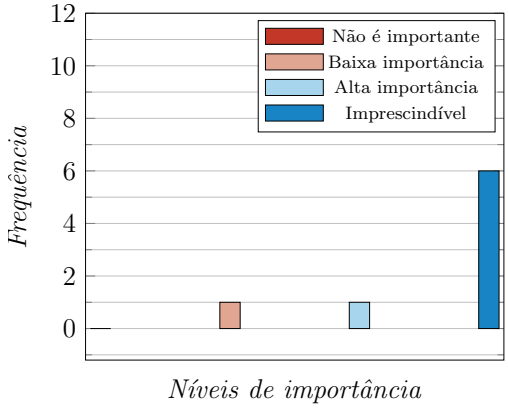
Figura 8: Gráficos de barras — frequência de resposta por ferramenta. Fonte: autores.



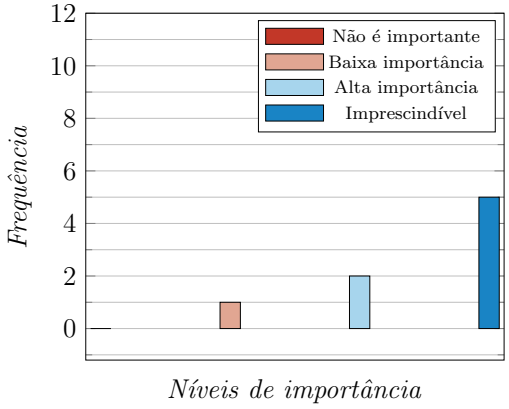
(g) Elipse



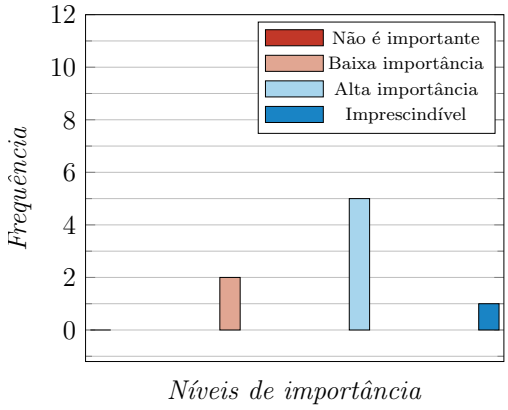
(h) *Undo*



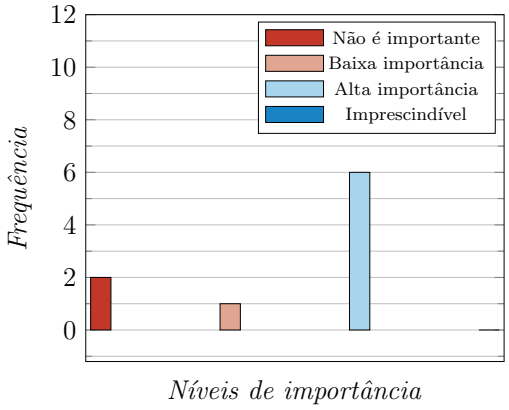
(i) *Redo*



(j) Seleção por área

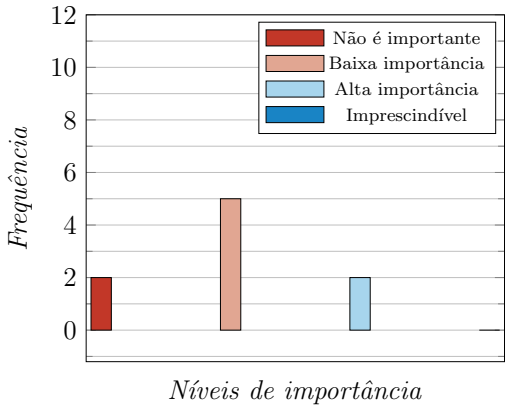


(k) Paleta de cores



(l) Cursor responsivo

Figura 8: Gráficos de barras — frequência de resposta por ferramenta. Fonte: autores.



(m) Preenchimento interno a partir de contorno

Figura 8: Gráficos de barras — frequência de resposta por ferramenta. Fonte: autores.

no valor "3" em uma escala de 1 a 4. Esse dado demonstra que, embora a criação de *Pixel Art* seja uma tarefa artística, há um espaço potencial para a automatização de atividades mecânicas e repetitivas.

As tarefas consideradas mais repetitivas incluem a criação de movimentos de animação (como interligar *keyframes* ou movimentar *pixels* gradualmente), ajustes de sombreamento e iluminação, remodelagem de personagens para diferentes posições, e ações como recolorir *pixels* e dividir *spritesheets* em imagens distintas. Esses exemplos reforçam a presença de processos mecânicos que não demandam criatividade constante, mas que consomem tempo e podem ser otimizados com ferramentas apropriadas.

Em relação ao uso de modelos generativos de inteligência artificial, a maioria dos participantes identificou oportunidades de automação, especialmente em atividades relacionadas à criação de animações, sombreamento e *dithering*. Algumas sugestões mencionaram funcionalidades já disponíveis em ferramentas como *Aseprite*, enquanto outras destacaram o uso de automação para conectar animações, ajustar sombras de acordo com a luz ou dividir *spritesheets* em imagens. Apenas um participante questionou a necessidade de automação, considerando tais atividades como parte integral do processo criativo.

Além disso, a maior parte dos participantes acredita que a utilização de modelos generativos, da forma como foi proposta no editor, pode auxiliar no trabalho artístico, especialmente na criação de silhuetas ou bases para personagens. Contudo, muitos expressaram preocupação com a presença de ruídos nas imagens geradas, que comprometem a qualidade visual e a utilidade prática das imagens

geradas. Outro ponto levantado foi o risco de que modelos generativos possam reduzir a criatividade no processo artístico. Ainda assim, a percepção predominante é de que os modelos, mesmo em estágios iniciais, podem ser úteis para economizar tempo e servir como suporte para tarefas repetitivas.

A presença de respostas citando a remodelagem de personagens em diferentes posições para construção de seus *frames* de animação ressaltam a relevância da utilização de modelos generativos como propostos pelo trabalho ao processo criativo de criação de *sprites* de personagens.

Figura 9: Nuvem de palavras obtida a partir de respostas do questionário. Fonte: autores.

A Figura 9 representa a nuvem de palavras formada a partir dos dados qualitativos da pesquisa, ou seja, das respostas das perguntas abertas do questionário. Como podemos observar, a frequência de palavras como “animação”, “camada(s)”, “seleção”, “tamanho”, “*zoom*”, “*dithering*”, “iluminar”, “luz”, “borracha”, “pincel”, “lápis” e “*frames*” ressaltam o destaque dado pelos participantes a ferramentas associadas a sistemas de camadas, ao suporte a animação com *frames*, à alteração dinâmica da espessura de lápis/pincel e de borracha, à alteração do tamanho ou da resolução do *canvas* e ao refinamento associado a sombreamento e

iluminação.

Além disso, a presença da palavra "interessante" sugere um padrão de interesse pela proposta da interface, apesar de que a presença da palavra "borrado" sugerir que a presença de ruídos nas imagens geradas pelos modelos GAN insere-se como um fator crítico do editor e como um potencial obstáculo ao uso efetivo da ferramenta por artistas de jogos.

5.2 Novas Funcionalidades Para o Editor

Nesta seção, será descrita a implementação das seguintes funcionalidades para o editor:

- Botões de desfazer alteração (*undo*) e refazer alteração desfeita (*redo*);
- Desenho de elipse preenchido;
- Paleta de cores utilizadas;
- Cursor responsivo.

5.2.1 Botões *Undo* e *Redo*

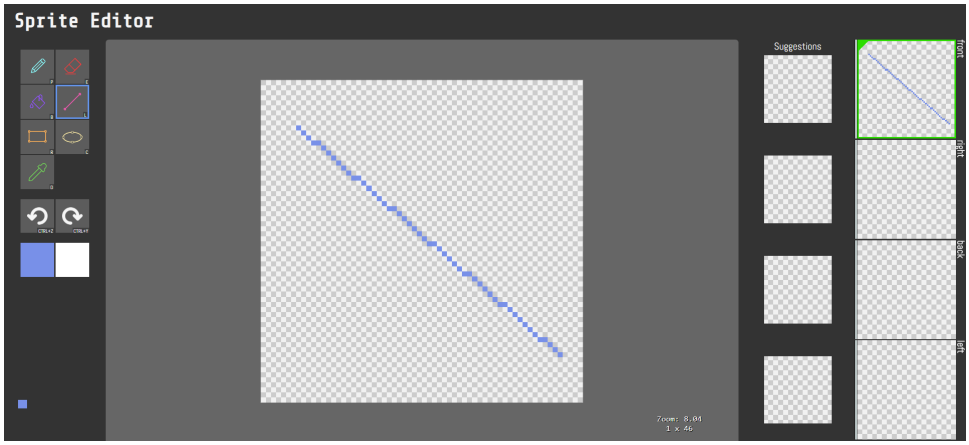
Ao realizar uma alteração no *canvas*, é possível utilizar o botão *Undo* para desfazer a última alteração na pilha de alterações. A Figura 10a demonstra o desenho de uma linha azul e, ao clicar no botão *Undo*, com bordas destacadas na lateral esquerda da interface, a alteração é apagada do *canvas*, como demonstrado na Figura 10b. O resultado da ação é notado na Figura 10c. Porém, é possível, também, refazer alterações desfeitas por meio do botão *Redo*. A Figura 11a exemplifica o clique neste botão, com bordas destacas na lateral esquerda da interface. Podemos ver na Figura 11b o resultado da operação: a recuperação da linha azul.

5.2.2 Desenho de Elipse Preenchido

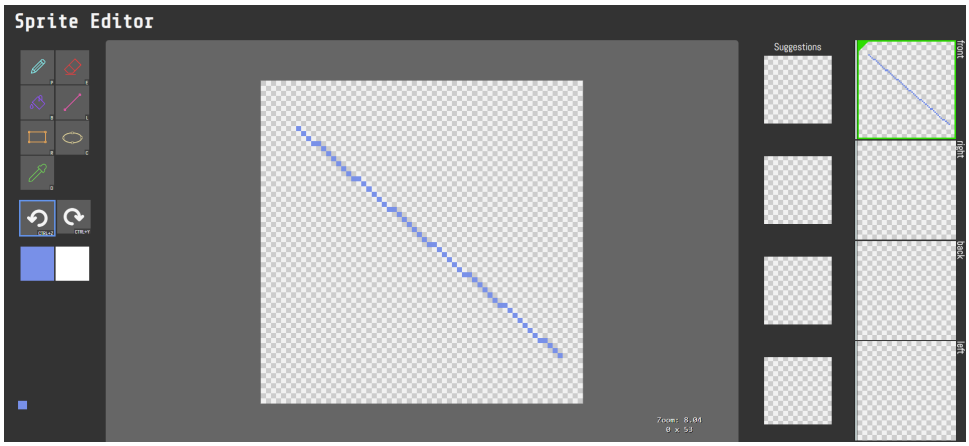
A Figura 12a mostra o botão de desenho de elipse selecionado. Após selecionado, é possível clicar no *canvas* com o botão esquerdo ou direito do *mouse* na região desejada e arrastar o cursor para que o desenho da elipse seja feito de forma dinâmica, conforme demonstrado pela Figura 12b.

5.2.3 Paleta de Cores Utilizadas

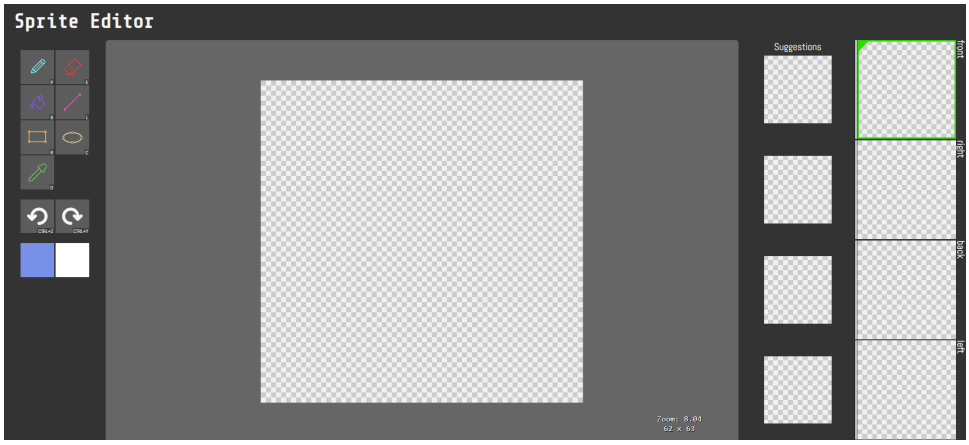
Ao criarmos uma nova elipse, de cor rosa, como na Figura 13a, é possível visualizar, no canto inferior esquerdo, que uma nova cor foi adicionada à paleta de cores utilizadas no desenho atual. Se apagarmos a linha azul, como na Figura 13b, é possível notar, também, a paleta de cores deixa de constar a cor azul utilizada.



(a) Desenhando linha

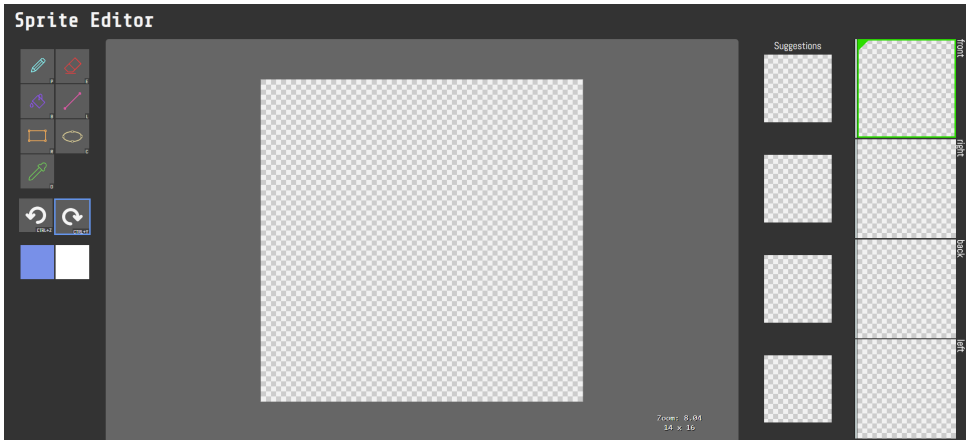


(b) Selecionando botão *Undo*

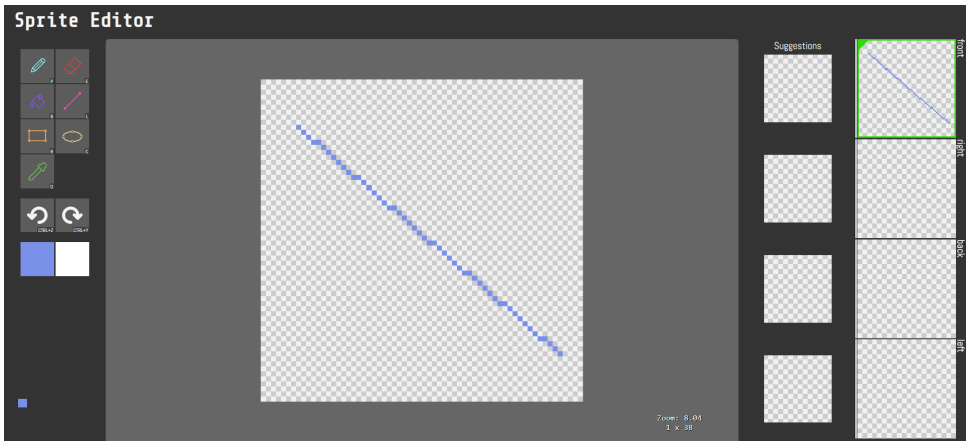


(c) Resultado da ação

Figura 10: *Undo*

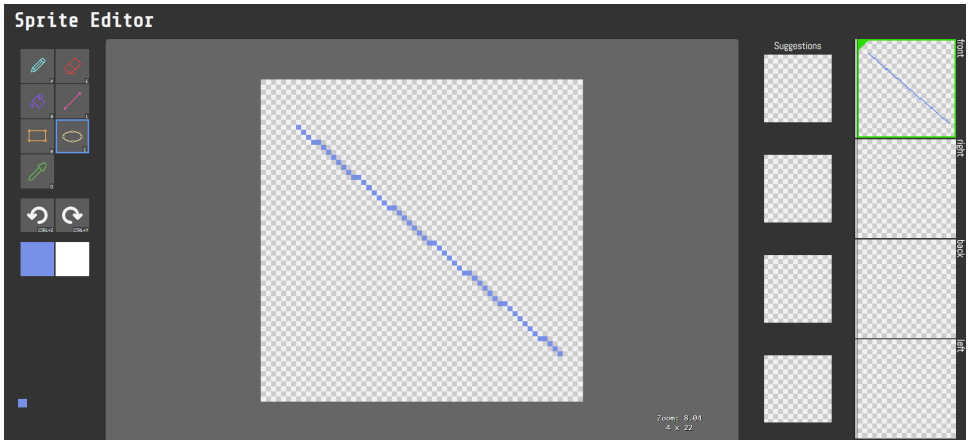


(a) Selecionando botão *Redo*

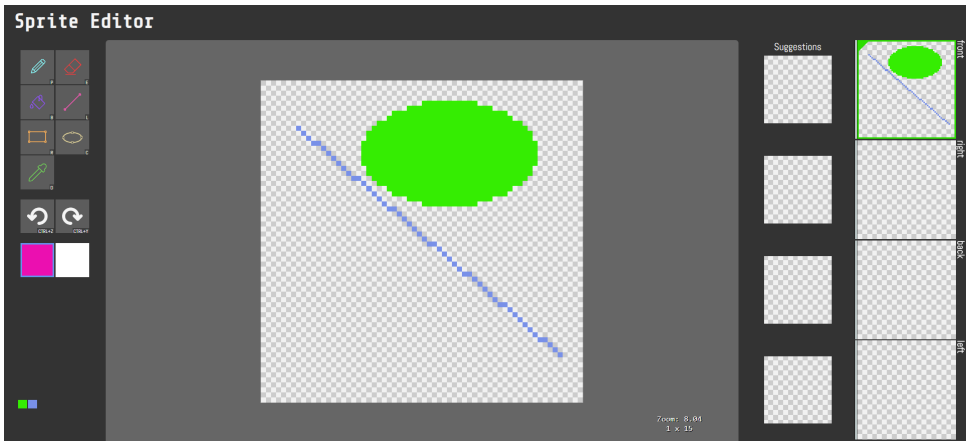


(b) Resultado da ação

Figura 11: *Redo*. Fonte: autores.

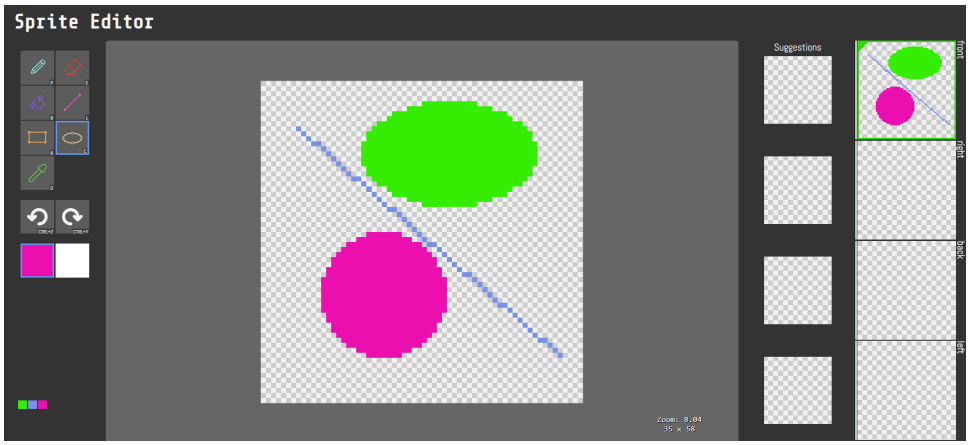


(a) Selecionando botão de elipse

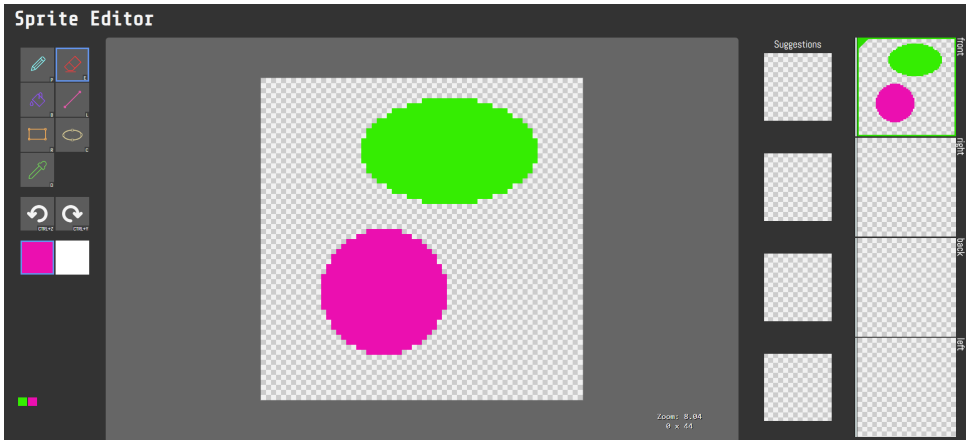


(b) Resultado da ação

Figura 12: Desenho de elipse preenchido. Fonte: autores.

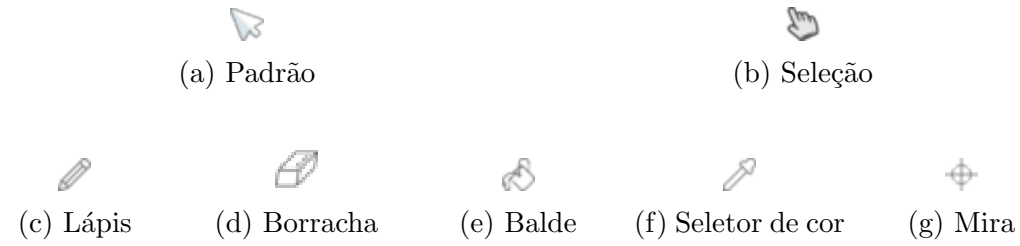


(a) Adição de nova cor à paleta



(b) Remoção de cor da paleta

Figura 13: Adição e remoção de cor da paleta. Fonte: autores.



(a) Padrão (b) Seleção

(c) Lápis (d) Borracha (e) Balde (f) Seletor de cor (g) Mira

Figura 14: Ícones do cursor. Fonte: autores.

5.2.4 Cursor Responsivo

O cursor foi adaptado para ser responsivo a depender da ferramenta de edição selecionada ou da área sobre a qual o mouse está pairando. Caso o mouse esteja sobre uma área com a qual é possível interagir, como clicar, a seta padrão, demonstrada na Figura 14a, se altera para o cursor de seleção da Figura 14b. Cada uma das ferramentas cujo funcionamento altera o estado atual do *canvas* possui seu próprio ícone de cursor, a observar pela Figura 14.

5.3 Resultados dos Experimentos

Nesta seção, serão descritos os resultados dos experimentos conduzidos nos modelos treinados, detalhados anteriormente. Primeiro, serão analisados os resultados relacionados dos modelos CollaGAN, e, em seguida, dos modelos StarGAN. A análise comparativa dos experimentos de cada arquitetura será feita na seguinte ordem:

1. Comparação entre modelo de base e modelo com quantização diferenciável de paleta e perda de cobertura de paleta;
2. Comparação entre casos com pré-processamento via redimensionamento de imagens, variando o fator de redimensionamento entre 2 e 3;
3. Comparação entre casos com aumento de dados via redimensionamento de imagens, variando o fator de redimensionamento entre 2 e 3;
4. Comparação entre modelo com somente quantização diferenciável de paleta e perda de cobertura de paleta com modelo com essas otimizações, porém com incrementos provenientes do melhor caso entre os itens 2 e 3.
5. Comparação entre os melhores modelos do item 1 e do item 4.

5.3.1 CollaGAN

A Tabela 1 mostra os resultados dos valores das métricas FID e MAE para cada um dos experimentos executados para modelos CollaGAN. O experimento com melhores resultados está destacado em negrito.

Tabela 1: Resultados de experimentos com modelos CollaGAN

	FID	MAE
Base	2.441	0.0549
Paleta	3.573	0.0826
Pré-processamento (2)	2.538	0.0512
Pré-processamento (3)	**2.235**	**0.0489**
Aumento de dados (2)	3.153	0.0613
Aumento de dados (3)	4.377	0.0654
Paleta + pré-processamento (2)	3.319	0.0685
Paleta + pré-processamento (3)	2.684	0.0640

O resultado da comparação dos modelos de base e com quantização diferenciável de paleta é contraintuitivo, pois o primeiro obteve melhores métricas em relação ao segundo, além de uma presença menor de ruídos nas imagens geradas.

Os modelos com o aumento de dados via redimensionamento de imagens não apresentaram bons resultados. As imagens geradas com ambos os fatores apresentam uma alta frequência de ruídos e borrões. Contudo, apresentam o formato externo, ou o contorno, correto dos personagens — assim como nos outros modelos.

Como pode ser visto na Tabela 1, o modelo com melhores resultados foi aquele treinado com redimensionamento como pré-processamento, com fator igual a 3. Quando comparado ao modelo análogo com fator igual a 2, o primeiro apresenta também melhores resultados. Visualmente, esse caso também apresenta imagens menos borradas, mas ainda com presença de ruídos. Portanto, a ampliação de imagens utilizada para fins de pré-processamento de dados pode oferecer melhores métricas e, consequentemente, melhores resultados quando o modelo da arquitetura CollaGAN é treinado com minilotes com uma amostra única e com uma taxa de aprendizado equivalente a 0.00001.

Figura 15: Sintetização de imagem por CollaGAN. Fonte: autores.

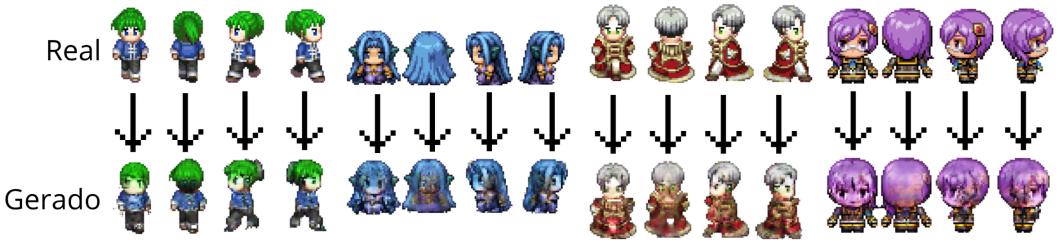


Figura 16: Imagens de teste de modelo CollaGAN com pré-processamento por redimensionamento de imagens com fator 3. Fonte: autores.

A Figura 15 exemplifica a sintetização de imagem de um personagem no domínio "frente" feita com o melhor modelo da CollaGAN mencionado anteriormente, a partir de imagens nos domínios "trás", "esquerda" e "direita". Por sua vez, a Figura 16 expõe algumas das imagens de teste geradas por esse modelo em questão.

Além disso, notamos que adicionar o pré-processamento com fator 3 melhorou o modelo com otimizações de paleta, obtendo-se assim melhores imagens quando comparado com o modelo com as otimizações de paleta utilizadas isoladamente. Os resultados também indicam que o pré-processamento favorece mais a métrica FID, ou seja, a avaliação global das imagens geradas dos personagens, em detrimento da avaliação *pixel* a *pixel* via MAE.

5.3.2 StarGAN

A Tabela 2 mostra os resultados dos valores das métricas FID e MAE para cada um dos experimentos executados para modelos StarGAN. O experimento com melhores resultados está destacado em negrito.

Tabela 2: Resultados de experimentos com modelos StarGAN

	FID	MAE
Base	3.167	0.0175
Paleta	**2.344**	**0.0161**
Pré-processamento (2)	3.647	0.0185
Pré-processamento (3)	3.817	0.0171
Aumento de dados (2)	4.735	0.0230
Aumento de dados (3)	5.873	0.0255
Paleta + pré-processamento (2)	4.196	0.0199
Paleta + pré-processamento (3)	3.358	0.0178

Como é possível notar na Tabela 2, o modelo com as otimizações relacionadas à paleta, sem o redimensionamento de imagens, obteve os melhores resultados:

FID de 2.344 e MAE de 0.0161. Além disso, houve uma maior convergência de treinamento em relação ao modelo de base, visto que o conjunto de teste obteve métricas mais próximas daquelas obtidas pelo conjunto de treinamento. Durante a inspeção visual, foi observado que as imagens de teste geradas pelo modelo com quantização diferenciável de paleta apresentaram uma menor presença de artefatos visuais. As imagens geradas indicam que há uma maior conservação de detalhes dos personagens para fator 3.

Figura 17: Sintetização de imagem por StarGAN. Fonte: autores.

Figura 18: Imagens de teste de modelo StarGAN com otimizações de paleta. Fonte: autores.

A Figura 17 exemplifica a sintetização de imagem de um personagem no domínio "frente" feita com o melhor modelo da StarGAN mencionado anteriormente (i.e.

com quantização diferenciável de paleta e adição da perda de cobertura de paleta na função objetivo), a partir da imagem de origem e dos rótulos do domínio de origem e do domínio alvo. Por sua vez, a Figura 18 expõe algumas das imagens de teste geradas por esse modelo em questão.

Novamente, os modelos treinados com aumento de dados via redimensionamento de imagens, seguido de recorte pseudoaleatório, obtiveram resultados piores em ambos conjuntos de treino e teste, com uma convergência pouco acentuada e maiores variações durante os passos de treinamento. A inspeção visual indica que com fatores maiores (e.g. 3), há uma maior presença de *blurr*, o que pode significar que a utilização de imagens de *sprites* de dimensões maiores em modelos StarGAN que utilizam a técnica em questão pode dificultar a sintetização de imagens com informações discretizadas, conforme desejado em *Pixel Art*.

Os resultados também indicam que, nos modelos StarGAN em que há redimensionamento de imagens, o fator 2 obtém, no geral, melhores resultados de FID, enquanto o fator 3 obtém menores MAE. Por fim, o único modelo com redimensionamento que obteve melhores métricas em relação ao modelo de base foi o modelo treinado pré-processamento com fator igual 3 — que foi o melhor modelo para a arquitetura CollaGAN, e que obteve melhor valor de MAE em relação ao modelo de base.

6 Conclusão

Os resultados da pesquisa de usuário conduzida indicam que os artistas de jogos participantes se deparam com uma frequência significativa com tarefas repetitivas e mecânicas. Além disso, a maior parte acredita que a utilização de modelos generativos de arquiteturas *GANs* aplicados à tradução de imagem-para-imagem pode ser uma ferramenta útil para otimizar o processo de criação de *Pixel Art*. A redução do tempo dedicado a tarefas repetitivas e mecânicas, como a criação de animações, remodelagem de *sprites* e reajustes de iluminação e sombreamento é vista como um benefício do uso dos modelos como proposto pela interface analisada.

É importante observar que para o público-alvo é substancial garantir ao artista o controle criativo e artístico de sua obra. O estudo sugere que a adoção desta abordagem pode ser ainda mais promissora com uma eventual minimização dos ruídos em imagens geradas pelos modelos.

Em relação às técnicas implementadas para as arquiteturas estudadas, nossos resultados mostraram que modelos StarGAN se beneficiam da aplicação da quantização diferenciável da paleta e da perda de cobertura de paleta na função objetivo de seu gerador, ambas otimizações propostas inicialmente por Coutinho, Chaves e Chaimowicz 2025. Modelos treinados com estas modificações superam o modelo de base e obtêm melhores métricas FID e MAE. Mais ainda, as imagens sintetizadas por seus geradores apresentam maior nitidez e menos ruídos de alta frequência. Ainda no contexto da StarGAN, a utilização da interpolação por vizinho mais próximo como aumento de dados não demonstrou bons resultados, diferente de sua utilização como etapa de pré-processamento, cujo modelo utilizado com fator de redimensionamento igual a 3 obteve um menor MAE em relação ao modelo base.

Em relação aos modelos CollaGAN, notamos que a aplicação das otimizações de paleta, com os hiperparâmetros e configurações utilizadas, não superaram o modelo de base. Porém, neste cenário, o pré-processamento com fator de redimensionamento igual a 3 potencializou os resultados dos modelos. Além disso, novamente notamos que o redimensionamento como técnica de aumento de dados seguido de recorte pseudoaleatório não pode ser considerada uma otimização válida para modelos desta arquitetura.

Sendo assim, podemos concluir que duas das três técnicas investigadas (i.e. A quantização diferenciável de paleta em conjunto com a perda de cobertura de paleta e o pré-processamento de Imagens através de redimensionamento por interpolação por vizinho mais próximo) se configuram como notáveis otimizações para diferentes cenários envolvendo ambas as arquiteturas estudadas. Porém, para análises mais assertivas sobre os impactos das técnicas abordadas, são necessários novos estudos que abordem a variação de hiperparâmetros e de outras condições de treinamento.

Referências

Baldwin, Alexander et al. (2017). "Towards pattern-based mixed-initiative dungeon generation". Em: *Proceedings of the 12th International Conference on the Foundations of Digital Games*. FDG '17. Hyannis, Massachusetts: Association for Computing Machinery. ISBN: 9781450353199. DOI: 10.1145/3102071.3110572. URL: https://doi.org/10.1145/3102071.3110572.

Bresenham, Jack Elton (1965). "Algorithm for computer control of a digital plotter". Em.

Choi, Yunjey et al. (2018). "StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation". Em: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797. DOI: 10.1109/CVPR.2018.00916.

Coutinho, Flávio e Luiz Chaimowicz (2022a). "Generating Pixel Art Character Sprites using GANs". Em: *Anais do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*. Natal/RN: SBC, pp. 61–66. URL: https://sol.sbc.org.br/index.php/sbgames/article/view/23456.

— (out. de 2022b). "On the Challenges of Generating Pixel Art Character Sprites Using GANs". Em: *"Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment"*. Vol. 18. AAAI Press, pp. 87–94. DOI: 10.1609/aiide.v18i1.21951. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/21951.

— (2024a). ""Pixel art character generation as an image-to-image translation problem using GANs"". Em: *Graphical Models* 132, p. 101213. ISSN: 1524-0703. DOI: https://doi.org/10.1016/j.gmod.2024.101213. URL: https://www.sciencedirect.com/science/article/pii/S1524070324000018.

— (2024b). "A Missing Data Imputation GAN for Character Sprite Generation". Em: *Anais do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital*. Manaus/AM: SBC, pp. 436–455. DOI: 10.5753/sbgames.2024.241116. URL: https://sol.sbc.org.br/index.php/sbgames/article/view/32329.

Coutinho, Flávio, Lucas G.S. Chaves e Luiz Chaimowicz (2025). "Imputation of Missing Pixel Art Character Poses with Differentiable Palette Quantization". Em: *Entertainment Computing* 55. URL: https://doi.org/10.1016/j.entcom.2025.101021.

Gonzalez, Rafael e Richard Woods (2017). ""Digital Image Processing"". Em: "Fourth, Global". "Pearson Education Limited". Cap. "Digital Images Fundamentals".

Goodfellow, Ian et al. (2014). ""Generative Adversarial Nets"". Em: *"Advances in neural information processing systems"*. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

He, Kaiming et al. (2015). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. arXiv: 1502.01852 [cs.CV]. URL: https://arxiv.org/abs/1502.01852.

He, Zengyou (jan. de 2015). "Data Mining for Bioinformatics Applications". Em: *Data Mining for Bioinformatics Applications*, pp. 1–83.

Heusel, Martin et al. (2017). "GANs trained by a two time-scale update rule converge to a local nash equilibrium". Em: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., pp. 6629–6640. ISBN: 9781510860964. DOI: https://doi.org/10.48550/arXiv.1706.08500. arXiv: 1706.08500 [cs.LG]. URL: https://arxiv.org/abs/1706.08500.

Isola, Phillip et al. (jul. de 2017). ""Image-to-Image Translation with Conditional Adversarial Networks"". Em: *"2017 IEEE Conference on Computer Vision and Pattern Recognition"*, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632.

Kennedy, John (s.d.). *"A Fast Bresenham Type Algorithm For Drawing Ellipses"*.

Khalifa, Ahmed et al. (2020). "PCGRL: procedural content generation via reinforcement learning". Em: *"AIIDE'20: Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment"*. AIIDE'20. AAAI Press. ISBN: 978-1-57735-849-7. arXiv: 2001.09212 [cs.LG]. URL: https://arxiv.org/abs/2001.09212.

Kumar, L. (jan. de 2025). "Predicting Parkinson's: analyzing patterns with data and analytics". Em: pp. 153–185. ISBN: 9780443298882. DOI: 10.1016/B978-0-443-29888-2.00005-2.

Labes, Emerson Moisés (1998). *"Questionario: Do Planejamento À Aplicação Na Pesquisa"*. "Grifos". ISBN: 858631109X.

Lee, Dongwook et al. (2019). "CollaGAN: Collaborative GAN for Missing Image Data Imputation". Em: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2482–2491. DOI: 10.1109/CVPR.2019.00259.

Liu, Yize (jul. de 2024). "Research On Game Development and Revenue Based on Generative Artificial Intelligence: A Case Study of Netease". Em: *Journal of Education, Humanities and Social Sciences* 35, pp. 434–440. DOI: 10.54097/51m2s794.

Pang, Yingxue et al. (2021). "Image-to-Image Translation: Methods and Applications". Em: *IEEE Transactions on Multimedia* 24, pp. 3859–3881. DOI: 10.1109/TMM.2021.3109419.

Preece, Jennifer, Helen Sharp e Yvonne Rogers (2002). *"Interaction Design: Beyond Human-Computer Interaction"*. "John Wiley & Sons", p. 223. ISBN: 978-0471492788.

Shorten, Connor e Taghi M Khoshgoftaar (jul. de 2019). ""A survey on Image Data Augmentation for Deep Learning"". Em: *"Journal of Big Data"* 6.1, "60". DOI: https://doi.org/10.1186/s40537-019-0197-0.

Silber, Daniel (2015). *Pixel Art for Game Developers*. A K Peters/CRC Press, pp. xv, 90, 252. ISBN: 978-1482252309.

Summerville, Adam et al. (2018). "Procedural Content Generation via Machine Learning (PCGML)". Em: *IEEE Transactions on Games* 10.3, pp. 257–270. DOI: 10.1109/TG.2018.2846639.

Togelius, Julian, Noor Shaker e Mark J. Nelson (2016). "Introduction". Em: *Procedural Content Generation in Games*. Cham: Springer International Publishing, pp. 1–15. ISBN: 978-3-319-42716-4. DOI: 10.1007/978-3-319-42716-4_1. URL: https://doi.org/10.1007/978-3-319-42716-4_1.

Ulyanov, Dmitry, Andrea Vedaldi e Victor Lempitsky (2017). *Instance Normalization: The Missing Ingredient for Fast Stylization*. arXiv: 1607.08022 [cs.CV]. URL: https://arxiv.org/abs/1607.08022.

Zhu, Jun-Yan et al. (2017). "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". Em: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.