Geração de Arte de Personagens 2D em Jogos Eletrônicos Através de Inteligência Artificial

Pedro H. P. Rosário[1], **Leonardo B. Estácio**[1]

[1]IFSC – Instituto Federal de Santa Catarina – Câmpus Lages
R. Heitor Villa Lobos, 225 - São Francisco, Lages - SC, 88506-400

pedro.henric.rosario@gmail.com, leonardo.bravo@ifsc.edu.br

Abstract. *Currently, the creation of art for 2D characters in electronic games is done manually and demands a significant amount of human and financial resources. With recent advancements in diffusion models, there is an opportunity for greater integration of artificial intelligence into artistic development, aiming to streamline the process and reduce costs. This article aims to demonstrate the functional integration of image generation through diffusion models in the development of 2D character art for electronic games, including the necessary steps to turn the generated image into a usable and animatable artistic asset. An exploratory bibliographic research approach was conducted, followed by the practical application of techniques and technologies most suitable for this context. Satisfactory results were achieved in generating high-quality characters of various types and styles. Finally, successful integration and animation of the generated character within game development tools, such as Unity, were demonstrated.*

Resumo. *Atualmente, a criação de arte para personagens 2D de jogos eletrônicos é realizada de forma manual e demanda uma quantidade significativa de recursos humanos e financeiros. Diante dos recentes avanços nos modelos de difusão, surge a oportunidade de maior integração de inteligencia artificial com o desenvolvimento artístico, visando otimizar o processo e reduzir custos. Este artigo almeja, portanto, demonstrar a integração funcional de geração de imagens por meio de modelos de difusão no desenvolvimento de arte de personagens 2D para jogos eletrônicos, incluindo os passos necessários para transformar a imagem gerada em um ativo artístico utilizável e animável. Foi realizado uma pesquisa de abordagem bibliográfica exploratória e, posteriormente, a aplicação prática prática das técnicas e tecnologias mais adequadas pra este contexto. Houve resultados satisfatórios na geração de personagens de alta qualidade de diferentes tipos e estilos. Por fim, foi evidenciado o sucesso na integração e animação do personagem gerado dentro do contexto de ferramentas de desenvolvimento de jogos, como Unity.*

1. Introdução

A arte é um elemento que assume um papel de destaque na promoção da imersão em jogos eletrônicos, abrangendo elementos como música, efeitos sonoros, enredo, representações visuais de personagens e objetos no universo virtual. Os desenvolvedores frequentemente empregam esses recursos de maneira estratégica para aumentar o engajamento do jogador

e guia-lo a vivenciar experiencias de jogos variadas, conforme foi visionado pela equipe criativa. Além disso, uma direção artística realizada com êxito durante o processo de desenvolvimento desempenha um papel crucial em causar uma impressão inicial positiva a potenciais jogadores, podendo ser a diferença entre adquirirem o produto ou não (Schell, 2008).

Devido a significativa importância que a arte representa em videogames, é de considerável valor para empresas de desenvolvimento de jogos e pesquisadores da área aprimorar os meios de geração artística. A direção artística frequentemente implica em investimentos substanciais, tanto em termos de tempo quanto de recursos financeiros, ao longo do ciclo de desenvolvimento de um jogo eletrônico (Keith, 2010). Em questão de recursos humanos, empresas podem enfrentar desafios para adquirir profissionais com o necessário talento criativo e experiência artística para assegurar a viabilidade comercial e qualidade do produto final, especialmente empresas *indie*[1]. Como resultado, diversas técnicas e tecnologias inovadoras são continuamente investigadas e implementadas com o intuito de maximizar a qualidade artística e minimizar os custos associados aos recursos envolvidos.

No atual cenário do ano de 2023 na indústria de desenvolvimento de jogos eletrônicos, o processo de criação artística envolve uma combinação de técnicas tradicionais e diferentes tipos de tecnologias. Empresas de desenvolvimento de jogos frequentemente dispõem de equipes compostas por artistas que fazem uso de ferramentas de modelagem 3D, pintura digital e animação para criar personagens, cenários e objetos (Keith, 2010). Para gerar diferentes tipos de conteúdo para um jogo eletrônico, pode ser utilizado também princípios de geração de conteúdo procedural (PCG) (Togelius et al., 2011), que representa a criação automatizada de conteúdo por meio de algoritmos, muitos dos quais incorporam técnicas de Inteligência Artificial (IA) (Volz et al., 2018; Hollingsworth and Schrum, 2019; Park et al., 2023).

As Redes Adversárias Generativas (Generative Adversarial Networks, ou GANs) (Creswell et al., 2018) e os modelos de difusão são técnicas de Inteligência Artificial (IA) que podem ser utilizadas para gerar imagens sintéticas. As GANs consistem em duas redes neurais adversárias: um gerador, que aprende a criar imagens mais realistas, e um discriminador, que se torna mais eficiente em distinguir imagens geradas das reais durante o treinamento conjunto da rede. A aplicação de GANs pode ser observada na geração de imagens no contexto de desenvolvimento de jogos eletrônicos, com resultados variados (Horsley and Perez-Liebana, 2017) (Hong et al., 2019) (Serpa and Rodrigues, 2019). Por sua vez, os modelos de difusão empregam um processo iterativo para difundir o ruído (Sohl-Dickstein et al., 2015) na imagem original, gerando uma sequência de imagens cada vez mais próximas da imagem final desejada e aprendendo durante o processo. Com os mais recentes avanços técnicos e tecnológicos na utilização de modelos de difusão (Dhariwal and Nichol, 2021) (Ramesh et al., 2022) (Saharia et al., 2022) (Rombach et al., 2022), emerge uma oportunidade para a aplicação dos avanços desta técnica na criação artística para jogos.

O objetivo geral desse trabalho é integrar funcionalmente os avanços da geração de imagens com modelos de difusão no desenvolvimento de arte de personagens 2D para

[1]Indie: termo utilizado no contexto de desenvolvimento para se referir a empresas ou jogos que possuem um processo de desenvolvimento com pouco ou sem apoio financeiro e com número limitado de integrantes

jogos eletrônicos. A proposta deste trabalho é explorar a utilização de inteligencia artificial para a geração de arte, com ênfase na representação de personagens, visando diminuir a intervenção humana e consequentemente potenciais custos de desenvolvimento. Cabe ressaltar que não se insere no escopo deste projeto criar novos modelos ou algoritmos de inteligência artificial, limitado-se a utilizar tecnologias de inteligência artificial já existentes.

Dentre os objetivos específicos estão:

- Conduzir uma pesquisa bibliográfica exploratória em artigos, livros e sites, com o intuito de levantar conceitos práticos e teóricos pertinentes à aplicação de inteligência artificial na geração de personagens 2D de jogos eletrônicos;
- Demonstrar a aplicação de tecnologias e técnicas que viabilizarão a criação de ativos artístico de personagens 2D em conjunto com inteligência artificial, permitindo que o mesmo possa ser animado e com isso, seu potencial emprego no decorrer do ciclo de desenvolvimento de um jogo eletrônico;
- Levantar os principais obstáculos e desafios tecnológicos encontrados durante a utilização dessas técnicas no desenvolvimento de arte para personagens de jogos eletrônicos.

Este trabalho está organizado da seguinte forma: Na seção 1 encontra-se a introdução, onde é fornecida uma visão geral do tema de pesquisa, contextualizando o problema a ser abordado e destacando os objetivos e hipóteses do estudo. Na seção 2, a metodologia, onde é realizado o esclarecimento da forma e etapas em que este trabalho será realizado. Na seção 3, o referencial teórico, onde é abordado mais detalhadamente artigos e trabalhos que deram embasamento a este artigo. Na Seção 4, o desenvolvimento, onde estão descritos os passos realizados referente a implementação prática e de natureza de pesquisa aplicada. Na seção 5, a conclusão, onde estão presentes as considerações finais em relação a este trabalho.

2. Metodologia

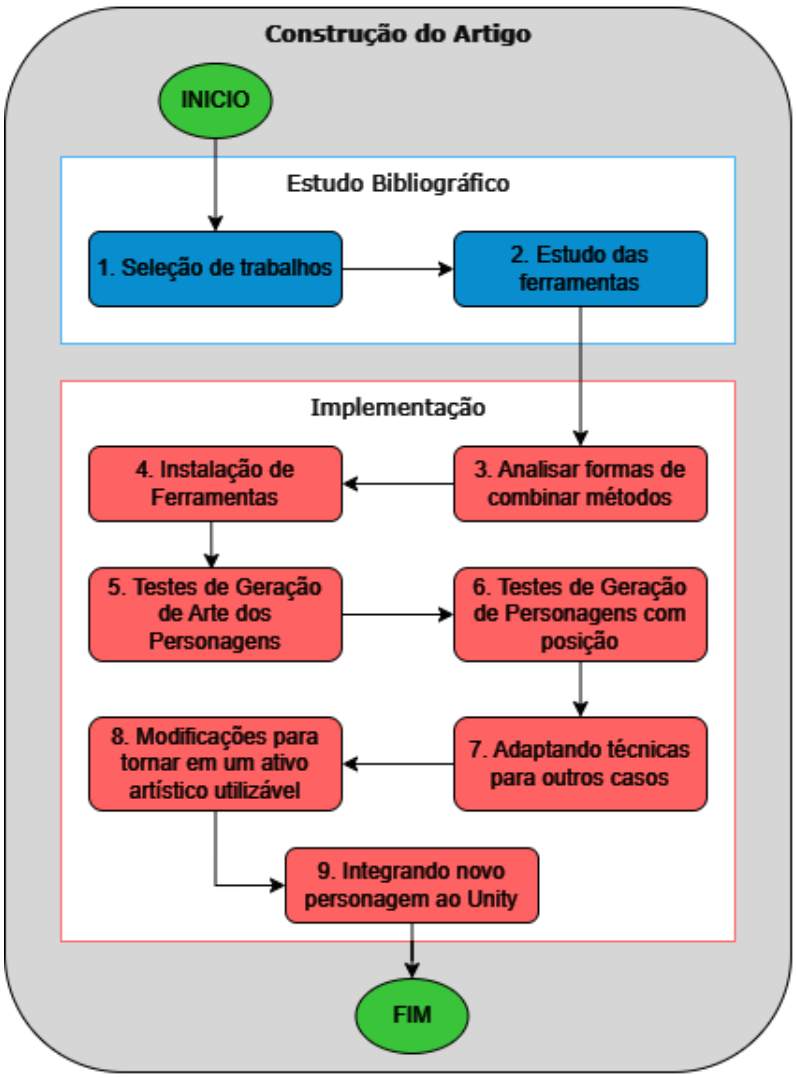
O trabalho é composto por duas etapas (Figura 1). A primeira etapa representa o Estudo Bibliográfico, que pode ser caracterizada como pesquisa bibliográfica e é sub-dividida nos passos de “1. Seleção de Trabalhos” e “2. Estudo das Ferramentas e métodos”. A segunda etapa representa a Implementação, que pode ser caracterizada como pesquisa aplicada e é sub-dividida nos passos de “3. Analisar formas de combinar métodos”, “4. Instalação de Ferramentas”, “5. Testes de Geração de Arte dos Personagens”, “6. Teste de Geração de Personagens com Posição”, “7. Adaptando Técnicas para outros casos de uso”, “8. Modificações para tornar em um ativo artístico utilizável” e por fim “9. Integrando novo personagem ao Unity”.

Do ponto de vista da natureza, esse trabalho é uma pesquisa aplicada. Quanto à abordagem do problema, trata-se de uma pesquisa qualitativa. Referente aos objetivos, classifica-se como uma pesquisa exploratória. Considerando os procedimentos técnicos, a pesquisa é bibliográfica.

2.1. Etapa de Estudo Bibliográfico

A etapa de Estudo Bibliográfico representa a pesquisa exploratória de literatura relacionadas a inteligência artificial, geração de imagens, geração de conteúdo procedural e a

Figura 1. Fluxograma de desenvolvimento



Fonte: O próprio autor (2023)

implementação dessas tecnologias no contexto de geração de arte de personagens para jogos eletrônico. Os resultados adquiridos por essa etapa podem ser observados na seção 3, o referencial teórico.

O passo 1 é chamado Seleção de Trabalhos, é caracterizado pela analise de fontes bibliográficas, onde foram selecionados 50 artigos e livros relevantes ao escopo do projeto. Diversas bases de dados e parâmetros de buscas distintas foram utilizadas. Após a leitura do resumo e introdução de todos os trabalhos, cada um recebeu uma nota de 1 a 5, de acordo com seu grau de relevância aos objetivos deste projeto. Os trabalhos mais relevantes receberam uma leitura mais aprofundada, e se posteriormente após a analise o seu conteúdo se manteve significante para este projeto, foi selecionado para ser utilizado como fundamentação teórica.

O passo 2 é chamado Estudo das ferramentas, é caracterizado pela analise das ferramentas a serem utilizadas. Cabe a esse passo levantar quais ferramentas seriam necessárias para se adequar ao caso de uso deste projeto, levando em consideração os

objetivos gerais e específicos, tal como o material lido anteriormente.

2.2. Etapa de Implementação

A etapa de Implementação representa a pesquisa aplicada dos conhecimentos técnicos e teóricos adquiridos. Os resultados obtidos por essa etapa podem ser observados na seção 4, o desenvolvimento.

O passo 3 é chamado Analisar formas de combinar métodos, e é caracterizada pelo procedimento implícito e informal de testes e pesquisa sobre o funcionamento das ferramentas analisadas anteriormente. Cabe a esse passo realizar diversos testes de tentativa e erro a fim de verificar como atingir o caso de uso deste projeto utilizando as ferramentas selecionadas. Para adquirir maiores conhecimentos práticos e teóricos específicos, foi utilizado informalmente canais de comunicação digitais que permitiam sanar dúvidas e entrar em contato com integrantes de comunidades do Stable Diffussion e Unity.

O passo 4 é chamado Instalação de Ferramentas, presente na etapa Implementação, e é caracterizada pela configuração do ambiente, visto que as ferramentas utilizadas são pré-requisito para a realização deste trabalho. A realização das configurações deste ambiente foram realizadas em um sistema operacional Microsoft Windows e estes passos podem não ser possíveis de replicar em outros sistemas operacionais, como sistemas Linux ou MacOS.

O passo 5 é chamado Testes de Geração de Arte dos Personagens, presente na etapa Implementação, e é caracterizada pelos testes iniciais na geração de imagens. Cabe a esse passo analisar os principais obstáculos na geração inicial de um personagem e mitiga-los através de *prompts* e *negative prompts*, comparar os modelos do Stable Diffusion utilizados e demonstrar a integração com o Unity.

O passo 6 é chamado Testes de Geração de Personagens com posição, e é caracterizada pela o processo necessário para a geração do personagem na posição desejada, permitindo assim que seja possível utilizá-lo como ativo artístico. Cabe a esse passo demonstrar a limitação da utilização de somente *prompts*, demonstração da limitação do método img2img e a demonstração da eficácia dos métodos DWOpenPose e Depth.

O passo 7 é chamado Adaptando técnicas para outros casos de uso, e é caracterizada pela modificação da geração de personagem para outros cenários desejados durante o desenvolvimento. Cabe a esse passo demonstrar que a geração de personagens por inteligência artificial não está limitado a só um tipo de personagem ou estilo.

O passo 8 é chamado Modificações para tornar em um ativo artístico utilizável, e é caracterizada pela utilização de uma ferramenta de edição de imagens para realizar a modificação dos personagens gerados com o objetivo de tornar em ativo artístico utilizável no desenvolvimento de jogos. Cabe a esse passo realizar a correção de inconsistências e divisão dos membros do personagem.

O passo 9 é chamado Integrando novo personagem ao Unity, e é caracterizada pela utilização do personagem gerado em um cenário do Unity. Cabe a esse passo importar o ativo artístico editado, criar e configurar seu o esqueleto e pesos para permitir por fim modificar a posição do personagem e anima-lo.

3. Referencial teórico

3.1. Introdução a Inteligencia Artificial

Segundo Luger (2009) a definição de Inteligência Artificial (IA) refere-se à capacidade de uma entidade artificial exibir inteligência na resolução de problemas complexos, sendo este tipo de sistema comumente reconhecido como um dispositivo computacional ou máquina. Inteligência, por sua vez, pode ser caracterizada por entender, memorizar, reconhecer padrões e se adaptar a mudanças, que em algoritmos de IA pode ser considerada como sua habilidade de alcançar objetivos no mundo e resolver problemas complexos de uma forma similar a um humano (Kumar and Thakur, 2012).

Inteligência artificial e o aprendizado de máquina (AM) são termos frequentemente usados em conjunto, mas não são sinônimos. Segundo Ongsulee (2017), "Evoluindo do estudo do reconhecimento de padrões e da teoria da aprendizagem na inteligência artificial, a aprendizagem de máquina explora o estudo e construção de algoritmos que podem aprender e fazer previsões sobre dados". Em outras palavras, a IA pode ser caracterizada como um campo mais amplo que se preocupa com o desenvolvimento de algoritmos e sistemas capazes de realizar tarefas que, de outra forma, exigiriam inteligência humana para serem executadas. O aprendizado de máquina, por sua vez, é um conjunto de técnicas específicas dentro da IA que se concentra na capacidade de os sistemas aprenderem a partir de dados.

A área de geração de imagens com aprendizado de máquina apresenta grande potencial na área de IA, visando gerar imagens de diferentes estilos a partir de dados de entrada variados. Segundo Goodfellow et al. (2020), "as técnicas de geração de imagens usando redes neurais profundas têm o potencial de produzir imagens realistas de alta qualidade de objetos, cenas e pessoas que nunca foram vistos antes". Isso é possível através do treinamento de redes neurais profundas em grandes conjuntos de dados de imagens reais, permitindo que a rede aprenda a extrair as principais características das imagens e a gerar novas imagens que se assemelhem a elas. Essa técnica tem diversas aplicações, como em jogos, filmes e animações, além de ter um grande potencial em áreas como design gráfico e publicidade.

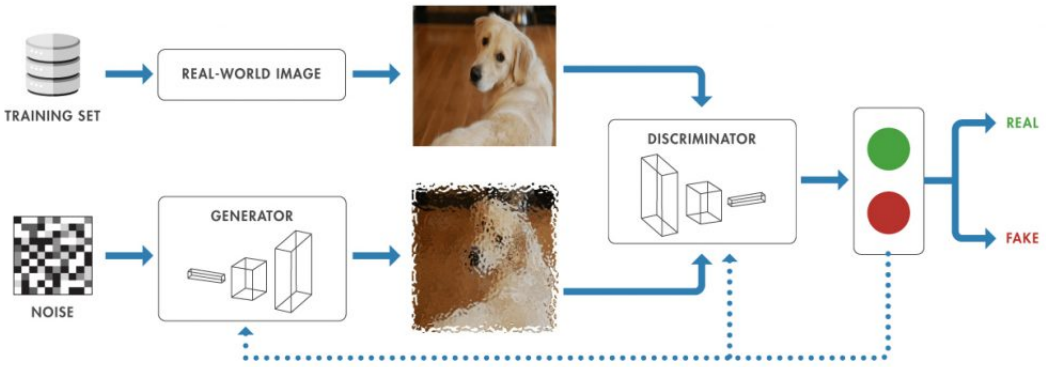
Duas técnicas de IA para a geração de imagens através de aprendizado de máquina que podem ser citadas como mais promissoras e estado da arte nos tempos atuais são Redes Adversárias Generativas (GANs) e Modelos de Difusão. Até o ano de 2021, os resultados adquiridos pelas técnicas GANs podiam ser considerados os mais avançados para a produção de imagens por meio de técnicas de aprendizado de máquina (Dhariwal and Nichol, 2021). Recentemente, trabalhos utilizando Modelos de Difusão como DALL-E 2 (Ramesh et al., 2022) demostraram melhores resultados em questão de qualidade e variedade na em comparação com redes GAN, apesar de em muitos casos necessitarem de um custo computacional mais elevado (Dhariwal and Nichol, 2021).

3.2. Utilização de GAN para geração de imagens em jogos

As GANs são uma classe de algoritmos de Inteligência Artificial que são usados para gerar dados sintéticos, como imagens, áudio ou texto, que se assemelham a dados reais. Segundo Creswell et al. (2018), GANs consistem em dois componentes principais: um gerador e um discriminador. O gerador é responsável por criar dados sintéticos a partir de um espaço de amostras latentes, geralmente seguindo uma distribuição de probabilidade

específica. O discriminador, por sua vez, recebe tanto dados reais quanto dados sintéticos gerados pelo gerador e tenta distinguir entre eles. O objetivo do gerador é produzir dados que possam "enganar" o discriminador, enquanto o objetivo do discriminador é identificar corretamente se os dados são reais ou sintéticos (Figura 2). Esse processo de treinamento iterativo permite que o gerador aprenda a gerar dados cada vez mais realistas .

Figura 2. Processo de geração de imagens utilizando GAN



Fonte: Molera (2021)

Com o intuito de otimizar o processo de geração de arte em jogos, alguns trabalhos experimentaram utilizar aprendizado de máquina para criar ou modificar imagens que possam ser utilizadas como ativos artísticos no desenvolvimento de jogos, principalmente através de implementações de algoritmos GANs. Apesar disso, a literatura de aplicações nesta área específica é escassa. Em seu artigo Serpa and Rodrigues (2019) atribuem a falta de trabalhos publicados sobre a geração automática de *sprites* e personagens em jogos devido à alta importância e detalhes necessários para esses ativos, já que uma geração mal feita pode ser facilmente percebida pelos usuários e afetar negativamente a qualidade do jogo.

Horsley and Perez-Liebana (2017) utilizaram Redes Adversárias Gerativas Convolucionais Profunda (DCGAN), com uma quantidade limitada de dados de treinamento, provando assim que era possível gerar *sprites* de personagens humanos, rostos e criaturas através de Inteligência Artificial. Hong et al. (2019) conseguiram gerar arte de personagens com animações através de Redes Adversárias Generativas Multi-discriminativas (MDGAN) que possuí dois discriminadores distintos, um representando a cor do personagem e o outro o esqueleto. Serpa and Rodrigues (2019), utilizando a arquitetura Pix2Pix, experimentaram com a edição de cores e sombreamento de personagens já existentes durante a linha de produção real de um jogo eletrônico em desenvolvimento.

3.3. Avanços dos Modelos de Difusão

Os Modelos de Difusão, também conhecidos como Modelos de Difusão Estocástico, são uma classe de algoritmos utilizados em Inteligência Artificial e processamento de imagens. O processo de Modelos de Difusão são caracterizados por diversas iterações de criar ruído, geralmente uma distribuição Gaussiana, por cima de um conjunto de dados e logo após tentar reverter os dados ao seu estado original (Sohl-Dickstein et al., 2015). O treinamento desta classe de algoritmos é realizada através da diminuição gradual entre a

diferença entre os valores previsos de ruído e os valores reais resultantes do algoritmo (Figura 3). Após diversas iterações, este processo é capaz de gerar imagens de alta qualidade, fidelidade e variedade (Dhariwal and Nichol, 2021).

Figura 3. Processo de geração de imagens utilizando Modelos de Difusão

Fonte: Arash Vahdat (2022)

A geração de imagens utilizando Modelos de Difusão foram um dos grandes avanços da Inteligência Artificial e aprendizado de máquina durante o período de 2021 e 2023. Segundo Dhariwal and Nichol (2021), as novas técnicas estado-da-arte para a geração de imagens, incluindo modelos de difusão, apresentam resultados melhores que Redes Adversárias Generativas em questão de qualidade e variedade dos dados gerados.

O lançamento do DALL-E 2, gerador de imagens com a rede neural CLIP (Ramesh et al., 2022), trouxe um novo precedente com geração de arte de alta qualidade e fidelidade através de entradas de texto do usuário. O Imagen, através do processamento de imagens utilizando codificadores T5-XXL, demonstrou resultados consideravelmente superiores em relação a imagens geradas através de CLIP (Saharia et al., 2022). Por fim, o Stable Diffusion democratizou o acesso a geração de imagens com Inteligência Artificial, disponibilizando um código fonte aberto e permitindo o treinamento dos Modelos de Difusão em ambientes com recursos computacionais limitados através da aplicação de codificadores automáticos pré-treinados (Rombach et al., 2022).

Sob conhecimento dos autores deste trabalho durante a data de sua escrita, não foram encontrados exemplos significativos de implementações aplicadas que utilizam modelos de difusão para a geração de arte personagens em jogos eletrônicos. Consequentemente, isso representa uma possibilidade tecnológica e cientifica para a utilização dos Modelos de Difusão na área de geração de arte para jogos eletrônicos.

3.4. Engenharia de *Prompt*

Segundo Liu and Chilton (2022), o processo de geração de imagens utilizando *prompts* de texto podem acabar resultando em uma metodologia baseada em ”força-bruta”, através de tentativa e erro. Com isso, surge a engenharia de *prompt* (ou *prompt engineering* em inglês), uma técnica utilizada no campo da inteligência artificial utilizada em em sistemas de geração de linguagem natural com o intuito da criação cuidadosa e a formulação de *prompts* e outros argumentos para direcionar o modelo a gerar resultados de acordo com as necessidades do usuário (White et al., 2023).

O processo de refinamento da engenharia de *prompt* pode ser caracterizado como iterativo (Liu and Chilton, 2022) (White et al., 2023), pois após a utilização de um *prompt*

inicial é necessário avaliar os seus resultados e posteriormente adapta-lo ainda mais para o caso de uso especifico e modelo utilizado. O método de definir qual *prompt* é mais adequado pro caso de uso desejado depende da avaliação de uma ou mais pessoas, e pode ser realizada de forma subjetiva ou levando principalmente em consideração alguma mudança especifica nos resultados obtidos (por exemplo presença ou exclusão de elemento da imagem, estilo diferente, modificação na forma dos elementos da imagem).

3.5. ControlNet

Segundo Zhang et al. (2023), o ControlNet é uma arquitetura fim-a-fim para modelos de difusão que aprende controles condicionais para grandes modelos de difusão *text-to-image* pré-treinados. Essa arquitetura bloqueia a modificações nos parâmetros do modelo principal e ao mesmo tempo realiza copia das suas camadas de codificadores. Utilizando na prática no contexto do Stable Diffusion, o ControlNet permite com que o modelo principal seja utilizado como a base mais importante para geração, enquanto um ou mais sub-modelos configuráveis sejam utilizados para direcionar e refinar a geração para o resultado desejado.

O ControlNet possuí diversos sub-modelos que podem direcionar os resultados adquiridos que conseguem ser utilizados simultaneamente em paralelo Zhang et al. (2023). Dois sub-modelos que podem ser citados são Depth Map, que utiliza um mapa de profundidade gerado através de uma imagem de entrada (Ranftl et al., 2020), e OpenPose, que utiliza de um esqueleto de posição humanada gerado através de figuras humanoides contidas na imagem de entrada (Cao et al., 2017).

3.5.1. Depth Map

Depth Map (ou em português mapa de profundidade) é uma representação bidimensional da informação de profundidade de uma cena, que através de uma imagem de entrada, atribui um valor de profundidade a cada pixel (Ranftl et al., 2020). O mapa de profundidade é utilizado para indicar a distância que os objetos estão da câmera, sendo utilizado para diversos casos de uso, como por exemplo realidade virtual, veículos autônomos, robótica, reconstrução 3D, reconhecimento de gestos e imagenologia médica.

Ranftl et al. (2020) apresentaram uma nova técnica estado-da-arte para a estimação de profundidade. Dois dos principais avanços da técnica são a utilização de aprendizado auto-supervisionado, que permite com que o treinamento utilize somente a imagem e não necessite de anotações externas de profundidade, e a utilização de estrições geométricas e fotométricas, que permite aprender a profundidade não só a partir de imagens únicas como também a partir de vídeos.

3.5.2. Openpose e DWOpenpose

O OpenPose é uma biblioteca de código aberto (ou *open source* em inglês) projetada para detecção de pose humana em tempo real a partir de imagens e vídeos. A detecção de pose refere-se à capacidade de identificar e rastrear as posições de diferentes partes do corpo humano, como cabeça, ombros, cotovelos, mãos, quadris, joelhos e pés, em uma imagem ou vídeo (Cao et al., 2017). O OpenPose utiliza técnicas de visão computacional

e aprendizado de máquina para realizar essa tarefa. Ele emprega uma rede neural convolucional (CNN) para analisar a imagem e identificar os pontos-chave que compõem a pose de uma pessoa. A saída do OpenPose é um conjunto de coordenadas 2D que representam a posição de cada articulação detectada.

O DWOpenpose por sua vez, é um novo modelo de estimação de posição humana desenvolvido para aprimorar a geração de imagens no ControlNet (Yang et al., 2023). Ele substitui o modelo de estimação de posição anterior, Openpose, com um maior grau de performance, maior qualidade de informação no esqueleto final e maior eficiência na geração de imagens no ControlNet. Segundo Yang et al. (2023), o DWOpenpose possuí diversas aplicações práticas, como por exemplo interação humano-computador, analise de preparamento físico, vigilância, animação e controle de personagem virtual.

3.6. Animação de personagens 2D

Em uma parte considerável dos jogos eletrônicos 2D, os personagens possuem animações para representar seu movimento e ações que exercem sobre o mundo virtual. Como a imagem em duas dimensões que representa um personagem é um ativo artístico que por si só não possuí movimento, é necessário utilizar alguma das diversas técnicas disponíveis para permitir que tal ativo seja animado. Existe duas técnicas mais comumente utilizadas no mercado, sendo essas animações através de *sprites* e animações esqueletais.

A animação por *sprites* envolve a criação de uma sequência de imagens (*sprites*) que representam os diferentes quadros de uma animação (Figura 4). Cada *sprite* contém uma pose específica do personagem, sendo o conjunto completo dos *sprites* do personagem chamado de *sprite set* (Hong et al., 2019) ou *sprite sheet*. O processo de animação através de *sprites* pode ser dividido nos seguintes passos:

- Criação de *Sprites*: Desenha-se cada quadro da animação em um software de edição de imagem. Cada imagem representa uma pose do personagem em um determinado momento;
- Organização: Os *sprites* são organizados em uma sequência, onde cada quadro é exibido em uma ordem específica para criar a ilusão de movimento.;
- Definição de Taxa de Quadros (*Frame Rate*): A taxa de quadros determina a velocidade da animação. Quanto mais quadros por segundo, mais suave será a animação.
- Implementação no Software: No ambiente de desenvolvimento (como um jogo ou software de animação), os *sprites* são carregados e exibidos em sequência, criando a ilusão de movimento.

A animação esquelética (ou *skeletal animation*, em inglês) envolve a utilização de uma estrutura de ossos (esqueleto) para animar personagens (Figura 5). Cada osso é associado a uma parte do corpo (como braços, pernas, cabeça) e as poses são criadas deformando o personagem em torno desses ossos (Mukundan and Mukundan, 2012). O processo de animação esqueletal pode ser dividido nos seguintes passos:

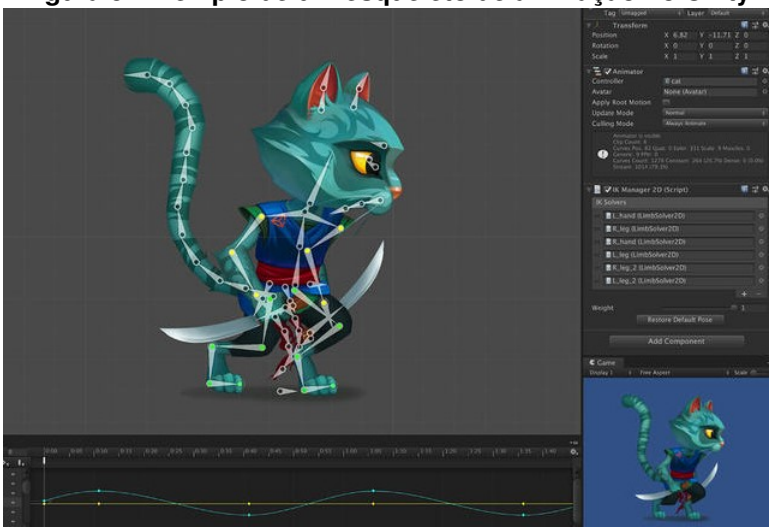
- Criação do Esqueleto (*rigging*): Cria-se uma estrutura de ossos que reflete a anatomia do personagem. Cada osso é associado a uma parte específica, como cabeça, braços, pernas, etc;
- Associação dos *Sprites*: Os *sprites* do personagem são associados aos ossos correspondentes. Isso é feito de forma que, quando um osso é movido, o *sprite* ligado a ele também se mova.

Figura 4. Exemplo de um *sprite sheet* de um personagem qualquer

Fonte: Ng (2017)

- Animação de *Keyframes*: Define-se os principais momentos da animação (*keyframes*), ajustando a posição e rotação dos ossos para criar poses diferentes;
- Interpolação: O software de animação calcula automaticamente os quadros intermediários, suavizando a transição entre poses;
- Implementação no Software: No ambiente de desenvolvimento, o esqueleto animado é carregado e os movimentos são reproduzidos em tempo real.

Figura 5. Exemplo de um esqueleto de animação no Unity



Fonte: Unity (2023)

Ambas as técnicas têm suas vantagens e são utilizadas em diferentes contextos. A escolha entre elas depende das necessidades específicas do projeto, como complexidade

da animação, eficiência de recursos e estilo visual desejado. Em alguns casos, é possível até combinar as duas técnicas para obter o melhor de cada.

Por padrão em técnicas de animação esquelética, o personagem é disposto em uma posição padrão de animação. Comumente em ambas animação de personagens 2D quanto 3D, é geralmente utilizado a posições de T-Pose, onde o personagem tem disposto seus braços abertos de forma perpendicular ao corpo, ou A-Pose, onde o personagem tem disposto seus braços abertos de forma diagonal ao corpo (Zwettler et al., 2021).

3.7. Geração de Conteúdo Procedural (PCG)

Um princípio de design utilizado para tornar a produção de conteúdo mais eficiente no desenvolvimento de jogos é a Geração de Conteúdo Procedural (PCG), que pode ser caracterizada pela criação de diferentes tipos de elementos de um jogo eletrônicos automaticamente através de algoritmos (Togelius et al., 2011). A presença de PCG em jogos pode ser observada desde 1980 até o ano de escrita deste artigo, estando presente em jogos eletrônicos com alto impacto para a industria, como por exemplo Rogue (Epyx, 1985), Civilization (MicroProse, 1991), Spore (Maxis, 2008), Minecraft (Mojang, 2011) e Diablo (Blizzard Entertainment, 1996).

Muitas implementações de Geração de Conteúdo Procedural utilizam algoritmos de IA e *machine learning*. Volz et al. (2018) elaboraram um gerador de novas fases utilizando Redes Adversárias Gerativas Convolucionais Profundas (DCGAN), que eram treinadas por segmentos de fases do jogo Super Mario Bros e posteriormente eram avaliadas por um agente simulado no algoritmo de busca A* de acordo com parâmetros customizados. Hollingsworth and Schrum (2019) desenvolveram um jogo chamado "Infinity Art Galery", que através de Redes Produtoras de Padrões de Composição (CPPNs), consegue gerar novas pinturas e esculturas proceduralmente em tempo de execução baseado em certas interações do jogador com as próprias obras de arte ou itens contidos no mundo. No artigo de Park et al. (2023), foi demonstrado uma vila virtual com 25 agentes, utilizando instancias do ChatGPT, que possuíam personalidades únicas e eram capazes de simular comportamentos sociais humanos como conversar entre si, seguir uma rotina diária, ter opiniões distintas e até mesmo planejar eventos complexos, como por exemplo uma festa, organicamente.

3.8. Ferramentas utilizadas

O desenvolvimento de jogos eletrônicos requer ferramentas adequadas para criar jogos de qualidade e de forma eficiente. Os desenvolvedores podem escolher entre várias opções de ferramentas e linguagens de programação para criar jogos que funcionem em diferentes plataformas. Dentre as tecnologias relevantes para a implementação prática dos objetivos estabelecidos neste trabalho, estão a linguagem Python e as ferramentas Unity, Automatic 1111, Magic Poser e GIMP.

A linguagem de programação Python é uma linguagem de alto nível, interpretada e de código aberto. Em maio de 2023, Python foi classificado como a linguagem de programação mais popular do mundo (Tiobe, 2023), com uma comunidade ativa e diversificada de desenvolvedores. Uma das principais razões para sua popularidade é a sua sintaxe simples e clara, que torna a linguagem fácil de aprender e usar, mesmo para iniciantes. Além disso, Python é conhecido por sua versatilidade e flexibilidade, sendo

aplicado em diversos campos, desde desenvolvimento web até ciência de dados e inteligência artificial.

Unity é uma das ferramentas de desenvolvimento de jogos mais populares e amplamente utilizadas na indústria de jogos. De acordo com Young (2021), Unity é utilizado no desenvolvimento de 50% de todos os jogos *mobiles*, o que representa aproximadamente 35 milhões de dólares, somente neste mercado. A ferramenta permite com que desenvolvedores utilizem a linguagem C#, em conjunto com uma ampla variedade de ferramentas e bibliotecas de terceiros para estender suas capacidades. Em conjunto com o Unity, será utilizado a biblioteca Stable Diffusion Unity Integration, que é capaz de invocar a geração de uma imagem pela API do Automatic1111.

O Automatic1111, ou também conhecido como Stable Diffusion Web UI, é uma GUI web que utiliza o Stable Diffusion para fazer gerações de imagens, como por exemplo operações de texto para imagem ou imagem para imagem. Essa ferramenta é de código aberto, flexível e modular permitindo diferentes tipos de argumentos e extensões. Para este trabalho será utilizado também sua Web API, em que permite que possa chamar muitas das funcionalidades do Automatic1111 por outra aplicação através de chamadas HTTP enquanto o aplicativo estiver rodando. Neste trabalho, o Automatic1111 rodará em um dispositivo no caminho *localhost*, e gerações podem ser realizadas diretamente na UI, acessada no seu endereço local, ou encaminhado chamadas HTTP para sua API.

Magicposer é uma ferramenta *mobile* grátis com adicionais pagos, que permite posicionar os membros de um personagem 3D de exemplo de acordo com a posição buscada. Esse personagem 3D de exemplo e sua posição tem o intuito de ser usada como referência anatômica para artistas, e neste trabalho será utilizada como uma das entradas para geração de novos personagens 2D por IA. Apesar de ser um aplicativo mobile, seu site oficial também oferece sua versão gratuita para ser acessada pelo próprio navegador de qualquer computador.

O GIMP, sigla para GNU *Image Manipulation Program*, é uma avançada ferramenta de edição de imagens de código aberto. Com suporte para diversos formatos de arquivo, como JPEG e PNG, é amplamente utilizado por designers gráficos e fotógrafos. Sua interface personalizável e ampla variedade de extensões e filtros permitem a realização eficiente de tarefas complexas, como retoques e composições. O GIMP é um exemplo notável da eficácia da comunidade de código aberto em aprimorar e expandir suas funcionalidades.

4. Desenvolvimento

Durante a etapa de Desenvolvimento, serão apresentado os passos para a criação e a utilização de um personagem gerado por Inteligência Artificial como ativo artístico animável no desenvolvimento de jogos. Nesta seção é abordado desde a instalação da ferramenta até a o momento em que o ativo artístico do personagem está em um cenário do Unity, com seu esqueleto configurado e pronto para ser animado. Cabe também a esta etapa fazer ajustes nos argumentos de geração do Automatic1111 para gerar um ativo artístico que necessita do mínimo de intervenção manual humana possível, e além disso, apontar obstáculos tecnológicos onde a automatização da geração de arte não foi possível de ser realizada.

Para atender o caso de uso deste projeto e posteriormente permitir a animação

deste personagem, em um primeiro momento é necessário com que o Automatic1111 gere arte para personagens em 2D que estejam na posição de frente, como T-Pose ou A-Pose, e uma posição de lado. Esta posição foi inicialmente escolhida pois é um facilitante para que os membros dos personagens sejam divididos e posteriormente animados separadamente, através de animação esqueletal. Para que o ativo artístico do personagem possa ser utilizado em diversas situações, faz-se necessário gerar o personagem disponibilizado em duas posições: de frente e lado. Nas imagens geradas, a posição dos membros do personagem deve ser consistente nas imagens geradas. É também ideal que o personagem não possua objetos na mão e seu estilo de arte se mantenha similar entre gerações.

Todos os testes foram realizados em um notebook Acer Nitro 5, que possuí uma placa de vídeo NVIDIA GTX 1050ti. A baixa disponibilidade de recursos computacionais deste dispositivo não afeta a qualidade ou fidelidade da imagem gerada, mas tem um impacto negativo direto na velocidade de execução deste processo.

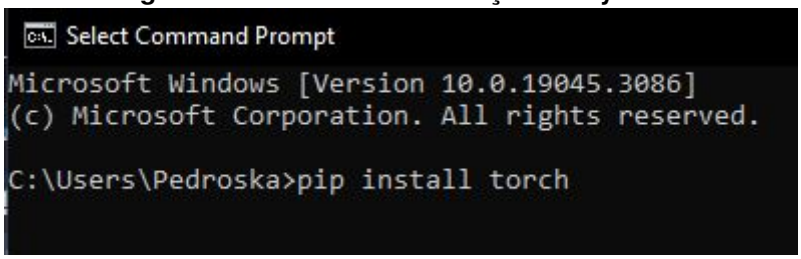
4.1. Instalação de Ferramentas

Foi necessário realizar os preparativos iniciais do ambiente para que este projeto fosse desenvolvido. Com isso, foram efetuados os os passos para a instalação do Automatic1111 e utilitários, Unity e sua biblioteca Stable Diffusion Unity Integration. É importante levar em consideração que os passos a seguir foram realizados em um computador com o sistema operacional Windows 10 e podem ser diferentes em outros tipos de sistema operacional como por exemplo MacOs ou Linux.

4.1.1. Instalação de Automatic1111 e seus utilitários

Para instalar o Automatic1111, é necessário ser instalada no ambiente o Python, sendo neste projeto utilizado a versão 3.10.6 e com seus binários adquiridos em seu site oficial. Também é necessário a biblioteca PyTorch do Python, que pode ser instalada através da utilização do comando “pip install torch” (Figura 6). Após realizar este passo, é necessário adquirir o Automatic1111 em seu repositório do Github (Figura 7), seja através de clonar o repositório ou baixar o repositório como um arquivo ZIP e posteriormente descompactar. Para este projeto, utilizaremos o Automatic1111 versão 1.3.2 e PyTorch versão 2.0.1.

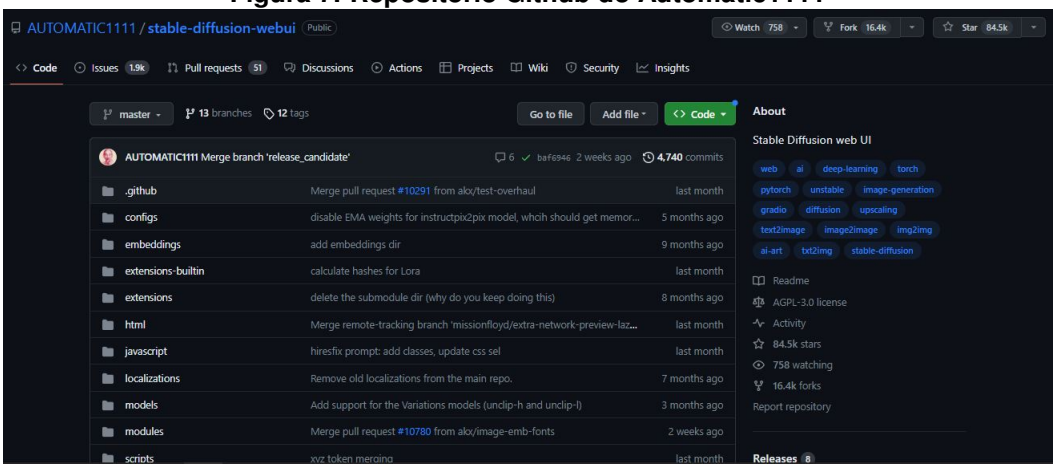
Figura 6. Comando de instalação do PyTorch



Fonte: O próprio autor (2023)

Com o Automatic1111 descompactado, é necessário realizar seus ajustes finais. É necessário colocar o argumento “–api” na linha “set COMMANDLINE_ARGS=”

Figura 7. Repositório Github do Automatic1111



Fonte: O próprio autor (2023)

presente no arquivo “webui-user.bat” (Figura 8) presente na pasta raiz da ferramenta, habilitando assim chamadas HTTP para o Unity e potencialmente outros aplicativos. Por fim, é preciso adquirir um modelo pré-treinado do Stable Diffusion e colocar na pasta “Models” da ferramenta. Existem diversos modelos na disponíveis na Internet e também é possível realizar o treinamento de seu próprio modelo customizado. Os modelos utilizados pelo processador do Automatic1111 geralmente são no formato “.safetensors” ou “.ckpt” e para serem instalados devem ser copiados para o caminho “`\stable-diffusion-webui\models\Stable-diffusion`"”. Por razões de demonstração, será utilizado quatro modelos para a geração:

- v1-5-pruned-emaonly: modelo oficial do Stable Diffussion, simples e com mínimo de treinado, destinado a geração básica de imagens;
- dreamshaper_8: é um dos modelos mais utilizados para gerar imagens expressivas de alta qualidade, também possuindo resultados notáveis para geração de arte no gênero de fantasia medieval.
- aZovyaRPGArtistTools_v3: modelo treinado levando em consideração principalmente a utilização para casos de uso em que necessita arte com estilo de RPG (*Role Playing Game*) ou fantasia medieval em geral.
- AnythingV5Ink_v5PrtRE: modelo treinado para gerar arte no estilo de ilustração japonesa, como anime e manga, sendo entre os quatro modelos o que resultará em imagens de estilo mais distintas.

Figura 8. Formato do arquivo webui-user.bat

webui-user.bat

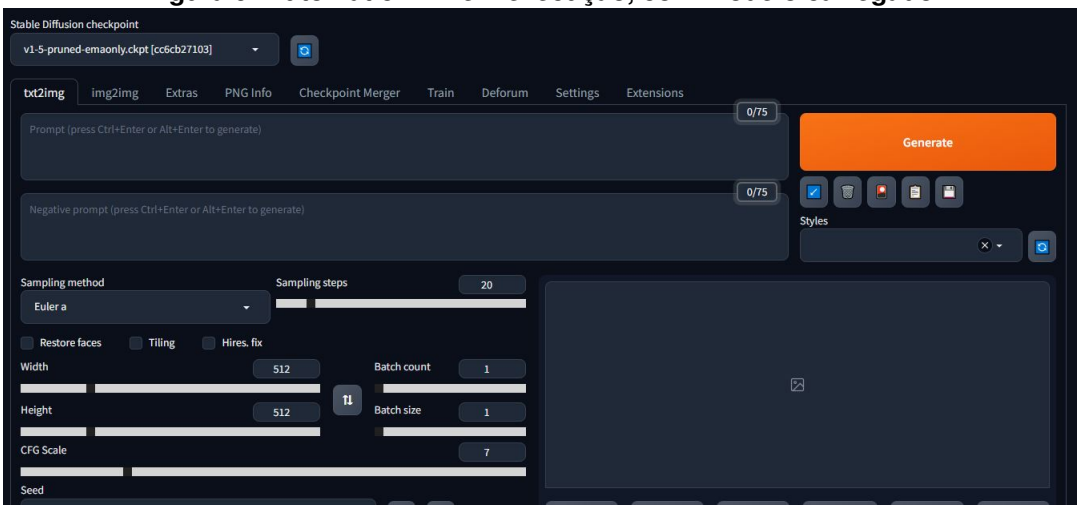
```
@echo off

set PYTHON="C:\Users\Pedroska\AppData\Local\Programs\Python\Python310\python.exe"
set GIT=
set VENV_DIR=
set COMMANDLINE_ARGS= --api
git pull
call webui.bat
```

Fonte: O próprio autor (2023)

Para executar o Automatic1111, é necessário abrir o arquivo “webui-user.bat” presente em sua pasta, que resultará na execução da ferramenta (Figura 9) em uma das portas do dispositivo local, sendo esta porta por padrão a 7860. Durante a primeira execução, é esperado que o Automatic1111 baixe arquivos e bibliotecas de dependência, o que pode acabar tomando alguns minutos dependendo dos recursos computacionais do dispositivo e velocidade da internet.

Figura 9. Automatic1111 em execução, com modelo carregado



Fonte: O próprio autor (2023)

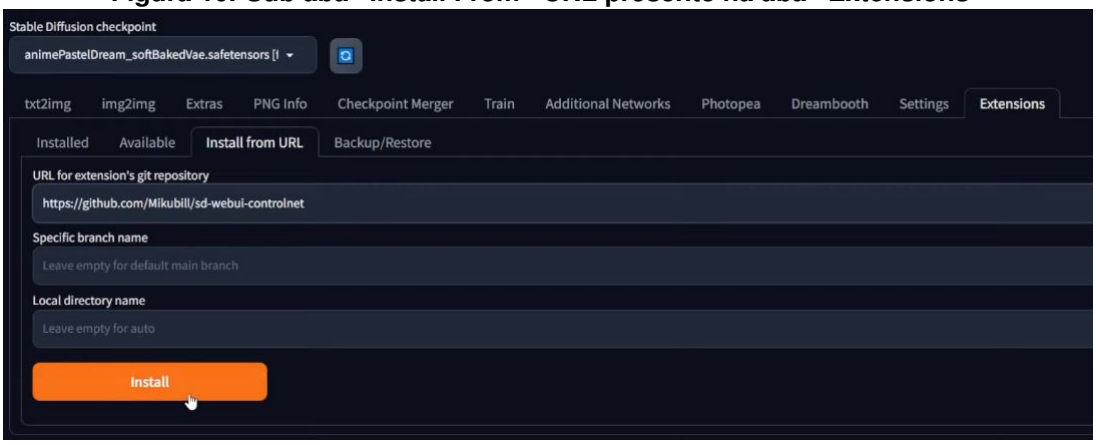
Após as configurações iniciais do Automatic1111 estiverem prontas, é necessário configurar as extensões Controlnet e DWOpenpose. Com a ferramenta Automatic1111 aberta, é preciso ir na aba *“Extensions”* e posteriormente na sua sub *“Install from URL”* (Figura 10), digitando o repositório da extensão desejada e posteriormente clicando no botão “Install”. Este processo deve ser realizado utilizando como argumento o link “https://github.com/Mikubill/sd-webui-controlnet” para instalar a extensão Controlnet e o link “https://github.com/IDEA-Research/DWPose” para instalar a extensão DWOpenpose. Por fim, é necessário abrir a sub-aba *“Installed”* e apertar o botão *“Apply and Restart UI”* (Figura 11) para aplicar as mudanças.

O último passo é a instalação dos modelos que serão utilizados pelo Controlnet, pois os seus preprocessadores necessitam de um modelo definido ao realizar os métodos de Depth e OpenPose. Os modelos utilizados pelos preprocessadores do ControlNnet geralmente são no formato “.safetensors” ou “.pth” e para ser instalados devem ser copiados para o caminho “`\stable-diffusion-webui\extensions\sd-webui-controlnet\models`”. Para o caso de uso deste projeto será utilizado os modelos “control_v11f1p_sd15_depth.pth” e “control_v11p_sd15_openpose.pth”. É importante constar que os modelos utilizados pelos preprocessadores do Controlnet não são iguais aos modelos utilizados pelo processador do Automatic1111.

4.1.2. Instalação de Unity e biblioteca de integração

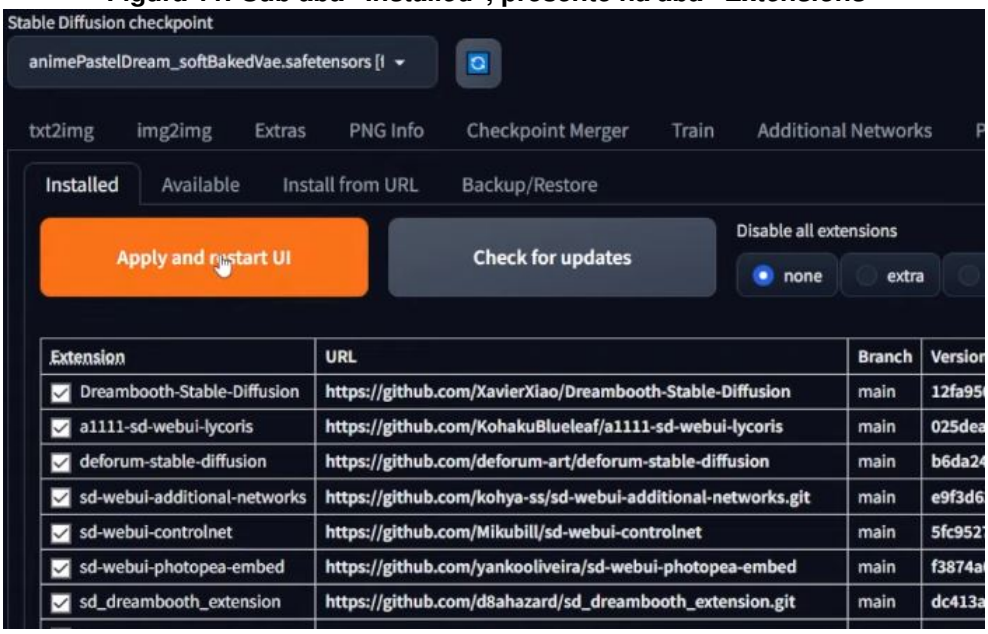
A ferramenta Unity pode ser instalada através de seus binários (Figura 12) disponibilizados pela Unity Technologies em seu site oficial, sendo recomenda a utilização da sua

Figura 10. Sub aba *"Install From"* URL presente na aba *"Extensions"*



Fonte: O próprio autor (2023)

Figura 11. Sub aba *"Installed"*, presente na aba *"Extensions"*



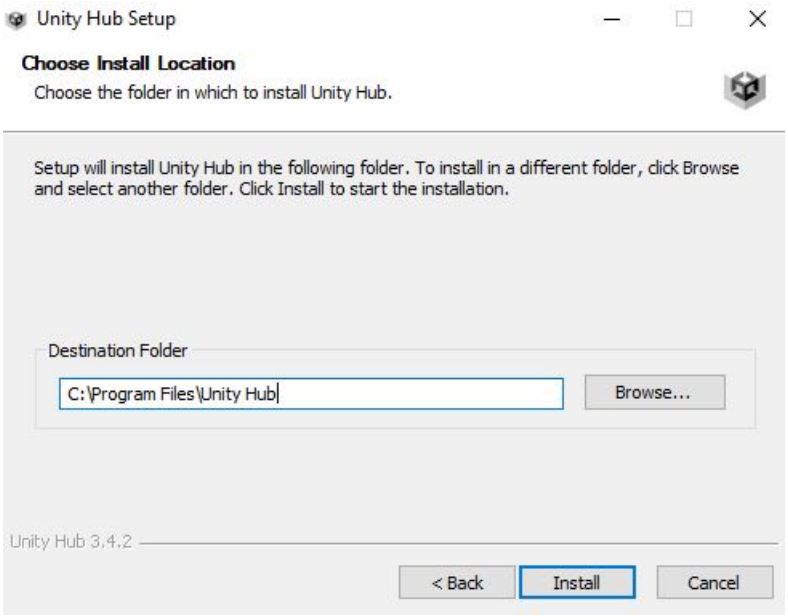
Fonte: O próprio autor (2023)

última versão. Durante o desenvolvimento deste trabalho, será utilizado o Unity versão "2022.3.0f1".

Para instalar a biblioteca Stable Diffusion Unity Integration, é necessário adquirir em seu repositório do Github, seja através de clonar o repositório ou baixar o repositório como um arquivo no formato ZIP e descompactar. É possível utilizar a biblioteca através do projeto Unity de demonstração (Figura 13) presente em sua pasta ou criando um projeto novo e posteriormente importando o arquivo "table-diffusion-unity-integration.unitypackage".

Por fim, com um projeto do Unity que possua a biblioteca Stable Diffusion Unity Integration aberto, é preciso abrir a entidade StableDiffusionConfiguration no menu *"Hierarchy"* e clicar no botão *"List Models"* (Figura 14). A biblioteca e consequentemente

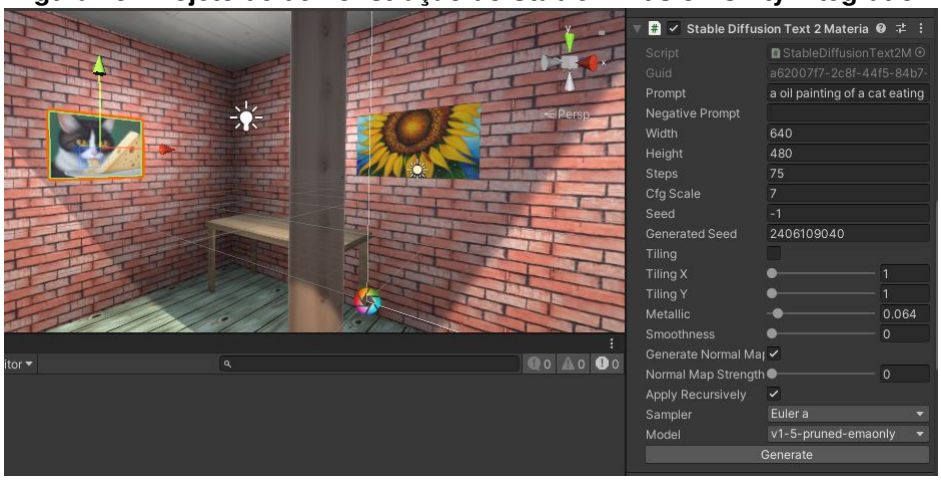
Figura 12. Instalador do Unity



Fonte: O próprio autor (2023)

sua integração com a API do Automatic1111 só estará funcionando corretamente caso os modelos instalados estejam listados ali nesta entidade. Este último passo é necessário ser realizado toda vez em que o Unity for iniciado.

Figura 13. Projeto de demonstração do Stable Diffusion Unity Integration

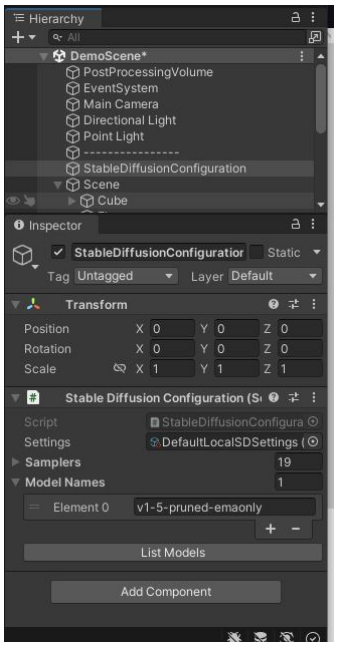


Fonte: O próprio autor (2023)

4.2. Testes de Geração de Arte dos Personagens

O objetivo deste passo é analisar os principais obstáculos na geração inicial de um personagem para o caso de uso deste trabalho e tentar mitiga-los através de engenharia de *prompt* do *prompt* principal e do *negative prompt*. Cabe também a este passo comparar os quatro modelos selecionados, a fim de que posteriormente no andamento deste trabalho

Figura 14. Barra lateral contendo o Botão de de *"List Models"*



Fonte: O próprio autor (2023)

utilize somente o modelo que possua os melhores resultados, selecionado de forma subjetiva a critério dos autores de acordo com a qualidade dos resultados. Por fim, é posto em prática um exemplo da integração da geração do Automatic1111 com o Unity.

Os principais testes realizados neste passo foram a geração através dos métodos txt2img, txt2img com *negative prompts* e a integração do Automatic1111 com o Unity através da biblioteca Stable Diffusion Unity Integration. Para os testes envolvendo a biblioteca de integração, foi utilizado o projeto de demonstração oficial da mesma.

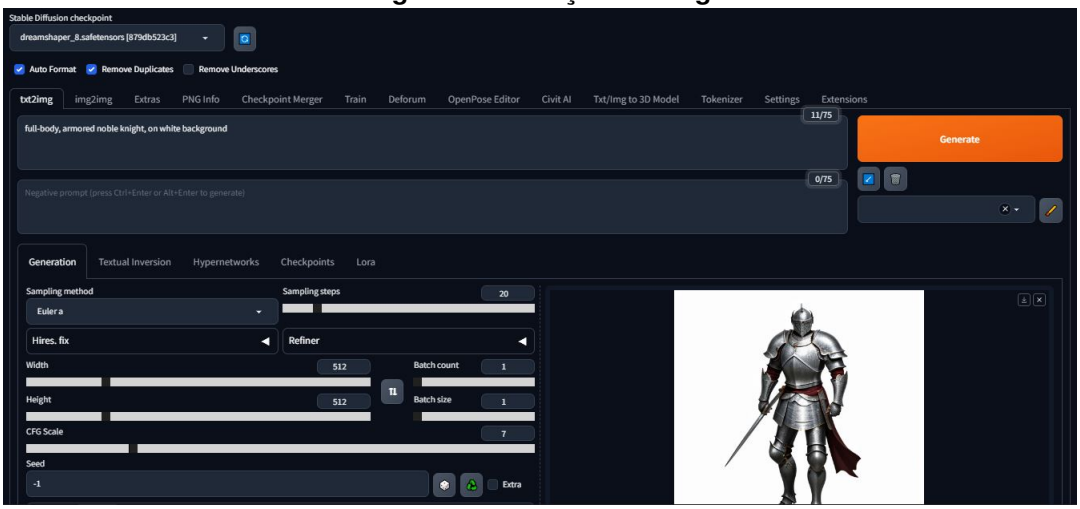
Todas as imagens de personagens de exemplo geradas nessa etapa representam exclusivamente um cavaleiro medieval vestido com armadura. Não foi gerado outros tipos de personagens para evitar que ocorra variações irrelevantes para o escopo dessa seção. Como não houve modificações nas configurações de resolução, gerações resultaram em imagens de saída com resolução padrão de 512x512 *pixels*.

4.2.1. Testes da Geração txt2img

O primeiro teste realizado foi a geração de um personagem diretamente no Automatic1111 através do método txt2img (Figura 15), que é responsável por gerar uma imagem através de um *prompt* de texto. O *prompt* de texto é um dos argumentos mais importantes para a geração da imagem no Stable Diffusion, representando a frase em que o modelo de difusão fará a interpretação do significado para criar a imagem.

Foi utilizado o *prompt* de texto *"full-body, armored noble knight, on white background"*, com o intuito de gerar um personagem de exemplo com o corpo inteiro a mostra sobre um fundo branco. Argumentos como *"full-body"* e *"on white background"* tem como objetivo definir caraterísticas da imagem que facilitam a utilização do personagem

Figura 15. Geração txt2img



Fonte: O próprio autor (2023)

gerado como um ativo artístico em um jogo eletrônico. Com exceção do *prompt* de texto, nenhum outro parâmetro foi modificado.

Foram realizadas várias gerações utilizando os modelos "v1-5-pruned-emaonly", "dreamshaper_8", "aZovyaRPGArtistTools_v3", "AnythingV5Ink_v5PrtR". Nas imagens resultantes (Figura 16), pode-se ser observado os seguintes obstáculos deste método para o caso de uso deste trabalho:

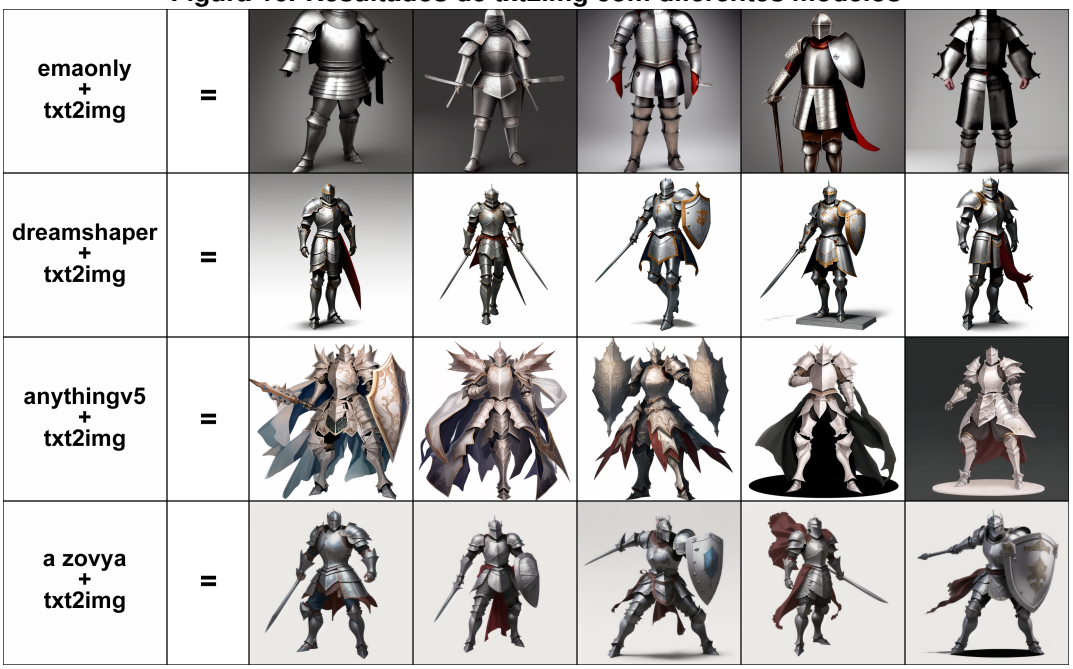
- A posição dos membros do personagem variam consideravelmente, o que tornaria inviável para uma futura animação dos mesmos;
- Personagens possuíam espadas, escudos, capas e outros aparatos que poderiam afetar a animação dos mesmos, principalmente nas gerações do modelo "AnythingV5Ink_v5PrtR";
- Várias imagens do modelo resultantes "v1-5-pruned-emaonly" possuíam membros do personagens fora da imagem, cortados e sem aparecer;
- Várias das gerações possuíam sombra ou uma base por baixo dos pés do personagens, sendo este um dificultador de uma futura animação dos mesmos.

Levando em consideração todos esses desafios, fica claro a necessidade de maior customização dos parâmetros na geração do imagens para se adequar melhor ao caso de uso deste projeto. Uma forma de potencialmente mitigar vários destes obstáculos seriam utilizar argumentos exclusores, que diminuíssem a ocorrência de resultados indesejados.

4.2.2. Testes da Geração txt2img com *Negative Prompts*

Para diminuir a ocorrência de elementos que possam atrapalhar na futura animação de personagens foi utilizado *negative prompts*. Os *negative prompts* funcionam de forma similar ao *prompt* principal porém de forma inversa, servindo como uma lista de argumentos a serem evitadas pelo algoritmo durante a geração da imagem. Essa técnica será utilizada para diminuir as ocorrências de objetos que possam afetar futuras animações, como armas e capas.

Figura 16. Resultados de txt2img com diferentes modelos



Fonte: O próprio autor (2023)

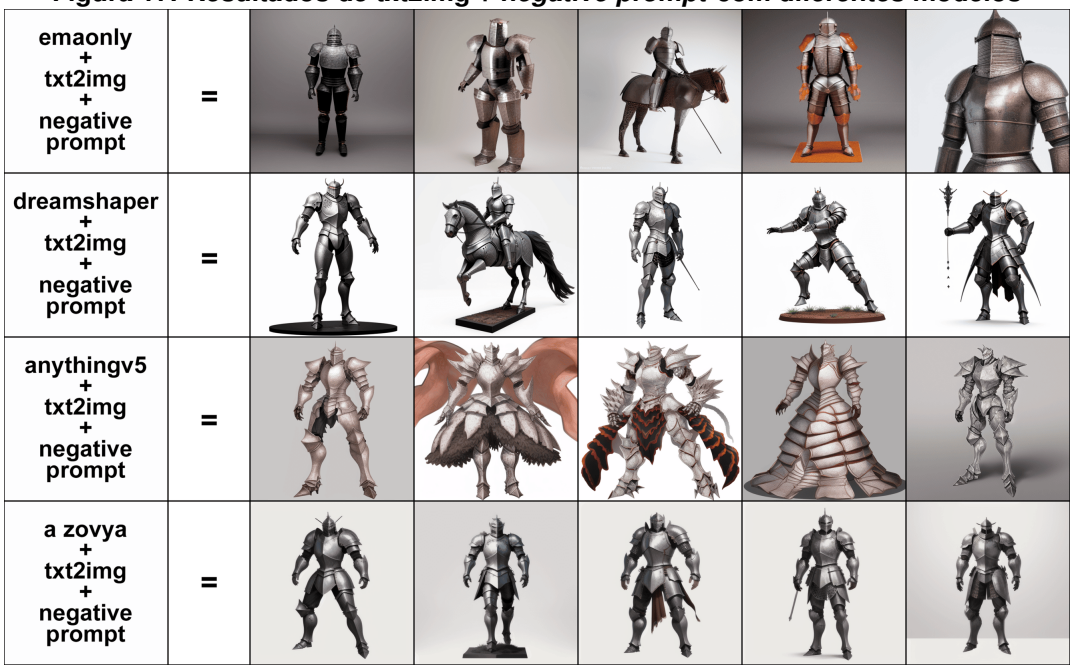
Foi utilizado o *prompt* de texto *"full-body, armored noble knight, on white background"*. Como *negative prompt*, foi utilizado *"(holding item), (object in hands), (holding weapon), shield, sword, wearing cape, long hair"*, onde argumentos com parenteses recebem mais peso durante a geração que o normal durante a geração. Não foi encontrado uma forma consistente de retirar a sombra e base da geração de alguns personagens.

Foram realizadas várias gerações utilizando os modelos "v1-5-pruned-emaonly", "dreamshaper_8", "aZovyaRPGArtistTools_v3" , "AnythingV5Ink_v5PrtR". Nas imagens resultantes (Figura 17), pode-se ser observado os seguintes obstáculos deste método para o caso de uso deste trabalho:

- Apesar de diminuir consideravelmente a presença de armas na mão dos personagens e escudos na mão do personagem, não impediu completamente a geração dos mesmos;
- Muitas das gerações do modelo "AnythingV5Ink_v5PrtR". possuíam diversos elementos complexos que impossibilitariam uma futura animação do personagem;
- Raramente, algumas das gerações dos modelos "aZovyaRPGArtistTools_v3" e "dreamshaper_8" resultaram em um cavalo de montaria, mostrando como a geração de arte por IA pode ser imprevisível e como é necessário refinar os argumentos iterativamente para cada resultado esperado.
- Como não foi utilizado nenhum *negative prompt* para retirar sombra e base de debaixo dos pés dos personagens, esse problema persiste em algumas ocorrências;
- Como não foi realizado nenhuma modificação que possa afetar posição inconsistente dos membros do personagem, esse problema persiste.

Os resultados deste método foram satisfatórios e houve um aumento considerável na consistência. Porém, os *negative prompts* utilizados não foram capazes de impedir

Figura 17. Resultados de txt2img + *negative prompt* com diferentes modelos



Fonte: O próprio autor (2023)

completamente que fosse gerado armas e outros itens na mão dos personagens. Além disso, a presença de um elemento inesperado de um cavalo em algumas imagens demonstra a imprevisibilidade da geração de imagens através de IA. Melhores técnicas de engenharia de *prompt* propriamente implementados e o aprimoramento iterativo destes argumentos poderiam potencialmente garantir resultados ainda melhores.

4.2.3. Testes da Geração diretamente no Unity

Por fim, foi realizado o teste da geração txt2img com *negative prompt* diretamente no Unity através da biblioteca Unity Stable Diffusion Integration Tool. O objetivo deste trecho é demonstrar a integração entre o Automatic1111 e a ferramenta Unity, sendo a integração de inteligencia artificial diretamente com ferramentas de desenvolvimento de jogos uma questão tecnológica de muito potencial a ser aprimorado em trabalhos futuros. Porém, por conta das limitações da compatibilidade da biblioteca com os métodos necessários para realizar os passos subsequentes deste projeto, ela será utilizada somente neste trecho.

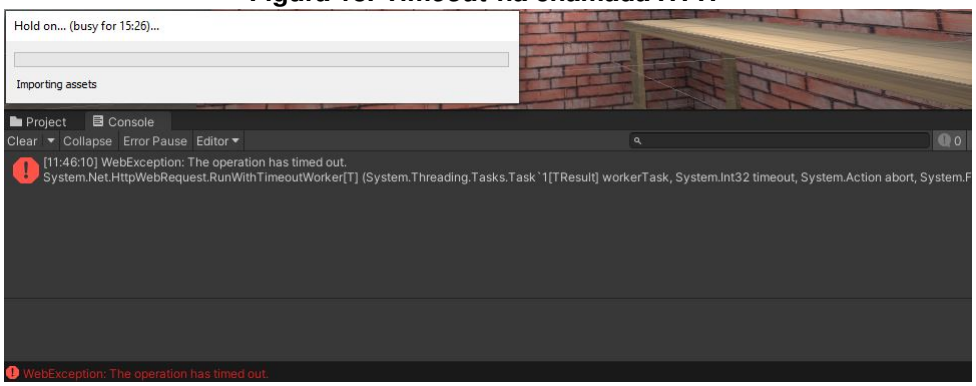
Todos os argumentos como *prompt*, *negative prompt* e modelo utilizado foram configurados diretamente no Unity. O Automatic1111 executando localmente serviu como uma API para chamadas HTTP que recebia de entrada as instruções enviadas pela biblioteca de integração e enviava como saída uma imagem diretamente ao Unity.

Foi utilizado *Prompt* de texto *"full-body, armored noble knight, on white background"*. Como Negative Prompt, foi utilizado *"(holding item), (object in hands), (holding weapon), shield, sword, wearing cape, long hair"*. A geração novamente foi realizada utilizando os modelos "v1-5-pruned-emaonly", "dreamshaper_8", "aZovyaRPGAr-

tistTools_v3", "AnythingV5Ink_v5PrtR".

O primeiro teste utilizou o argumento padrão de 75 *steps*[2],porém por conta dos recursos limitados do dispositivo, a geração de imagem acabou demorando excessivamente e resultando em uma exceção de *timeout*[3] da chamada HTTP (Figura 18), fazendo assim com que a imagem não fosse gerada.

Figura 18. *Timeout* na chamada HTTP



Fonte: O próprio autor (2023)

Com isso, foi diminuído a quantidade de *steps* das próximas gerações e o teste foi refeito. Houve sucesso na geração através de 10, 25 e 40 *steps*, provando assim que a integração do Automatic1111 com o Unity estava funcionado. Para cada um dos modelos, a imagem foi gerada com sucesso e importada automaticamente como textura de um dos objetos presentes no projeto de demonstração da Stable Diffusion Integration Tool (Figura 19). É importante levar em consideração que as gerações foram realizadas na resolução 640x480 para se adequar ao objeto presente no projeto de demonstração, e com isso, tal proporção pode resultar em saídas diferentes dos testes anteriores.

4.3. Testes de Geração de Personagens com posição

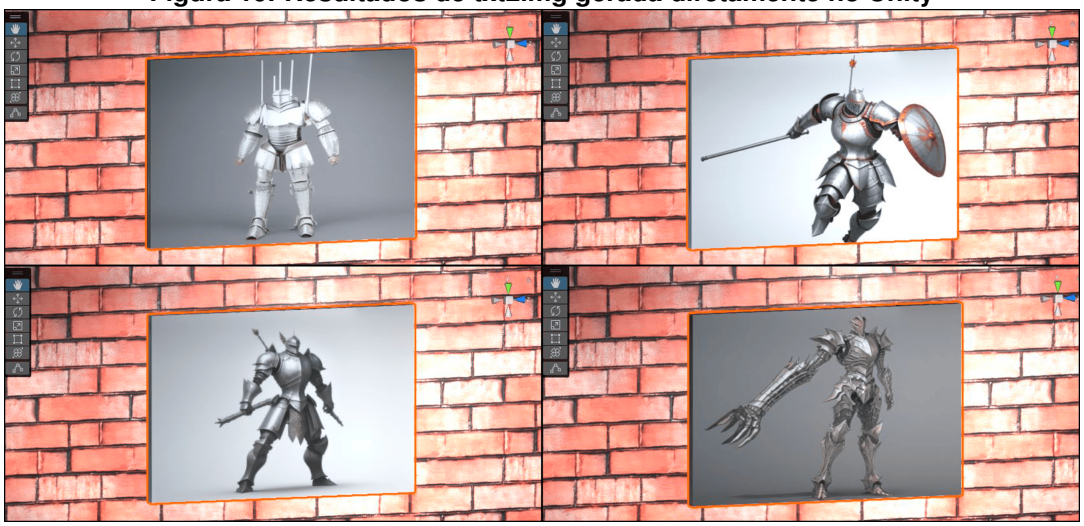
Para a utilização de personagens gerados como ativo artístico no desenvolvimento de jogos, é necessário que este seja possível de anima-los. Visto que para animar um personagem através da técnica de esqueleto é necessário que os membros do personagem sejam divididos um dos outros, o objetivo desse passo é gerar personagens com uma forma que facilite seu recorte.

São utilizados três imagens modeladas através do MagicPoser como referência e, no caso dos métodos img2img, DWOpenPose e Depth, como argumento de entrada para geração. Essas três imagens de referência são um personagem 3D em posição de A-Pose virado para frente, um personagem 3D em posição de lado com os braços virados para trás de seu corpo e uma imagem contendo ambas posições ao mesmo tempo (Figura 20). A posição de A-Pose foi escolhida ao invés da posição de T-Pose pois esta possuí uma posição humana mais natural, que facilita para que o modelo gere imagens mais anatomicamente realistas.

[2]Steps: passos, utilizado no contexto de ciência da computação para se referir ao número de iterações realizados

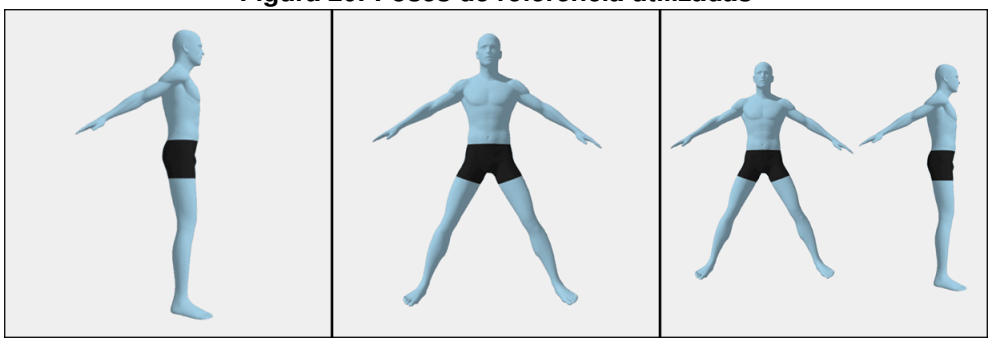
[3]Timeout: utilizado no contexto de ciência da computação para se referir a quando o tempo alocado para um evento expirou

Figura 19. Resultados de txt2img gerada diretamente no Unity



Fonte: O próprio autor (2023)

Figura 20. Poses de referência utilizadas



Fonte: O próprio autor (2023)

Levando em conta os resultados do passo “Testes de Geração de Arte dos Personagens”, é necessário delimitar um único modelo a ser utilizado para evitar resultados redundantes e que não cabem ao escopo deste trecho. A analise de qual modelo utilizar foi realizada de forma completamente subjetiva, de acordo com a opinião dos autores do trabalho, levando em consideração as prévias imagens resultantes. Com isso, foi delimitado a ser utilizado o modelo “dreamshaper_8”, que será o modelo padrão do processador do Automatic1111 deste ponto em diante para novas gerações.

Todas as imagens de personagens de exemplo geradas nessa etapa representam exclusivamente um cavaleiro medieval vestido com armadura. Não foi gerado outros tipos de personagens para evitar que ocorra variações irrelevantes para o escopo dessa seção. Como não houve modificações nas configurações de resolução, gerações resultaram em imagens de saída com resolução padrão de 512x512 *pixels*.

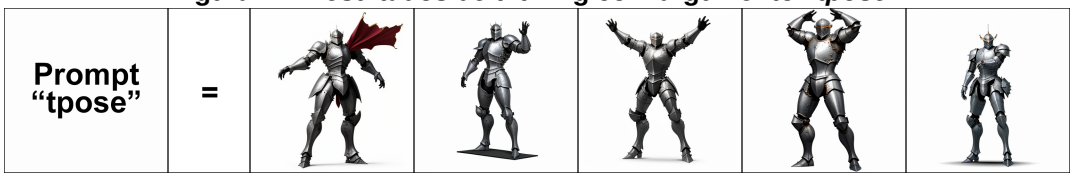
4.3.1. Testes da Geração txt2img com Argumento *"tpose"*

A utilização de *prompts* de texto pode afetar diretamente a posição do personagem em certo grau. Apesar disso, seu uso é limitado e existe técnicas mais consistentes para realizar tal feito. Cabe a este trecho demonstrar a limitação deste método.

Foi utilizado o *prompt* de texto *"full-body, armored noble knight, on white background, tpose"*, sendo o argumento *"tpose"* adicionado somente neste teste com intuito de demonstrar a tentativa de gerar um personagem similar a posição de referência T-Pose virado para frente. Como *negative prompt*, foi utilizado *"(holding item), (object in hands), (holding weapon), shield, sword, wearing cape, long hair"*. A geração foi realizada utilizando somente o modelo "dreamshaper_8".

Nas imagens resultantes (Figura 21), pode-se ser observado que a presença do argumento "tpose" não foi suficiente para manter a consistência dos membros do personagem a ponto de tornar este uma ativo artístico viável para animação.

Figura 21. Resultados de txt2img com argumento *"tpose"*



Fonte: O próprio autor (2023)

4.3.2. Testes da Geração img2img

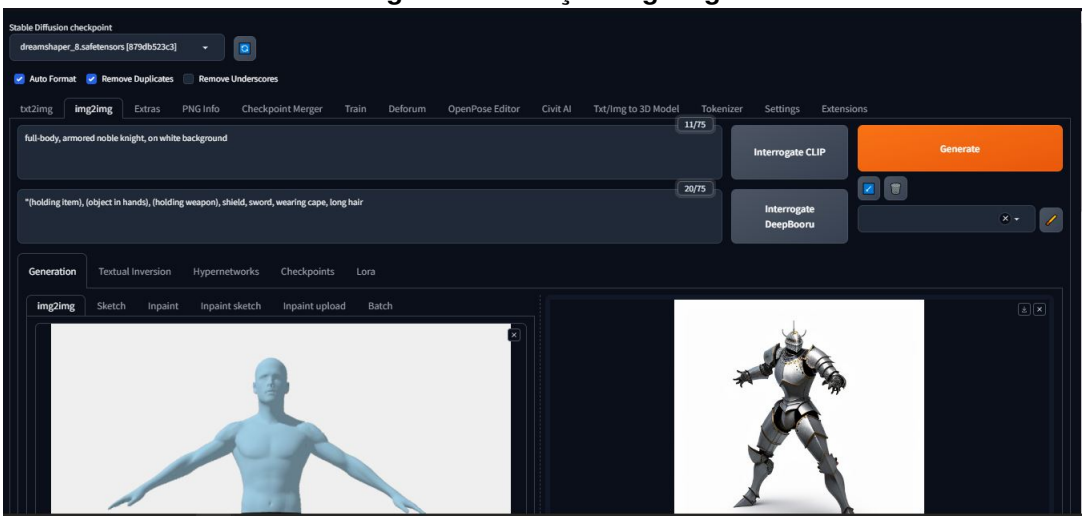
O próximo passo foi testar a utilização do método img2img (Figura 22), que permite não só *prompt* de texto similar aos métodos anteriores, mas também uma imagem base que pode ser usada de modelo pra geração. O método img2img utiliza a imagem de entrada como um dos passos iniciais na geração, ao invés de um ruído gaussiano aleatório. É importante constar que o modelos do Stable Diffusion utilizando o img2img podem levar em consideração diversas características da imagem além de posição de elementos, como também o estilo e cor, para gerar a imagem resultante.

Foi utilizado o *prompt* de texto *"full-body, armored noble knight, on white background"*. Como *negative prompt*, foi utilizado *"(holding item),(object in hands), (holding weapon), shield, sword, wearing cape, long hair"*. A geração foi realizada utilizando somente o modelo "dreamshaper_8".

Para definir a posição do personagem, foi realizado diferentes gerações utilizado como argumento no img2img as entrada de imagens de referência, com personagem em posição de frente (A-Pose), posição de lado e posições frente (A-Pose) + de lado na mesma imagem. Nas imagens resultantes (Figura 23), pode ser observado os seguintes obstáculos deste método para o caso de uso deste trabalho:

- Para personagens de frente, a consistência dos membros dos personagens melhorou consideravelmente, sendo possível gerar personagens que pudessem ser usado

Figura 22. Geração img2img

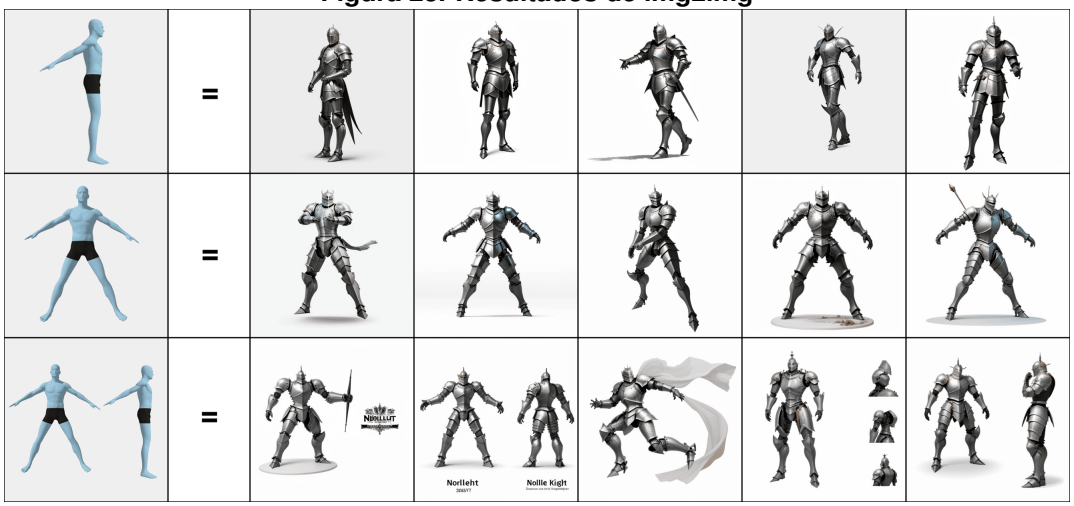


Fonte: O próprio autor (2023)

como ativo artístico. Apesar disso, houve ainda um grau de inconsistência na posição dos mesmos;

- Não foi capaz de gerar nenhum personagem de lado em posição consistente, o que é um grande limitador desta técnica. É possível analisar este problema em ambas as posições frente e frente + lado.

Figura 23. Resultados de img2img



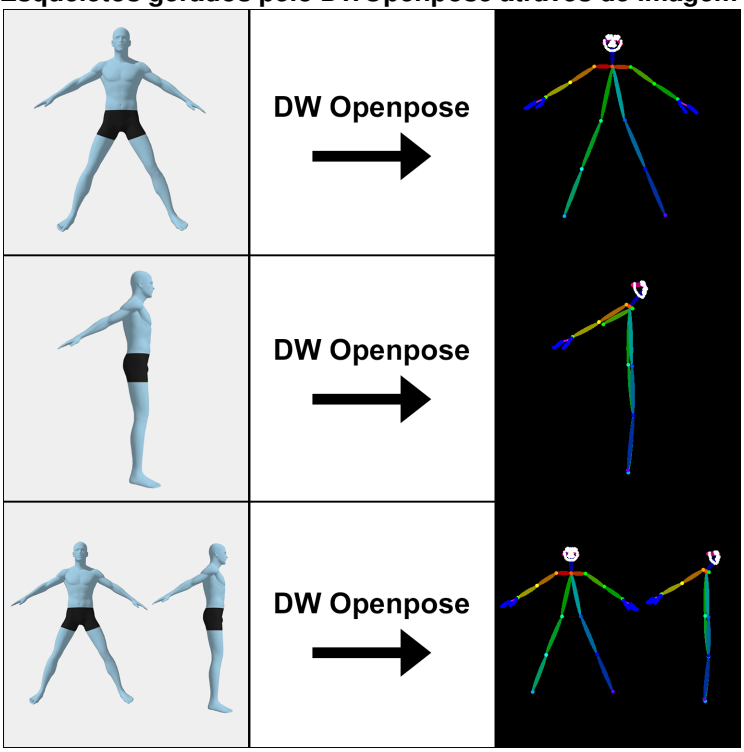
Fonte: O próprio autor (2023)

É visível que houve maior consistência na posição de personagens, principalmente para personagens de frente. Apesar disso, o grau de inconsistência contido na geração de personagens em posição de lado e posições A-Pose + de lado na mesma imagem torna este método inviável para o caso de uso deste projeto. É necessário a utilização de um método que possa garantir um maior grau de importância para a posição desejada de membros do personagem para a geração de arte.

4.3.3. Testes da Geração txt2img com DWOpenpose

Uma das formas mais eficazes de definir a posição de membros de um humanoide gerado pelo Stable Diffusion é através do método Openpose em conjunto com um preprocessador DWOpenpose. Este módulo do ControlNet transformará as imagens de entrada em um esqueleto (Figura 24) que será utilizado para definir a posição dos membros imagem de saída. Como a posição do personagem tem alta importância para o caso de uso deste trabalho, é necessário configurar o módulo para ter um impacto considerável durante a geração do personagem.

Figura 24. Esqueletos gerados pelo DWOpenpose através de imagem de entrada

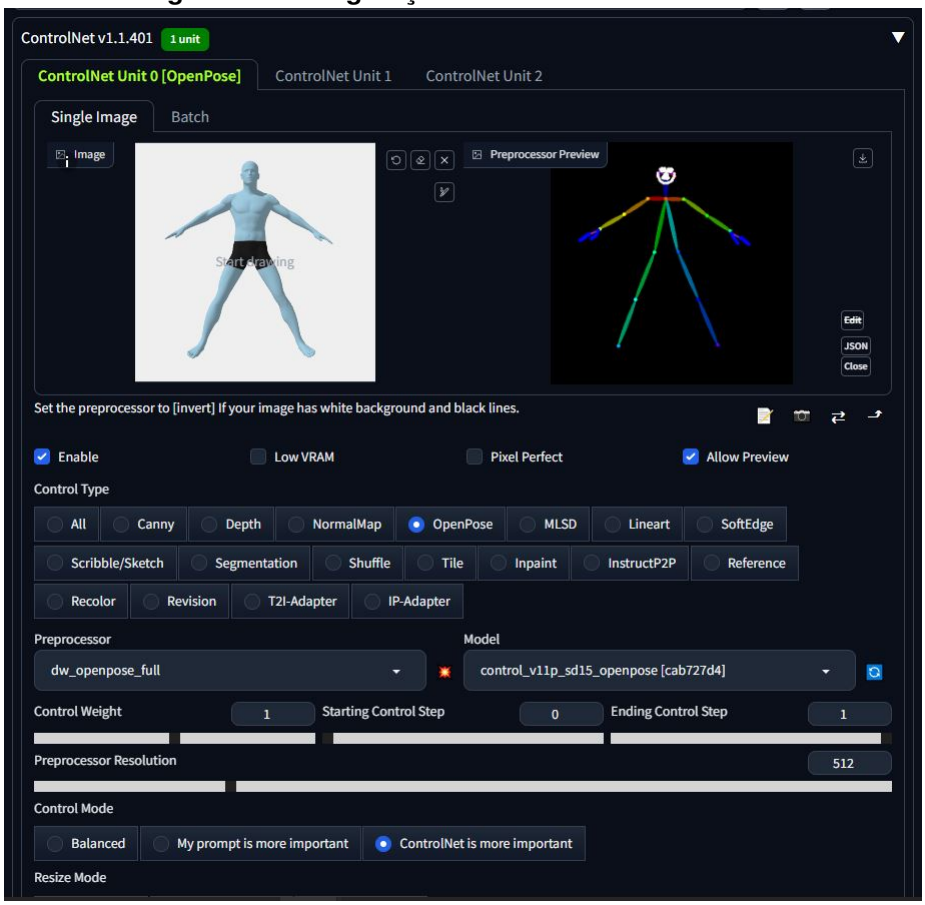


Fonte: O próprio autor (2023)

Foi utilizado o *prompt* de texto *"full-body, armored noble knight, on white background"*. Como *negative prompt*, foi utilizado *"(holding item)", (object in hands), (holding weapon), shield, sword, wearing cape, long hair"*. A geração foi realizada utilizando somente o modelo "dreamshaper_8". Foi habilitada a *''Unit 0"* do ControlNet utilizando um *"ControlType"* do tipo "OpenPose", com esse módulo possuindo os campos *"Preprocessor"* com o argumento "dw_openpose_full", *"Model"* com o argumento "control_v1pp_sd15_openpose" e *"Control mode"* com o argumento *"ContronlNet is more important"* (Figura 25).

Para definir a posição do personagem, foi realizada diferentes gerações utilizado no Openpose como entrada as imagens de referência com personagem em posição de frente (A-Pose), posição de lado e posições frente (A-Pose) + de lado na mesma imagem, que o algoritmo transforma um esqueleto de referência. Nas imagens resultantes (Figura 26), pode ser observado os seguintes obstáculos deste método para o caso de uso deste trabalho:

Figura 25. Configuração da *"Unit 0"* do ControlNet



Fonte: O próprio autor (2023)

- Apesar de grande grau de consistência dos membros mapeados pelo Openpose, em situações em que o personagem era gerado de lado outros membros indesejados acabavam sendo criados juntos, que poderia ser um empecilho para o recorte do personagem;
- Personagens possuíam uma tendência muito grande de serem gerados com saias e outros tipos de vestimentos no quadril, atrapalhando assim a forma de suas pernas;
- No caso das gerações de personagens nas posições de frente + de lado na mesma imagem, o mesmo personagem acabou sendo gerado com pequenas inconsistências em suas duas posições, criando-se a necessidade de alteração manual humana.

Em relação a posição dos membros, os resultados adquiridos utilizando DWOpenpose foram muito satisfatórios. O nível de consistência adquirido neste passo já permitiria com que os personagens gerados fossem utilizados como um ativo artístico. Porém certos dificultadores que atrapalham na forma do personagem poderiam ser mitigados, com o objetivo de diminuir a necessidade de edição manual humana. Busca-se então um método que permita dar ênfase na forma desejada do personagem para ser utilizado em conjunto com os métodos previamente testados

Figura 26. Resultados de txt2img com DWOpenpose

Fonte: O próprio autor (2023)

4.3.4. Testes da Geração txt2img com DWOpenpose e Depth

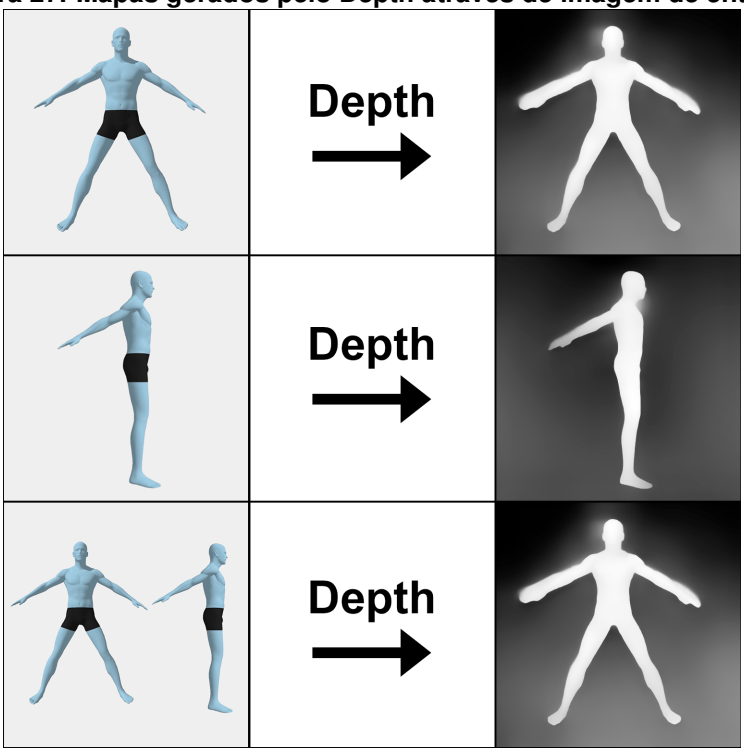
Com o intuito de definir a forma de uma imagem durante a geração e diminuir a ocorrência de elementos indesejados como saias ou membros adicionais, é utilizado o método Depth ao processo de geração. Este método é um dos módulos disponibilizados pelo ControlNet, que transforma uma imagem de entrada em um mapa de profundidade (Figura 27), sendo o mesmo utilizado como um dos argumentos durante os passos da geração. Já que o ControlNet permite até 3 unidades (*units*) simultâneas, é possível combinar este método de forma simultânea ao DWOpenpose.

É importante levar em consideração que definir a forma do personagem tem potencial de ser um grande limitante nas possibilidades artísticas da imagem de saída. Caso não houver um mapa de profundidade adequado para o personagem gerado, é necessário que o Depth não possua um peso muito elevado sob a geração por padrão. Por conta que o mapa de profundidade utilizado é de uma imagem de um personagem de referência genérico, foi realizado os testes com a configuração padrão e também com uma configuração que cede menos prioridade ao método do Depth, para demonstrar este comportamento.

Foi utilizado o *prompt* de texto *"full-body, armored noble knight, on white background"*. Como *negative prompt*, foi utilizado *"(holding item), (object in hands), (holding weapon), shield, sword, wearing cape, long hair"*. A geração foi realizada utilizando somente o modelo "dreamshaper_8". Foi habilitada a *"Unit 0"* do ControlNet utilizando um *"ControlType"* do tipo "OpenPose", com esse módulo possuindo os campos *"Preprocessor"* com o argumento "dw_openpose_full", *"Model"* com o argumento "control_v1pp_sd15_openpose" e *"Control mode"* com o argumento *"ContronlNet is more important"*. Foi também habilitado a *"Unit 1"* do ControlNet utilizando um *"ControlType"* do tipo "Depth", com esse módulo possuindo os campos *"Preprocessor"* com o argumento "depth_midas" e *"Model"* com o argumento "control_v1pp_sd15_depth" (Figura 28).

Para definir a forma do personagem, foi realizado diferentes gerações utilizado no

Figura 27. Mapas gerados pelo Depth através de imagem de entrada



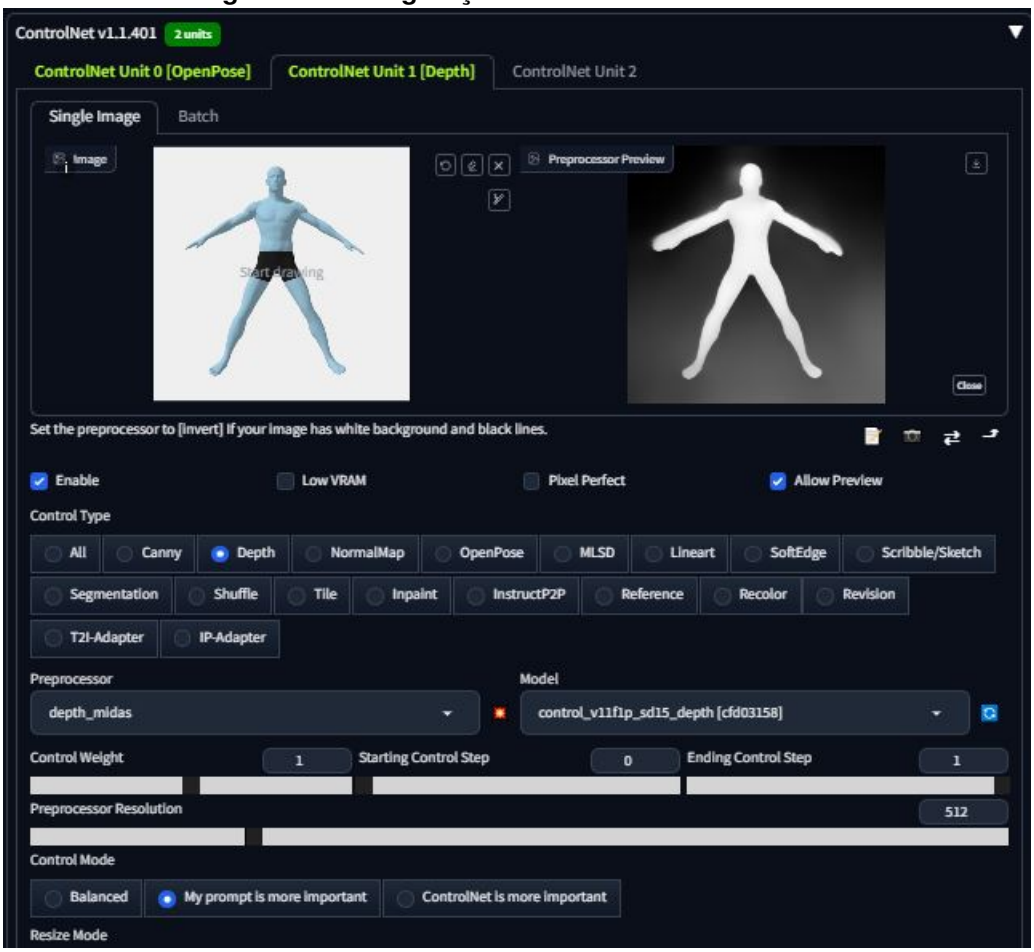
Fonte: O próprio autor (2023)

Depth com *"Control mode"* configurado para o argumento *"Prompt is more important"* e também *"Balanced"*. Somente a posição frente + lado foi utilizado como entrada. Nas imagens resultantes (Figura 29), pode ser observado resultados muito significativos, sendo esses:

- Não houve nenhuma ocorrência de personagens com elementos como saias;
- Em questão de membros adicionais, houve uma diminuição considerável utilizando o *"Control mode"* como *"Prompt is more important"* e uma diminuição quase que total utilizando o *"Control mode"* como *"Balanced"*.
- Utilizar o *"ControlMode"* como *"Balanced"* resultou em imagens de grande consistência em questão de posição e forma, porém é visto que utilizar um mapa de profundidade genérico em conjunto com o módulo do Depth diminui a variedade artística das imagens geradas e limita sua forma;
- Pequenas diferenças de detalhes entre as duas posições nas gerações na posição frente + lado, pois o método Depth não tem capacidade de mitigar tais inconsistências

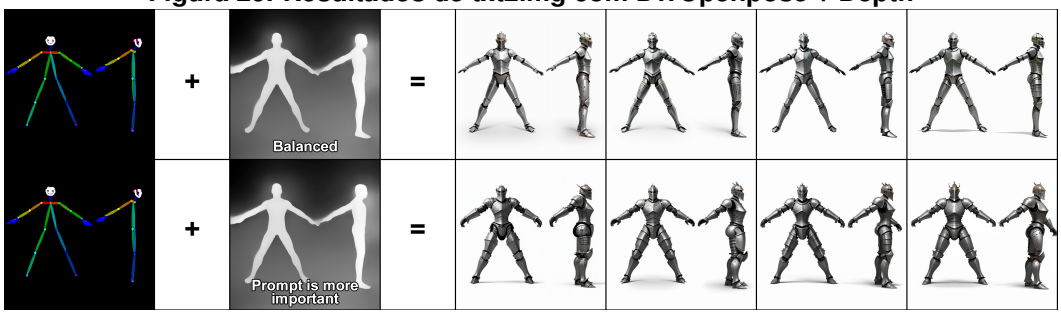
O uso do Depth em conjunto com DWOpenpose se demonstrou a melhor forma de definir a posição do personagem. A presença de elementos indesejados diminuíram consideravelmente e a consistência da posição dos personagens aumentou, permitindo que os personagens fossem utilizados como ativo artístico válido com um grau menor de interação manual humana. Apesar disso, é necessário analisar qual *"Control mode"* utilizar de acordo com seu caso de uso. *'Control mode"* configurado como *"Balanced"* resultaria em imagens com mais consistência que poderiam ser utilizadas como ativos com menos interação manual humana, mas que necessariamente precisaria de um mapa de

Figura 28. Configuração da "Unit 1" do ControlNet



Fonte: O próprio autor (2023)

Figura 29. Resultados de txt2img com DWOpenpose + Depth



Fonte: O próprio autor (2023)

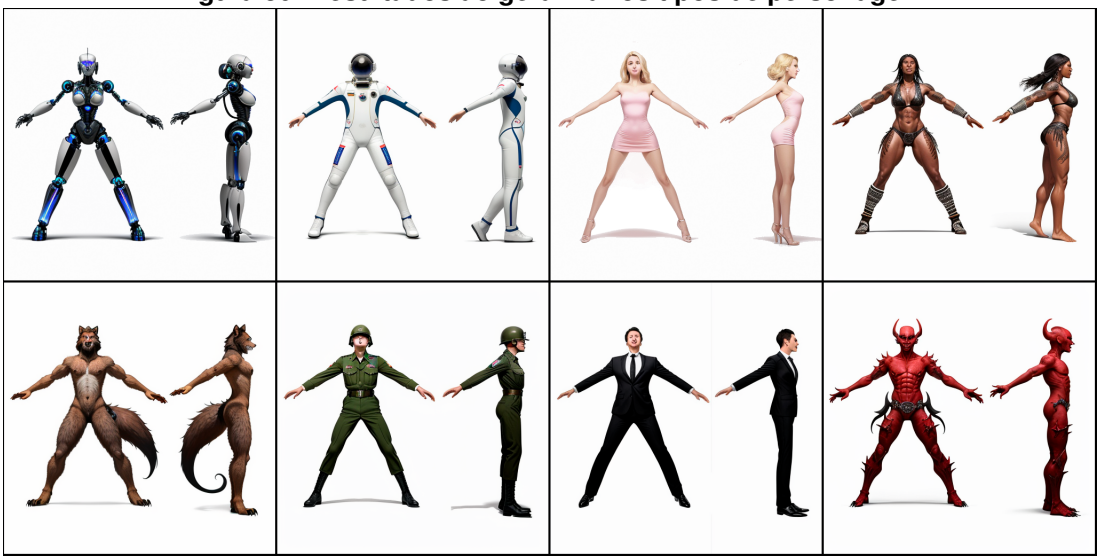
profundidade adequado para o caso de uso utilizado. Por sua vez, o "*Control mode*" configurado como *"Prompt is more important"* consegue gerar resultados satisfatórios utilizando um mapa de profundidade genérico, porém gera mais inconsistências na posição de outros membros que necessitariam de edição.

4.4. Adaptando técnicas para outros usos

Durante as etapas anteriores foi utilizado um *prompt* de texto para gerar um personagem de exemplo de um cavaleiro medieval vestido com uma armadura sobre um fundo branco. Apesar de que o tipo do personagem se manteve constante anteriormente, é importante ressaltar que a técnica demonstrada nesse artigo não estando limitado a um só tipo de personagem ou estilo.

Os métodos estabelecidos tem a capacidade de ser escaláveis para inúmeros outros tipos de personagens e estilos. Para demonstrar a variedade de personagens que podem ser gerada, foi utilizado o modelo "dreamshaper_8" e *prompts* de texto variados para gerar diversos tipos de personagens distintos, como por exemplo um robô, um astronauta, uma atriz, uma guerreira, um lobisomem, um soldado, um executivo e um demônio (Figura 30). Muitos outros resultados seriam possíveis, sendo o nível em que podem ser satisfatórios varia de acordo com utilização de boas práticas de engenharia de *prompt*, da utilização, do treinamento de um modelo adequado pro caso de uso desejado e ou até mesmo da utilização de outras técnicas e parâmetros disponíveis para o Automatic1111 e ControlNet que não foram abordados nesse artigo.

Figura 30. Resultados ao gerar vários tipos de personagem



Fonte: O próprio autor (2023)

4.5. Modificações para tornar em um ativo artístico utilizável

Para que seja possível animar os personagens 2D através de animação esqueletal é necessário possuir uma imagem do personagem com seus membros dispostos de uma forma que estejam divididos e separados uns dos outros (Figura 31). Já que mesmo o personagem está disposto nas posições de frente e lado, qualquer elemento ou detalhe indesejado o deve ser removido de tal forma que as duas poses se mantenham consistentes entre uma e outra. Por fim, a imagem final deve possuir um fundo transparente e ser do formato PNG.

A questão de como realizar remoção de inconsistências de um personagem e a divisão dos membros programaticamente é um grande obstaculo tecnológico a ser superado por trabalhos futuros. É importante que o Stable Diffusion consiga gerar os detalhes

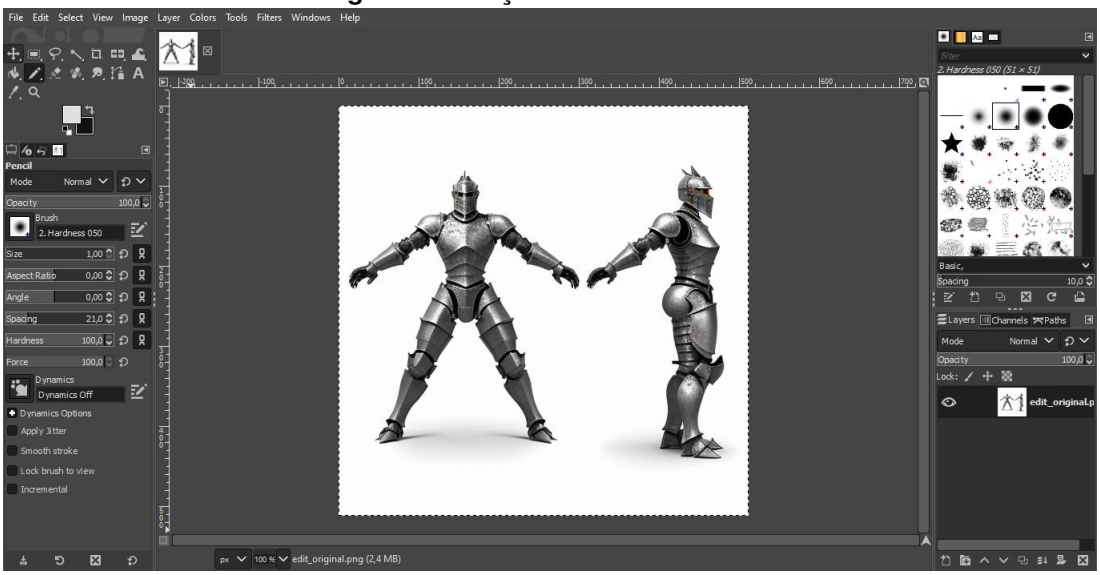
Figura 31. Exemplo de Personagem com membros divididos

Fonte: Elias (2020)

e forma do personagem de tal forma que o mesmo fique idêntico em ambas posições de frente e lado. Posteriormente, seria necessário que um algoritmo consiga identificar quais regiões da imagem de entrada que possuem cada membro do personagem, e por fim, gerar uma imagem de saída para cada posição contendo seus membros separados e divididos corretamente, que necessitaria um novo algoritmo de processamento de imagens só para este propósito.

Por conta dessas limitações tecnológicas e por razões de demonstração, será realizado a edição manual dos personagens para transforma-lo em um ativo artístico válido. Para isso, é utilizado a ferramenta de edição de imagens GIMP (Figura 32) para o tratamento de inconsistências e divisão de membros. Após a realização deste processo, haverá como saída duas imagens por personagem, representando o ativo artístico dos membros divididos do personagem na posição de frente e de lado.

Figura 32. Edição utilizando o GIMP



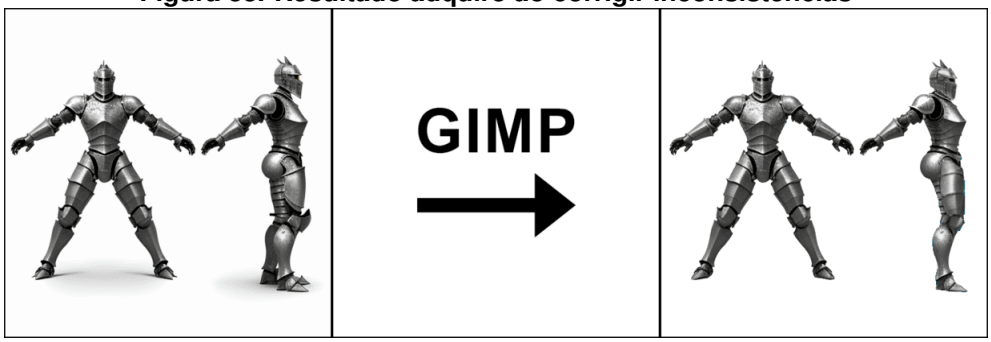
Fonte: O próprio autor (2023)

4.5.1. Corrigindo inconsistências

Através do GIMP, é editado manualmente a imagem do personagem gerada com o intuito de remover as inconsistências da geração. As inconsistências mitigadas vão desde o estilo do personagem até sua forma, se aplicando a posição de frente e de lado.

Primeiramente são eliminados elementos indesejados, como membros adicionais e sombra. Posteriormente, são feitos modificações para que o estilo de ambas as gerações do personagem de lado e de frente sejam o mais idênticas possíveis. Após este processo, a imagem resultante (Figura 33) é consideravelmente mais consistente.

Figura 33. Resultado adquiro ao corrigir inconsistências



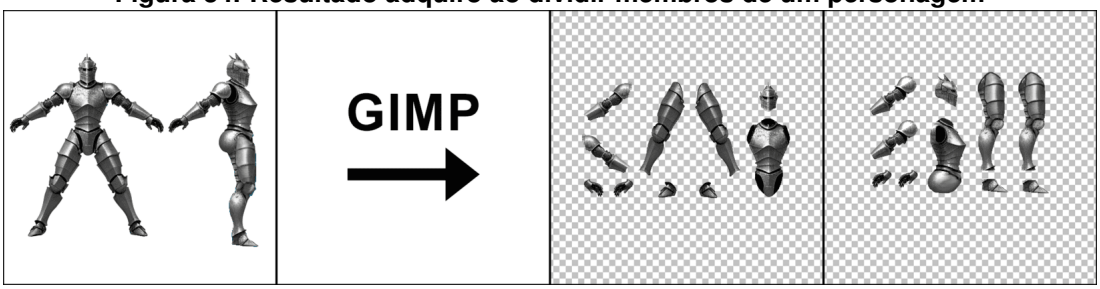
Fonte: O próprio autor (2023)

4.5.2. Separando membros do personagem

Por fim, é utilizado novamente o GIMP para realizar o recorte dos membros do personagem. Caso o recorte dos membros afete a forma do personagem (no caso de por exemplo, braço gerado que tampa a visibilidade do tronco), é necessário editar o ativo artístico para manter garantir a forma desejada do membro. É importante também tornar o fundo da imagem transparente e salvar a mesma no formato PNG.

A quantidade de membros resultantes depende de como será realizado o esqueleto durante o processo de animação. Mas recomenda-se no mínimo o recorte de cabeça, tronco, perna mãos e pés. Após realizar o recorte com êxito, este processo resultará em duas imagens PNG de saída com os membros divididos e de fundo transparente (Figura 34), que estariam prontas para efetivamente serem utilizadas como um ativo artístico.

Figura 34. Resultado adquiro ao dividir membros de um personagem

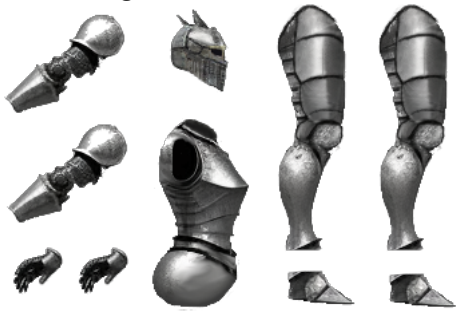


Fonte: O próprio autor (2023)

4.6. Integrando novo personagem ao Unity

A partir do momento que possui disponível um personagem consistente e de membros divididos que possa ser utilizado como ativo artístico válido (Figura 35), é necessário realizar a integração do mesmo em um cenário do Unity. Este trecho abrangerá desde o momento que o ativo é importado ao Unity até o momento que o personagem se torna viável para animações. Durante esse passo, será demonstrado explicitamente a integração de um personagem em posição de lado, porém é importante constar que os métodos realizados nesse passo se aplicam também a um personagem em posição de frente. Teoricamente, seria possível reutilizar o esqueleto, pesos e animação criados para diferentes personagens.

Figura 35. Personagem de lado com membros divididos



Fonte: O próprio autor (2023)

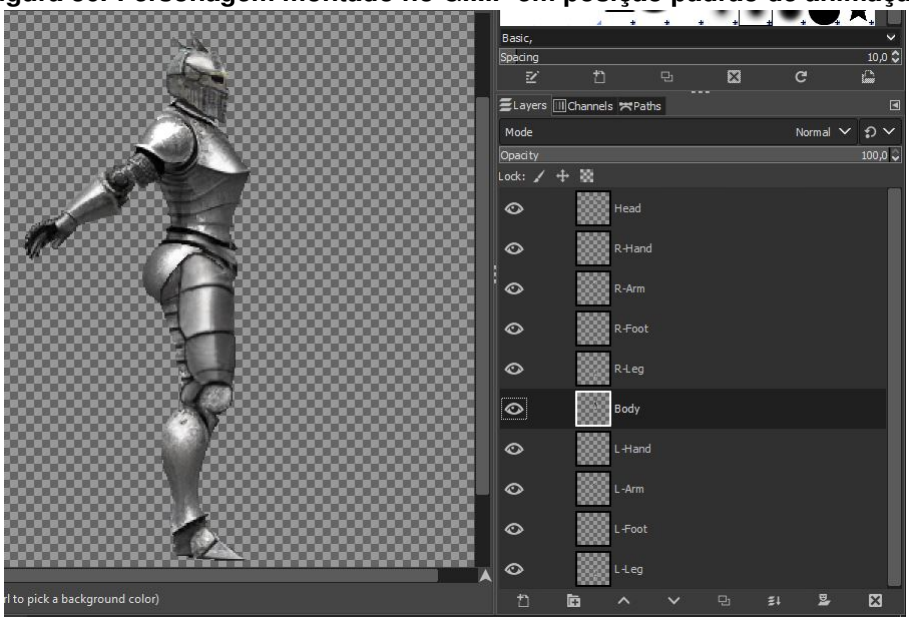
4.6.1. Importando os ativos artísticos

Para utilizar um ativo no sistema de animação esqueletal 2D do Unity é necessário agrupar os membros novamente na posição de animação padrão que o personagem ficará. A forma de agrupar pode ser feito de duas formas:

- Importar o arquivo de membros separados diretamente no Unity: é necessário importar a imagem PNG contendo os membros separados do personagem e posteriormente monta-lo na posição esperada dentro da própria ferramenta;
- Importando um arquivo PSD ou PSB do personagem: o personagem é montado com os membros na posição esperada utilizando o sistema de camadas de imagem das ferramentas Photoshop ou GIMP, onde cada membro possuí sua camada própria. O arquivo é salvo no formato PSD ou PSB e por fim é importado para o Unity;

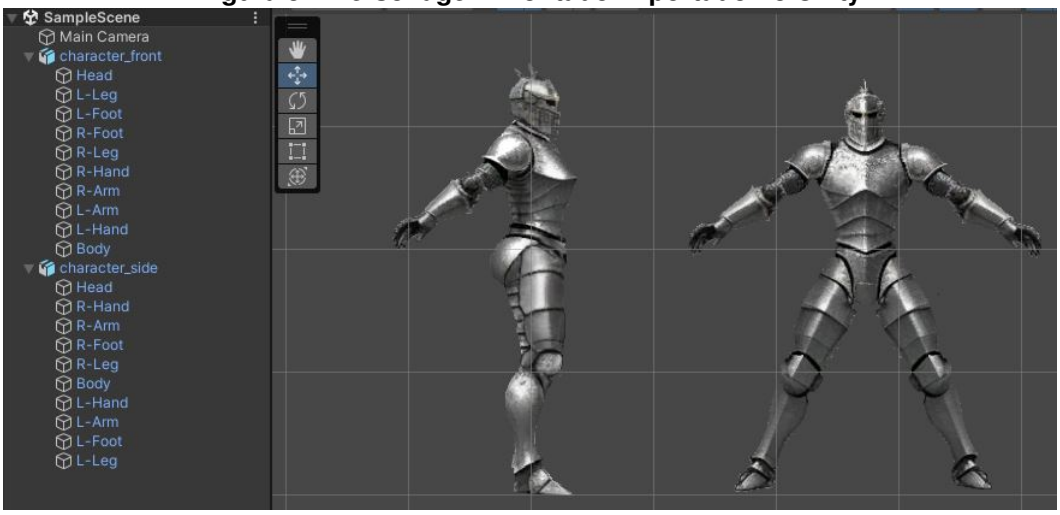
Como padrão, será utilizado a metodologia “Importando um arquivo PSD ou PSB do personagem”. Essa metodologia é mais comumente utilizada para personagens 2D, permitindo um maior grau de flexibilidade para realizar futuras potenciais modificações na arte do personagem. Com isso, foi utilizado a imagem contendo os membros separados do personagem para monta-lo em sua posição padrão de animação no GIMP (Figura 36) e posteriormente arquivo PSD resultante foi importado ao Unity (Figura 37).

Figura 36. Personagem montado no GIMP em posição padrão de animação



Fonte: O próprio autor (2023)

Figura 37. Personagem montado importado no Unity



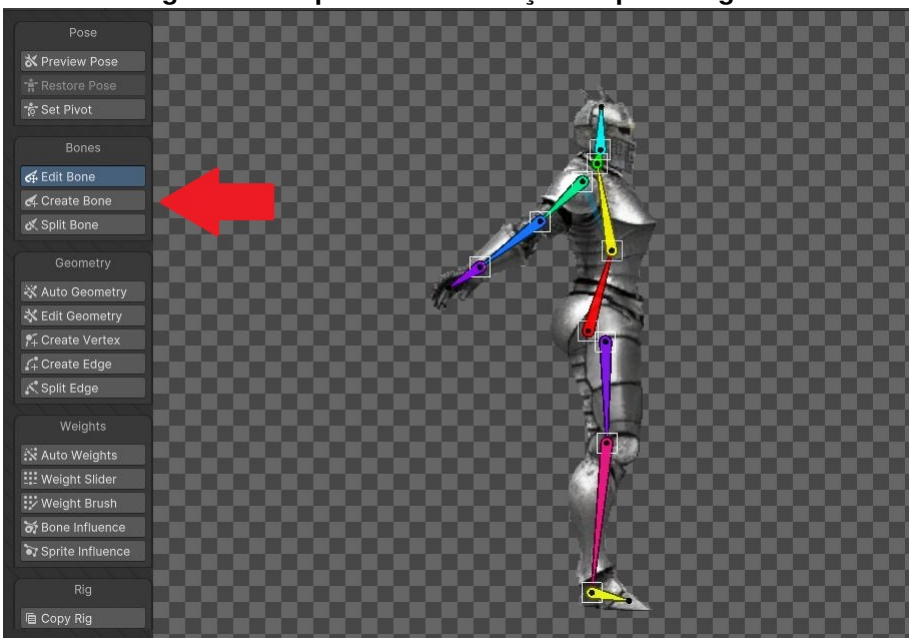
Fonte: O próprio autor (2023)

4.6.2. Criando o Esqueleto e Pesos

Após importar um personagem, é necessário criar o esqueleto dos membros (Figura 38) e o peso em que os afetam (Figura 39). Atualmente no ciclo de desenvolvimento artístico de jogos, o artista realiza manualmente criação de cada osso do esqueleto, e após estarem disponibilizados na posição correta, modifica o peso de cada um deles sobre seus respectivos membros.

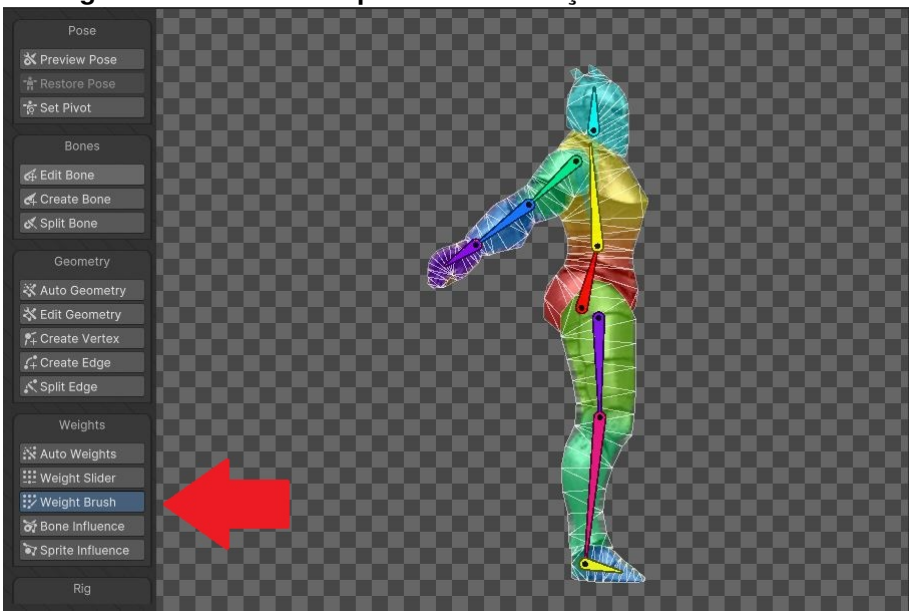
O peso (ou *weight*, do inglês) representa o quanto cada um dos ossos do esqueleto afetará cada membro ao ser manipulado durante a animação. Em uma das opções do Unity, é possível gerar os pesos automaticamente levando em consideração a proxi-

Figura 38. Esqueleto de animação do personagem



Fonte: O próprio autor (2023)

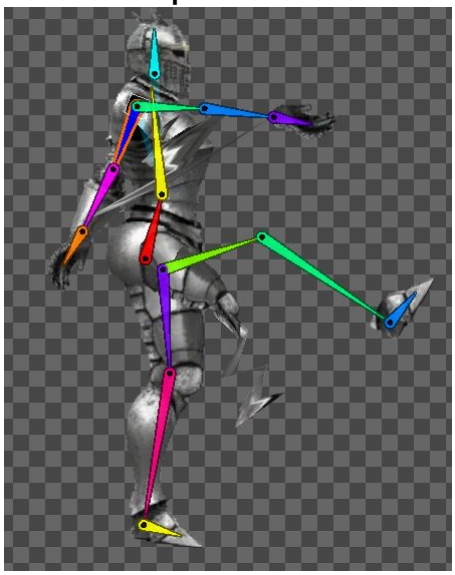
Figura 39. Pesos do Esqueleto de animação sobre os membros



Fonte: O próprio autor (2023)

midade do osso e membro. Porém, mesmo após a geração automática dos pesos, estes devem ser posteriormente modificados manualmente, pois ossos podem acabar afetando incorretamente membros não desejados (Figura 40).

Figura 40. Exemplo de Pesos do Esqueleto afetando membros de forma incorreta

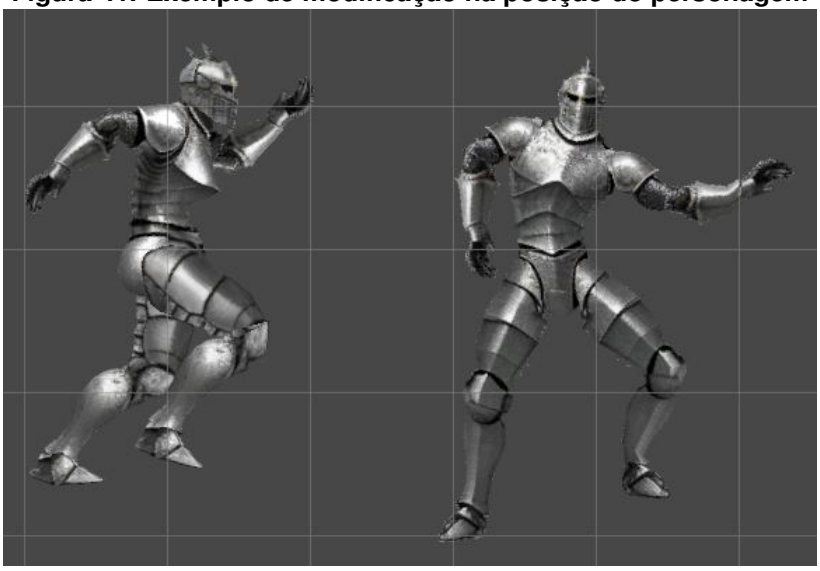


Fonte: O próprio autor (2023)

4.6.3. Modificando a Posição do Personagem e Animando

Com o personagem com seu esqueleto de animação e pesos já propriamente estabelecidos, já é possível modificar sua posição e animação. A posição do personagem pode ser modificada através do reposicionamento e rotação dos ossos, permitindo assim que o desenvolvedor disponha o personagem na pose que desejar.

Figura 41. Exemplo de modificação na posição do personagem

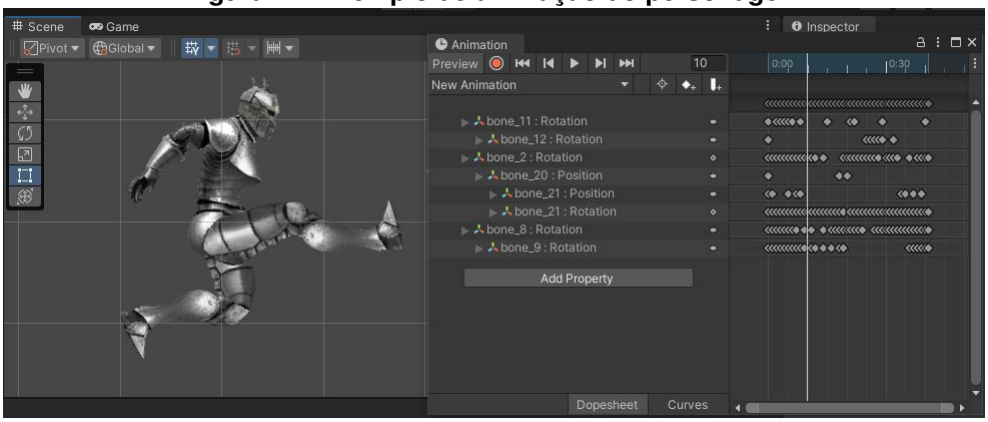


Fonte: O próprio autor (2023)

Em questão de animação esqueletal, os artísticos ativos podem ser animados através da definição de um estado inicial e final para o membro de um personagem, através de *keyframes*, onde o próprio Unity realiza a transição entre estados durante um intervalo

de tempo (Figura 41). Além disso, a ferramenta permite o gerenciamento individual de cada quadro, permitindo assim que o desenvolvedor possa ajustar detalhes da animação de acordo com a necessidade e também utilizar técnicas de animação mais manuais, como *stop motion*.

Figura 42. Exemplo de animação do personagem



Fonte: O próprio autor (2023)

5. Conclusão

Observam-se resultados satisfatórios na geração de imagens de personagens 2D mediante a utilização de modelos de difusão e sua subsequente adaptação para servirem como ativo artístico em jogos eletrônicos. Através da utilização da ferramenta Automatic1111, foram apresentados passos para sua configuração e utilização no contexto geração de personagens 2D, resultando em personagens de variados tipos e estilos, na posição de frente e lado, com maior grau de qualidade estética e de alto nível de consistência na posição de seus membros. Complementarmente, foi demonstrado comparações de como diferentes modelos e parâmetros podem afetar a geração e também a integração direta através do envio direto de imagens geradas para a ferramenta desenvolvimento de jogos Unity. Por fim, foi demonstrado as alterações manuais necessárias para tornar a imagem gerada em um ativo artístico válido, assim como importá-la e prepará-la para o processo de animação dentro do ambienta da ferramenta Unity.

Anteriormente a este estudo, um dos principais desafios no emprego de inteligencia artificial para esse contexto residia na capacidade da geração de personagens de qualidade e com posição consistente. Os recentes avanços nos modelo de difusão em relação às redes adversárias generativas na área de geração de imagens permitem um grau maior de qualidade e variedade nos resultados obtidos. Além disso, a adoção de métodos como DWOpenpose e Depth, através da arquitetura ControlNet, asseguram um grau de consistência de posição de membros essencial para a utilização do personagem gerado como ativo artístico em um jogo eletrônico.

O avanço de técnicas e tecnologias para a geração de personagens 2D por meio de inteligência artificial tem potencial de promover uma revolução no mercado artístico. Isso não se restringe apenas aos jogos eletrônicos, mas também abrange outros campos como animação de filmes e vídeos. A adoção de metodologias similares as demonstradas neste artigo tem potencial de diminuir consideravelmente o tempo de desenvolvimento

artístico, eventualmente resultando em uma economia significativa de recursos financeiros e humanos. Apesar do escopo deste trabalho ser dedicado a finalidade de desenvolvimento de jogos, métodos similares podem ser utilizados para filmes, séries, videoclipes musicais, marketing, educação, aplicações iterativas, comunicação corporativa, entre outras áreas.

No entanto, é possível observar limitações que dificultam a plena automatização no processo de geração de arte de personagens 2D. Um desafio tecnológico significativo a ser superado reside na exigência de intervenção manual utilizando ferramentas de edição de imagens para tornar o personagem em um recurso artístico funcional, pois atualmente é necessário que um humano corrija as discrepâncias nas posições diferentes do mesmo personagem (no caso deste trabalho, posições de frente e lado) e também realize a divisão de seus membros. Além disso, é pertinente citar como obstáculo a necessidade ocasional de realizar várias gerações até encontrar uma imagem com resultados satisfatórios.

Através dos conhecimentos, métodos e tecnologias abordados neste trabalho, cria-se a oportunidade de novos avanços a serem desbravados por trabalhos futuros. Destacam-se a necessidade de novos algoritmos para dividir membros de personagens programaticamente e a novos métodos para integração mais direta entre geração de imagens e ferramentas de desenvolvimento de jogos. Além disso, vê-se a demanda em adaptar a geração de modelos de difusão para diminuir inconsistências encontradas no mesmo personagem quando este foi gerado em duas ou mais posições diferentes na mesma imagem. É desejável também a exploração de outras técnicas e parâmetros do Automatic1111 e ControlNet que não foram abordadas neste artigo, incluindo o treinamento de novos modelos de difusão e *Low-Rank Adaptations* (LoRAs) dedicados ao caso de uso especifico. Com futuros avanços em técnicas de estimação de posição, como por exemplo DWOpenpose, a geração pode se estender para personagens não humanoides, como animais ou monstros Por fim, futuros trabalhos podem analisar a possibilidade de implementação de geração procedural em tempo de execução, enquanto o jogo está em andamento.

Referências

Arash Vahdat, K. K. (2022). Improving diffusion models as an alternative to gans, part 1.

Blizzard Entertainment (1996). Diablo. Disponível em: `https://www.gog.com/game/diablo`. Acessado em: maio de 2023.

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794.

Elias, D. (2020). Unity 2d animation, part 1 - bones & rig — unity tutorial — notslot.

Epyx (1985). Rogue. Disponível em: `https://store.steampowered.com/app/1443430/Rogue/`. Acessado em: abril de 2023.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Hollingsworth, B. and Schrum, J. (2019). Infinite art gallery: a game world of interac-

tively evolved artwork. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 474–481. IEEE.

Hong, S., Kim, S., and Kang, S. (2019). Game sprite generator using a multi discriminator gan. *KSII Transactions on Internet and Information Systems (TIIS)*, 13(8):4255–4269.

Horsley, L. and Perez-Liebana, D. (2017). Building an automatic sprite generator with deep convolutional generative adversarial networks. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 134–141. IEEE.

Keith, C. (2010). *Agile game development with Scrum*. Pearson Education.

Kumar, K. and Thakur, G. S. M. (2012). Advanced applications of neural networks and artificial intelligence: A review. *International journal of information technology and computer science*, 4(6):57.

Liu, V. and Chilton, L. B. (2022). Design guidelines for prompt engineering text-to-image generative models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–23.

Luger, G. (2009). Artificial intelligence: structures and strategies for complex problem solving. 6-th ed. *Boston: Adisson Wesley*.

Maxis (2008). Spore. Disponível em: `https://www.spore.com/`. Acessado em: maio de 2023.

MicroProse (1991). Civilization. Disponível em: `https://civilization.com/pt-BR/civilization-1/`. Acessado em: abril de 2023.

Mojang (2011). Minecraft. Disponível em: `https://www.minecraft.net/`. Acessado em: maio de 2023.

Molera, L. M. (2021). Synthetic image generation using gans.

Mukundan, R. and Mukundan, R. (2012). Skeletal animation. *Advanced Methods in Computer Graphics: With examples in OpenGL*, pages 53–76.

Ng, K. (2017). Ninja sprite (free) — unity asset store.

Ongsulee, P. (2017). Artificial intelligence, machine learning and deep learning. In *2017 15th international conference on ICT and knowledge engineering (ICT&KE)*, pages 1–6. IEEE.

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2020). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. 2022 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*.

Schell, J. (2008). *The Art of Game Design: A book of lenses*. CRC press.

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

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Tiobe (2023). Tiobe index for may 2023.

Togelius, J., Kastbjerg, E., Schedl, D., and Yannakakis, G. N. (2011). What is procedural content generation? mario on the borderline. In *Proceedings of the 2nd international workshop on procedural content generation in games*, pages 1–6.

Unity (2023). 2d animation for building skeletal animation — unity.

Volz, V., Schrum, J., Liu, J., Lucas, S. M., Smith, A., and Risi, S. (2018). Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the genetic and evolutionary computation conference*, pages 221–228.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Yang, Z., Zeng, A., Yuan, C., and Li, Y. (2023). Effective whole-body pose estimation with two-stages distillation. *arXiv preprint arXiv:2307.15880*.

Young, C. J. (2021). 7. unity production: Capturing the everyday game maker market. *Game production studies*, page 141.

Zhang, L., Rao, A., and Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models.

Zwettler, G. A., Praschl, C., Baumgartner, D., Zucali, T., Turk, D., Hanreich, M., and Schuler, A. (2021). Three-step alignment approach for fitting a normalized mask of a person rotating in a-pose or t-pose essential for 3d reconstruction based on 2d images and cgi derived reference target pose. In *VISIGRAPP (5: VISAPP)*, pages 281–292.