Processamento de Imagens - Universidade Federal de Rondônia.

Jáder Louis de Souza Gonçalves[1]

Nicolas Figueiredo Cavalcante Sales[2]

Wyllgner França de Amorim[3]

Orientador:Dr. Lucas Marques de Cunha[3]

https://github.com/visagetrack-project

1. Introdução

Em um mundo cada vez mais dinâmico e conectado, a busca por otimização e personalização se torna cada vez mais presente. No âmbito educacional e corporativo, essa necessidade se traduz na busca por métodos mais eficientes para garantir a presença e o engajamento dos funcionários ou alunos, elementos cruciais para o sucesso do processo de aprendizagem e produtividade. Tradicionalmente, a chamada de presença é realizada de forma manual, um processo moroso, sujeito a erros e que demanda tempo valioso que poderia ser dedicado a atividades mais produtivas. A mensuração do engajamento, por sua vez, é ainda mais desafiadora, muitas vezes se limitando a métodos subjetivos e pouco precisos, como a observação individualizada do comportamento em sala de aula ou reuniões. Essas deficiências geram consequências diretas na qualidade do ensino e na produtividade no trabalho. A falta de um acompanhamento preciso da presença pode levar a distorções nos índices de frequência e dificultar a identificação das pessoas em risco de evasão ou desmotivação. A dificuldade em medir o engajamento de forma objetiva impede a compreensão profunda do nível de participação e interesse dos indivíduos, privando educadores e gestores de informações valiosas para a personalização das estratégias de ensino e trabalho.

Diante desse cenário, o projeto Visage Track surge como uma solução inovadora e disruptiva. Através da utilização de tecnologias de reconhecimento

facial e análise de expressões, o Visage Track automatiza a chamada de presença e fornece análises detalhadas sobre o engajamento dos alunos e colaboradores, abrindo portas para um novo mundo de possibilidades na educação e no trabalho.

2. Revisão de Literatura

2.1 Unity

Segundo Santos (2021), o Unity se destaca como um motor de jogo multiplataforma, viabilizando a criação de games 2D e 3D para desktops, consoles, dispositivos móveis, realidade virtual e aumentada. Sua interface intuitiva e amigável facilita o aprendizado, tornando-o acessível tanto para iniciantes quanto para profissionais experientes. O Unity oferece um conjunto robusto de ferramentas, abrangendo modelagem 3D, animação, física, scripting, inteligência artificial, áudio e efeitos visuais. Essa amplitude permite o desenvolvimento de jogos de diversos gêneros e estilos, desde plataformas simples até títulos complexos com gráficos realistas. Um dos pontos fortes do Unity reside na sua comunidade vibrante e ativa, que oferece suporte e compartilha conhecimentos através de fóruns, tutoriais, plugins e assets gratuitos. Essa comunidade contribui para a constante atualização e aprimoramento do motor, impulsionando a inovação na indústria de games.

2.1.1 Criação de Ambientes

A documentação oficial da Unity Technologies (2023)[1], define ambientes Unity, também conhecidos como "Scenes", como espaços virtuais onde os jogos se desenrolam. São como palcos 3D onde objetos, personagens, efeitos visuais e outros elementos são organizados para compor uma experiência interativa. Cada ambiente possui características próprias, como geometria, iluminação, física e scripts que definem o comportamento e as interações dentro do espaço. Essa flexibilidade permite aos pesquisadores criar ambientes customizados para atender às necessidades específicas de suas pesquisas.

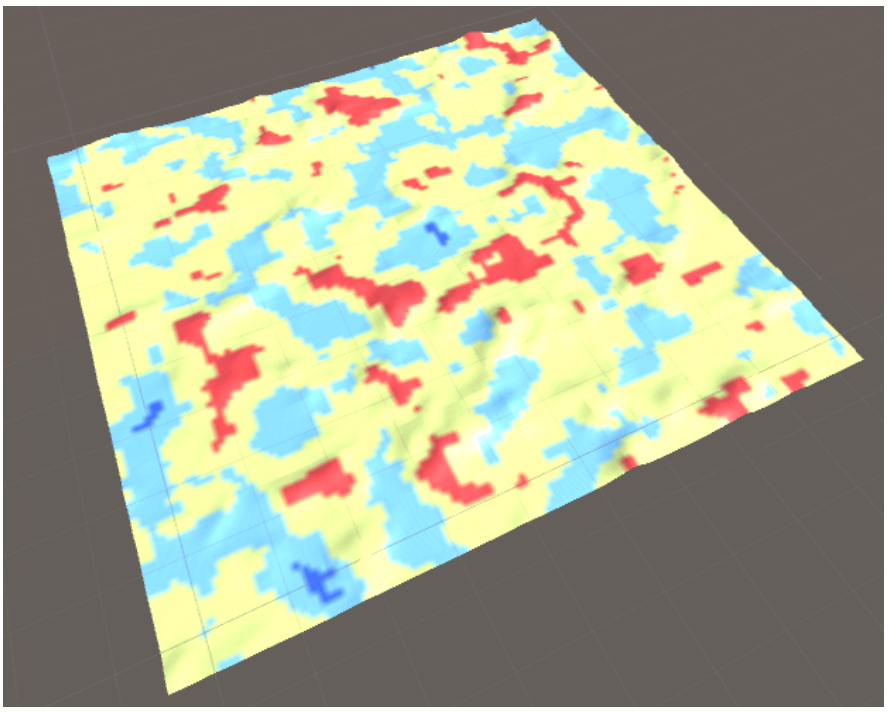
2.1.2 Geração Procedural

Segundo Oliveira (2023), a geração procedural desponta como uma ferramenta poderosa na criação de conteúdo digital, abrindo um universo de possibilidades para artistas e desenvolvedores. Através de algoritmos

sofisticados, mapas, texturas, modelos 3D, músicas e até histórias são gerados automaticamente, com base em regras e parâmetros predefinidos. Em vez de esculpir cada elemento manualmente, o artista define as características desejadas e o algoritmo se encarrega de dar vida à sua visão, moldando o conteúdo de acordo com as especificações. As vantagens dessa técnica são numerosas e atraentes. A economia de tempo e recursos é significativa, liberando tempo para o artista se concentrar em aspectos mais criativos do projeto. Além disso, a geração procedural permite a criação de mundos e experiências imensas e complexas, antes impossíveis de serem realizados manualmente. Imagine explorar um universo infinito em um jogo, com planetas, estrelas e galáxias gerados proceduralmente, cada um com suas características únicas, ou navegar por um mapa proceduralmente gerado em um RPG, com cidades, florestas e masmorras repletas de detalhes e desafios inesperados.

As texturas de ruído, como as da **Figura I**, desempenham um papel fundamental nesse processo, fornecendo a aleatoriedade e os detalhes cruciais para a criação de mundos virtuais vívidos e realistas. São como pincéis mágicos que, nas mãos habilidosas de um artista, transformam paisagens digitais em obras de arte.

Figura I - Textura base para geração Procedural



Fonte: Oliveira (2023).

Texturas de ruído são imagens sintetizadas através de algoritmos matemáticos que geram padrões aleatórios e granulares, distinguindo-se das texturas tradicionais, que são baseadas em imagens predefinidas. De acordo com a pesquisa de Arvaldo e Ignacio (2009), estas texturas são notáveis por sua infinita variabilidade e capacidade de personalização, adaptando-se a uma ampla gama de aplicações visuais. Entre os tipos mais relevantes de ruído, destaca-se o Ruído Perlin, uma abordagem clássica frequentemente empregada na simulação de terrenos, formações de nuvens e outros elementos naturais devido à sua capacidade de gerar padrões orgânicos complexos. O Ruído Voronoi, por sua vez, é apreciado pela criação de padrões de células irregulares que se assemelham a texturas de pedra, rachaduras em superfícies e mosaicos. Uma alternativa mais suave ao Ruído Perlin é o Ruído Simplex, que é especialmente útil para modelar terrenos com contornos mais suavizados e arredondados. O Ruído Gradiente é outra variação importante, produzindo transições suaves entre cores que são ideais para simular efeitos de iluminação e neblina. Por fim, o Ruído de Mármore imita a aparência de mármore e outras pedras, capturando a essência de seus padrões complexos e veios característicos. Essa diversidade de texturas de ruído permite uma ampla gama de aplicações, desde a geração de paisagens naturais até a criação de efeitos visuais detalhados em ambientes virtuais.

2.1.2.1 Geração de Caracteristicas Procedurais

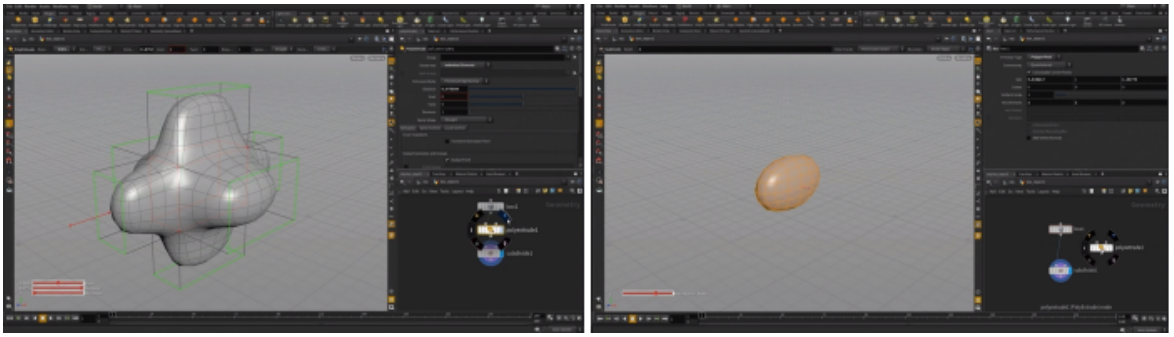
No estudo realizado por SHAN, Y (2021), a capacidade de gerar personagens 3D com características distintas de maneira procedural é destacada como um avanço significativo na criação de conteúdo digital. A implementação do sistema descrito por Shan permitiu a geração automática de uma ampla variedade de personagens, apesar de reconhecer que há espaço para aprimoramento na qualidade visual dos mesmos. Este sistema demonstrou ser particularmente eficiente na produção de um grande volume de personagens distintos, o que o torna uma ferramenta recomendada para projetos de desenvolvimento de jogos com ciclos curtos ou para aqueles que não requerem um alto padrão de qualidade visual. Por exemplo, jogos para dispositivos móveis, protótipos de jogos ou projetos desenvolvidos durante game jams poderiam se beneficiar enormemente deste sistema, aproveitando a sua capacidade de rápida geração de conteúdo para enriquecer o universo do jogo sem exigir extensos recursos de design gráfico.

Essa abordagem procedural para a criação de personagens não apenas otimiza o uso do tempo e dos recursos de desenvolvimento, mas também abre portas para a exploração de novas estéticas e narrativas. Em jogos onde a diversidade e a unicidade dos personagens podem contribuir significativamente para a experiência do jogador, o uso de sistemas como o proposto por Shan oferece uma maneira viável de injetar variedade e riqueza ao conteúdo do jogo. Isso é especialmente pertinente em contextos onde os desenvolvedores enfrentam limitações de tempo e orçamento, mas ainda desejam oferecer uma experiência rica e envolvente para os jogadores. Assim, a pesquisa de Shan sublinha a importância e o potencial dos métodos de geração procedural no campo do desenvolvimento de jogos, especialmente em projetos que visam um equilíbrio entre eficiência de produção e inovação no design de personagens.

2.1.2.2 Ferramentas para Geração de Caracteristicas

No trabalho de SHAN, Y (2021) aborda o uso de tecnologias avançadas na criação de personagens 3D de forma procedural, destacando ferramentas inovadoras que permitem aos desenvolvedores e artistas gerar personagens ricos e detalhados com eficiência e rapidez. Entre as tecnologias mencionadas, o Houdini se destaca por sua capacidade de oferecer um ambiente robusto de modelagem, animação e efeitos visuais, permitindo a criação complexa de personagens e ambientes com grande controle e precisão. Por outro lado, o Autodesk Character Generator e o Character Creator 3 são ferramentas que facilitam a criação de personagens detalhados, fornecendo uma vasta gama de opções de personalização que podem ser ajustadas para atender às necessidades específicas de um projeto, conforme podemos ver na Figura II.

Figura II - Interface do Houdini



Fonte: Shan Y (2021).

Além disso, SHAN enfatiza o potencial revolucionário da aplicação de

aprendizado de máquina na criação de personagens, especificamente através do Monster Mash, uma técnica que emprega redes neurais para aprender a gerar modelos 3D de rostos humanos a partir de um banco de imagens. Essa abordagem não apenas economiza tempo, ao reduzir a necessidade de modelagem detalhada por parte de um artista humano, mas também abre novos caminhos para a personalização e diversificação de personagens em jogos e mídias interativas.

A integração dessas tecnologias no processo de desenvolvimento de jogos promove uma nova era de criatividade e inovação, permitindo a criação de personagens e mundos cada vez mais complexos e imersivos. A capacidade de gerar rapidamente uma ampla variedade de personagens, cada um com suas próprias características únicas, não apenas otimiza os recursos de produção, mas também enriquece a experiência dos jogadores, oferecendo-lhes mundos dinâmicos e variados para explorar.

2.1.3 Renderização por requisição

Dentro do ecossistema da Unity, Segundo a própria documentação oficial da Unity Technologies (2023)[2], a funcionalidade de Renderização por Requisição se destaca como um recurso poderoso, especialmente no contexto de interações entre jogos Unity e sistemas baseados na web. Através da UnityWebRequest, a Unity oferece um sistema modular que permite a composição de solicitações HTTP e o manejo de respostas HTTP de maneira eficiente e versátil. Esse sistema foi projetado com o objetivo principal de facilitar a interação entre os jogos desenvolvidos na Unity e os back-ends de navegadores web, abrindo um leque de possibilidades para desenvolvedores. Uma das capacidades mais relevantes desse sistema é a sua integração direta com requisições web, o que permite funcionalidades avançadas como requisições HTTP fragmentadas, operações de POST/PUT em streaming, e controle total sobre cabeçalhos HTTP e verbos. Essa flexibilidade é crucial para o desenvolvimento de aplicações interativas que necessitam de comunicação constante com servidores web, seja para atualizar conteúdo, processar dados de jogadores ou mesmo interagir com outros sistemas online. Além disso, um aspecto particularmente interessante é a possibilidade de executar a aplicação Unity em linha de comando (CLI), expandindo ainda mais as fronteiras da automação e integração. Nesse contexto, a Unity pode responder a solicitações vindas da web, como a renderização de

imagens sob demanda. Por exemplo, uma aplicação pode estar configurada para aguardar uma requisição HTTP específica e, ao recebê-la, gerar uma imagem personalizada com base nos parâmetros fornecidos na requisição. Essa imagem pode então ser enviada de volta ao solicitante ou armazenada para acesso posterior.

Esse processo de Renderização por Requisição é particularmente útil em cenários onde a personalização e a geração de conteúdo em tempo real são essenciais. Aplicações que variam desde visualizações de produtos personalizáveis em lojas virtuais até jogos que oferecem conteúdo dinâmico baseado em eventos específicos do jogador podem se beneficiar dessa tecnologia. A capacidade de gerar conteúdo visual de forma dinâmica e sob demanda, diretamente a partir de requisições web, sem dúvida abre novas avenidas para criadores de conteúdo e desenvolvedores de jogos, permitindo-lhes criar experiências mais ricas e interativas para os usuários.

2.2 Processamento de Imagens

O Processamento de Imagens é um campo que se dedica à modificação e análise de imagens através do uso de computadores. Conforme explicado por Chavéz (2013), este campo envolve a alteração das informações contidas nas imagens sob diversos aspectos. O objetivo pode ser tanto a produção de uma nova imagem quanto a extração de informações específicas a partir da imagem original. Este processo é realizado através de um conjunto de técnicas que permitem capturar, representar e transformar imagens com o auxílio de tecnologias computacionais.

O emprego destas técnicas no processamento de imagens facilita a percepção humana e a interpretação automática, melhorando a qualidade visual das imagens e permitindo a extração e identificação de informações valiosas contidas nela. Essas técnicas são aplicadas em diversas áreas, incluindo a visão computacional e o reconhecimento de padrões, que são exemplos de análise de imagens. Dentro do Processamento de Imagens, pode-se distinguir entre três principais categorias com base no tipo de entrada e saída. A primeira é o próprio processamento de imagens, onde a entrada e a saída são ambas imagens. A segunda categoria é a síntese de imagens, na qual a entrada é uma representação tridimensional e a saída é uma imagem. Por último, a análise de imagens, que começa com uma imagem como entrada e produz como saída

informações sobre formas e outras características tridimensionais.

2.2.2 Convolução

Esta é uma operação matemática que combina duas funções para produzir uma terceira função. No contexto da pesquisa de Chávez (2013), processamento de imagens, a convolução é utilizada para aplicar um filtro sobre uma imagem. Isso é feito "convoluindo" a imagem com um kernel ou máscara, que é uma pequena matriz usada para aplicar efeitos como desfoque, nitidez, realce de bordas, entre outros. A convolução é uma ferramenta poderosa pois permite a aplicação de diversas operações de filtragem espacial de maneira eficiente e eficaz, sendo fundamental em muitas aplicações de processamento de imagens.

2.2.3 Texturização Procedural

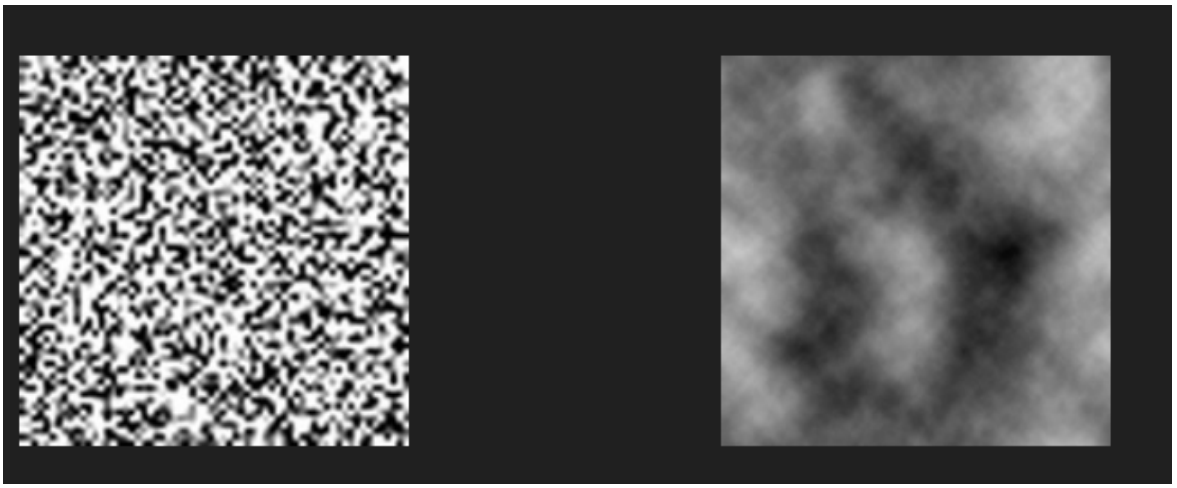
A texturização procedural é um método para gerar texturas de maneira algorítmica, em vez de usar imagens prontas. Essa técnica é amplamente utilizada em gráficos por computador, especialmente para criar paisagens, nuvens, superfícies de materiais e outros elementos visuais que requerem uma grande variedade de detalhes sem a necessidade de armazenar grandes quantidades de dados de textura. O trabalho de Íñigo Quílez em 2005 é um exemplo notável da aplicação da texturização procedural na geração de nuvens 2D que imitam o aspecto de nuvens volumétricas reais. Para criar essas camadas dinâmicas de nuvens 2D, o método começa com a definição de parâmetros pelo usuário, como a densidade das nuvens e a nitidez. Essas características são fundamentais para personalizar a aparência das nuvens e garantir que atendam aos requisitos específicos do projeto ou da cena. O cerne dessa técnica envolve o uso de ruído de Perlin em várias oitavas para construir uma função de movimento browniano fracionário (fbm) ou turbulência. O ruído de Perlin é uma função de ruído gradiente desenvolvida por Ken Perlin, que é especialmente útil para geração de texturas devido à sua aparência natural e à capacidade de ser facilmente ajustado para simular uma variedade de padrões naturais.

Quílez propõe uma abordagem econômica para a geração dinâmica de nuvens 2D, evitando a necessidade de gerar turbulência 3D em tempo real, o que poderia ser muito exigente em termos de recursos computacionais. Em vez disso, a ideia é pré-computar a turbulência em uma textura 3D e, então, selecionar a fatia correta com base no tempo. A interpolação linear entre duas fatias

adjacentes é realizada de forma eficiente com o uso de texturas 3D pelo hardware, aproveitando a interpolação trilinear.

Para ilustrar esse processo, Abaixo na **Figura III** é possível visualizar os resultados de Quilez.

Figura III - Antes e Depois de processo de Texturização Procedural



Fonte: Quílez (2005).

A inovação crucial descrita por Quílez é a decomposição da turbulência em suas oitavas constituintes e o armazenamento de cada uma em uma textura separada. Isso reduz significativamente a quantidade de armazenamento necessário, pois apenas as texturas de cada oitava são armazenadas, em vez de uma grande textura 3D. Além disso, ao mover cada oitava no plano 2D com diferentes velocidades e recompor a turbulência a cada quadro a partir dos ruídos de Perlin deslocados, é possível criar uma simulação convincente de nuvens dinâmicas.

2.3 Estatística e Cálculo Linear

2.3.1 Simulação Estatística

Na aplicação de simulações computacionais e análises estatísticas ao estudo de fenômenos complexos, como a dinâmica populacional de pragas de importância econômica, a estatística desempenha um papel crucial em modelar, analisar e interpretar os dados gerados. Esse tipo de abordagem permite que pesquisadores e profissionais compreendam melhor os padrões, tendências e potenciais impactos dessas pragas, contribuindo significativamente para o desenvolvimento de estratégias de manejo mais eficazes. No trabalho de Castro

e Oliveira (2010), por exemplo, a simulação computacional e a análise estatística são aplicadas ao estudo da dinâmica populacional do bicho-mineiro do cafeeiro e do ácaro rajado, pragas que representam sérias ameaças à produção de café devido aos danos significativos que podem causar às plantas. A utilização de simulações computacionais nesse contexto serve como uma ferramenta poderosa para criar ambientes simulados que refletem as condições reais do ecossistema do cafeeiro, permitindo a análise de diferentes cenários e o impacto de variáveis ambientais e de manejo na dinâmica populacional das pragas. A análise estatística, por sua vez, é aplicada para processar e analisar os dados gerados pelas simulações, identificando padrões, correlações e tendências nos dados. Isso envolve o uso de técnicas estatísticas, como análise de variância (ANOVA), regressão linear e análise multivariada, para entender as relações entre as variáveis e prever o comportamento futuro das populações de pragas. Essas técnicas permitem que os pesquisadores testem hipóteses específicas sobre os fatores que influenciam a dinâmica populacional das pragas e avaliem a eficácia de diferentes estratégias de controle.

A combinação de simulação computacional com análise estatística oferece uma abordagem robusta para o estudo de sistemas complexos, como a dinâmica populacional de pragas. Ao modelar a interação entre as pragas e seu ambiente, incluindo os efeitos de intervenções de manejo, os pesquisadores podem obter insights valiosos sobre como reduzir o impacto dessas pragas de maneira eficiente e sustentável. O trabalho de Castro e Oliveira ilustra bem como essas ferramentas podem ser aplicadas para fornecer uma compreensão profunda de questões críticas em agronomia e ecologia, facilitando a tomada de decisões baseada em evidências no manejo de pragas.

2.3.2 Regressão Linear

Segundo Chein (2019), a Regressão Linear é uma técnica estatística que visa modelar e investigar a relação entre duas ou mais variáveis. Esta técnica assume que existe uma relação linear entre a variável dependente (ou variável de interesse) e uma ou mais variáveis independentes (ou variáveis preditoras). A regressão linear pode ser simples, quando envolve apenas uma variável independente, ou múltipla, quando envolve duas ou mais variáveis independentes.

A Regressão Linear pode ser utilizada na estatística para diversos fins, incluindo previsão, inferência, e determinação de relações causais entre variáveis. Por exemplo, na área de economia, pode ser usada para prever o consumo das famílias com base na renda; em biologia, para modelar o crescimento de uma espécie em função de variáveis ambientais; ou em medicina, para avaliar o efeito de tratamentos em pacientes.

Através do modelo de regressão linear, é possível estimar os coeficientes que descrevem a magnitude e direção da relação entre as variáveis. Esses coeficientes são cruciais para entender como a variável dependente muda em resposta a alterações nas variáveis independentes. Além disso, a análise dos resíduos (diferença entre os valores observados e os valores previstos pelo modelo) permite avaliar a adequação do modelo, identificar possíveis outliers e verificar a presença de padrões não lineares não capturados pelo modelo linear.

2.3.3 Outliers

Segundo Ben-Gal (2005), outliers são observações que se desviam tanto das outras observações a ponto de levantar suspeitas sobre se elas foram geradas por um mecanismo diferente. Esses pontos de dados podem aparecer como anomalias ou valores atípicos e são significativamente diferentes do padrão de dados. A presença de outliers pode ser atribuída a uma variedade de causas, incluindo erros de medição, erros de entrada de dados, ou podem indicar variação experimental genuína ou novidade nos dados. Outliers são importantes em estatística e análise de dados porque podem levar a conclusões errôneas se não forem devidamente identificados e tratados. Eles afetam a média e outros tipos de estimativas estatísticas, podendo distorcer significativamente os resultados globais de uma análise. Por isso, a detecção de outliers é uma etapa crucial na pré-análise de dados, ajudando a garantir a precisão e a confiabilidade das conclusões extraídas.

2.4 Inteligencia Artificial

De acordo com Coppin (2010), Inteligência Artificial é um campo da ciência da computação dedicado a criar sistemas capazes de realizar tarefas que, até então, requeriam intervenção humana. A IA engloba uma gama de técnicas e metodologias que permitem às máquinas aprender, raciocinar, perceber, inferir, comunicar e tomar decisões de forma autônoma. Esta área

busca simular a capacidade cognitiva humana em máquinas, permitindo-lhes executar tarefas complexas, variando desde o reconhecimento de padrões até a solução de problemas de maneira inteligente.

2.4.1 Machine Learning

Coppin (2010), define Machine Learning como uma subárea dentro da Inteligência Artificial que se concentra no desenvolvimento de algoritmos capazes de aprender a partir de dados e melhorar seu desempenho ou fazer previsões com base em experiências passadas. O Machine Learning utiliza uma variedade de técnicas de estatística e análise de dados para permitir que os computadores 'aprendam' sem serem explicitamente programados para cada tarefa específica, o que significa que eles podem se adaptar e melhorar com a experiência.

2.4.2 Deep Learning

De acordo com Ferneda (2006), Deep Learning é uma técnica avançada de Machine Learning que envolve redes neurais com muitas camadas (daí o "deep"). Estas redes são capazes de aprender representações de dados em níveis de complexidade crescente. O Deep Learning é fundamental para o avanço de tarefas como o reconhecimento de fala, a visão computacional e o processamento de linguagem natural. Ele se destaca por sua capacidade de processar grandes volumes de dados e identificar padrões complexos que seriam difíceis ou impossíveis de discernir por humanos ou por técnicas de ML menos avançadas.

2.4.3 CNN

Ferneda (2018), define as Convolutional Neural Networks (CNN) como um tipo específico de rede neural profunda otimizada para o processamento de dados com uma grade topológica, como imagens. As CNNs utilizam uma operação matemática chamada convolução, que permite a análise eficaz de padrões visuais em pequenas regiões dos dados de entrada. Esta característica torna as CNNs particularmente úteis para tarefas de visão computacional, como reconhecimento de imagens, detecção de objetos e análise de vídeo.

2.4.3.1 MobileNet

Segundo Pinto et al. (2020), a MobileNet é uma arquitetura de rede neural desenvolvida com o foco na eficiência de computação e baixo consumo de recursos, tornando-a adequada para dispositivos móveis e sistemas embarcados.

Esta arquitetura utiliza operações convolucionais profundas e separáveis para reduzir o número de parâmetros e a complexidade computacional, sem comprometer significativamente a precisão. A MobileNet é amplamente utilizada em aplicações de visão computacional em dispositivos móveis, incluindo reconhecimento de objetos, classificação de imagens e detecção de faces, oferecendo um equilíbrio entre desempenho e eficiência.

2.5 Desenvolvimento de Software

O desenvolvimento de software é um campo dinâmico que abrange uma variedade de linguagens de programação, frameworks e ferramentas projetadas para enfrentar diferentes desafios e necessidades da indústria tecnológica. Dentre essas ferramentas, algumas se destacam pela sua eficiência, flexibilidade e comunidade ativa.

2.5.1 Python

Segundo a documentação oficial da Python Software Foundation (2020), Python é uma linguagem de programação de alto nível, interpretada, e de propósito geral. Caracteriza-se pela sua legibilidade e simplicidade, facilitando o desenvolvimento rápido de aplicações em diversas áreas, como desenvolvimento web, ciência de dados, automação, entre outros. Python suporta múltiplos paradigmas de programação, incluindo programação orientada a objetos, imperativa e, em menor medida, funcional.

2.5.1.2 TensorFlow

Segundo o TensorFlow Team (2023), desenvolvedores da blblioteca, ele é um software de código aberto para computação numérica que facilita a construção e o treinamento de redes neurais. É utilizado para desenvolver e treinar modelos de machine learning. O TensorFlow destaca-se pela sua flexibilidade e capacidade de operar tanto em CPUs quanto em GPUs, tornando possível o processamento de grandes conjuntos de dados e a implementação de complexos modelos de deep learning.

2.5.2 Dart

Segundo Alberto (2023), Dart é uma linguagem de programação otimizada para o desenvolvimento front-end para a web e para a criação de aplicações móveis de

alto desempenho. É desenvolvida pelo Google e caracteriza-se pela sua velocidade, concisão e escalabilidade, oferecendo uma sólida plataforma para o desenvolvimento de aplicações modernas.

2.5.2.1 Flutter

Segundo Alberto (2023), Flutter é um framework de UI open-source criado pelo Google, que utiliza a linguagem Dart. Permite o desenvolvimento de aplicações nativas para móveis, web e desktop a partir de uma única base de código. Flutter é conhecido pela sua rápida renderização, interface rica e componentes de UI personalizáveis, facilitando a criação de aplicativos bonitos e de alta performance.

2.5.3 GoLang

De acordo com Lima (2021), GoLang é uma linguagem de programação estaticamente tipada desenvolvida pela Google, conhecida por sua simplicidade, eficiência e suporte a concorrência. GoLang, ou simplesmente Go, é projetada para ser rápida, fácil de aprender, e capaz de lidar com sistemas escaláveis e complexos, tornando-se uma escolha popular para desenvolvimento de back-end e serviços em nuvem.

2.5.4 Comunicação

2.5.4.1 Protocolo HTTP

Segundo Ferreira (2023), o protocolo HTTP (Hypertext Transfer Protocol) é o fundamento da comunicação de dados na World Wide Web. Ele define um método para a troca de mensagens entre clientes e servidores, facilitando a transferência de documentos de hipertexto, como páginas web. HTTP é um protocolo sem estado, o que significa que cada pedido de um cliente é processado pelo servidor independentemente dos outros pedidos.

2.5.4.2 API

De acordo com Fernandes (2018), uma API (Application Programming Interface) é um conjunto de rotinas, protocolos e ferramentas para construir aplicações de software. APIs permitem a comunicação entre diferentes sistemas de software, facilitando a integração e a criação de soluções que podem se beneficiar de funcionalidades já existentes em outros serviços ou plataformas, promovendo

assim a interoperabilidade entre diferentes tecnologias e aplicações.

2.6. Tipos de Metodologias

De acordo com a pesquisa feita em "Vida Universitária (2020)", o panorama do desenvolvimento acadêmico e científico abrange uma diversidade de metodologias de pesquisa, cada uma com sua especificidade, objetivo e aplicação dentro do universo científico. Neste contexto, compreender as distintas abordagens torna-se fundamental para a construção de um conhecimento sólido e aplicável às diversas áreas do saber. A pesquisa exploratória emerge como o ponto de partida para muitos acadêmicos e cientistas, constituindo-se como uma ferramenta essencial quando se aventuram em campos pouco conhecidos ou quando buscam um entendimento mais profundo sobre um tema específico. Esta abordagem, marcada pela flexibilidade e abertura, permite uma exploração inicial que pode pavimentar o caminho para investigações mais detalhadas, servindo frequentemente como alicerce para o desenvolvimento de hipóteses e questionamentos mais precisos.

A pesquisa descritiva, conforme identificado na "Vida Universitária (2020)", assume um papel crucial ao oferecer um retrato detalhado das características de determinado fenômeno, população ou área de estudo. Esta metodologia se dedica a mapear e registrar aspectos específicos, padrões de comportamento ou as relações existentes entre variáveis, contribuindo significativamente para a sistematização do conhecimento sobre o tema em questão. Quando o objetivo se volta para a compreensão das causas e efeitos que regem determinados fenômenos, a pesquisa explicativa entra em cena.

Este tipo de pesquisa se aprofunda na análise das razões por trás das observações feitas, buscando estabelecer conexões causais que expliquem como e por que certos eventos ocorrem. Tal abordagem é indispensável para a formulação de teorias que possam ser posteriormente testadas e validadas. A pesquisa experimental, por sua vez, destaca-se por sua capacidade de testar hipóteses em condições controladas, manipulando variáveis para observar os efeitos gerados. Este rigoroso controle experimental permite uma análise precisa sobre as relações de causa e efeito, sendo uma ferramenta valiosa nas ciências naturais e aplicadas, onde a validação de teorias exige evidências empíricas concretas.

3. Metodologia

[1]Jáder Louis de Souza Gonçalves, Ciência da Computação, UNIR, jaderlouis@proton.me.

[2] N. Figueiredo Cavalcante Sales, Ciência da Computação, UNIR. nicolascavalcante0101@gmail.com.

[3] Wyllgner França de Amorim, Ciência da Computação, UNIR, wyllgner_franca@hotmail.com.

[3] Orientador: Dr. Lucas Marques de Cunha, UNIR, lucas.marques@unir.br

No início do projeto, a pesquisa exploratória emergiu como um componente vital na etapa preliminar, capacitando a equipe a alcançar uma compreensão profunda das tecnologias emergentes a serem utilizadas. Esta fase inicial foi marcada por uma investigação aprofundada sobre frameworks de ponta, variados tipos de Redes Neurais Convolucionais (CNNs), e a deliberação acerca da escolha da CNN mais adequada ao contexto do projeto, além de explorar técnicas avançadas de algoritmos. A natureza aberta e flexível dessa abordagem exploratória permitiu a identificação e seleção de ferramentas e metodologias inovadoras, que foram posteriormente adaptadas e integradas ao projeto, criando um fundamento robusto para o desenvolvimento subsequente.

Adicionalmente, a pesquisa aplicada desempenhou um papel essencial na superação dos desafios específicos do projeto, em especial os relacionados à busca por soluções eficazes que não implicassem em complicadas questões jurídicas. Através desta metodologia, a equipe pôde se concentrar em estratégias práticas e factíveis, assegurando que o projeto progredisse de forma ética e alinhada às normativas vigentes, mantendo ao mesmo tempo seu caráter inovador.

Um aspecto particularmente inovador integrado ao projeto foi a implementação de técnicas de geração procedural para o desenvolvimento de componentes e soluções dinâmicas. Comumente empregada em desenvolvimento de jogos e simulações, essa abordagem envolve a geração automática de dados, texturas, formas ou comportamentos por meio de algoritmos, eliminando a necessidade de configuração manual extensiva. A adoção da geração procedural possibilitou a exploração de soluções criativas e extremamente adaptáveis, otimizando os processos e produzindo resultados singulares e personalizados, adequados às exigências específicas do projeto. Este desafio foi acentuado pela novidade que a técnica representava para a equipe, especialmente no que diz respeito à transformação de texturas ruidosas

em imagens úteis para a renderização das texturas dos blobs.

A pesquisa bibliográfica também assumiu um papel crucial, funcionando como um alicerce para a construção do conhecimento teórico que embasou o projeto. Imersa em literatura acadêmica, artigos científicos e estudos de caso relevantes, a equipe conseguiu apoiar-se em descobertas anteriores, absorvendo perspectivas fundamentais e detectando lacunas de conhecimento que o projeto poderia explorar. Esta etapa não apenas enriqueceu a base teórica do projeto, mas também inspirou uma série de inovações e aperfeiçoamentos metodológicos.

Finalmente, a pesquisa experimental destacou-se pela sua contribuição significativa na análise do comportamento dos "Blobs" e na estruturação de um ambiente estatístico rigoroso. Manipulando variáveis de forma controlada e observando os efeitos resultantes diretamente, foi possível obter uma visão detalhada das dinâmicas presentes no ambiente Unity. A habilidade de testar hipóteses sob condições bem definidas provou ser crucial para a validação de teorias e o ajuste do projeto com base em evidências concretas, resultando na otimização de processos e no aprimoramento do desempenho global do sistema desenvolvido.

3.1 Metodologia Geral do Projeto

No desenvolvimento deste projeto, iniciamos com a construção de um ambiente de simulação no Unity, uma escolha estratégica pela capacidade da plataforma em criar ambientes interativos complexos. Este ambiente foi projetado para incorporar câmeras virtuais, que funcionam como sensores para capturar imagens. Esta etapa crucial, concluída em duas semanas, permitiu a simulação da captura de expressões faciais em um contexto controlado.

Posteriormente, dedicamos uma semana à criação de uma simulação estatística procedural avançada. Este processo visava estabelecer uma base robusta para a prototipagem e a experimentação dentro do ambiente simulado. O objetivo era assegurar uma representação fiel e adaptativa das variáveis envolvidas na análise de expressões faciais.

Para avaliar a eficácia e a estabilidade do ambiente criado, procedeu-se a uma série de testes de longo prazo. Estes testes, executados ao longo de cinco dias, simularam o funcionamento do ambiente por períodos extensos de 200, 300

e 1000 dias. Esta fase foi essencial para identificar e corrigir qualquer inconsistência ou falha no ambiente simulado.

Após a validação do ambiente de simulação, a fase seguinte focou no treinamento de um modelo de rede neural convolucional (CNN), especificamente o MobileNet. A escolha desse modelo deveu-se à sua eficiência em reconhecer expressões faciais e comportamentos, além de detectar elementos específicos no ambiente, como a presença do "blob". Este treinamento, que durou duas semanas, foi um passo decisivo para a capacitação do sistema em quantificar o engajamento por meio da análise de expressões faciais.

A análise quantitativa do engajamento foi aprimorada através da utilização da equação da reta, correlacionando a frequência de expressões detectadas com o nível de engajamento. Esta correlação foi ajustada por meio de análise de regressão linear ao longo de três dias, permitindo uma mensuração precisa do engajamento.

Na fase final, com duração de uma semana, desenvolvemos o aplicativo que integrava as funcionalidades projetadas, incluindo três APIs de comunicação. Esta etapa consolidou os dados coletados em uma base única, facilitando a análise e a interação com um ambiente procedural criado por uma das APIs.É importante salientar a importância do uso de datasets públicos para o treinamento e teste de algoritmos em ambientes reais, visando a precisão na identificação de expressões faciais. No entanto, neste projeto, optamos por criar nosso próprio ambiente simulado no Unity para demonstrar o potencial do projeto.

O projeto está disponível para consulta na organização do GitHub no seguinte endereço: https://github.com/visagetrack-project

3.2 Tecnologias Utilizadas

No desenvolvimento deste projeto, foi adotado um amplo espectro de linguagens de programação e tecnologias para atender às diversas necessidades e objetivos estabelecidos. C# emergiu como a linguagem principal, sendo utilizada tanto na criação do ambiente onde os blobs residem quanto na implementação da API terciária. Esta escolha se deve à sua robustez e

integração nativa com o ambiente Unity, facilitando o desenvolvimento de elementos interativos e a gestão de dados dentro do simulador.

Para as operações de backend e manipulação de dados, bem como para a construção da API primária, optou-se pelo uso de Golang. Esta linguagem ofereceu uma solução eficiente para o gerenciamento de serviços de backend, dada sua performance e facilidade de manutenção, essenciais para o processamento e transferência de dados entre as diferentes partes do sistema.

Python, em conjunto com TensorFlow e Jupyter Notebook, foi escolhido para a modelagem de Deep Learning, análise de dados e operações realizadas através da API secundária,. a flexibilidade de Python e as poderosas bibliotecas de TensorFlow se mostraram indispensáveis para o treinamento de modelos de inteligência artificial, análise de grande volume de dados e implementação de funcionalidades complexas de processamento de imagens.

Para o desenvolvimento da interface de usuário, foi utilizado Dart em conjunto com o framework Flutter, permitindo a criação de uma experiência de usuário consistente e cross-platform. Esta abordagem possibilitou a implementação de uma interface que pode ser facilmente adaptada para diferentes plataformas, garantindo uma interação fluida e acessível para os usuários em variados dispositivos.

Também foi trabalho por parte em C++, que embora tenha sido a linguagem menos utilizada no projeto, desempenhou um papel específico no processamento de dados em disco, como arquivos CSV, imagens e XMLs. Sua participação, ainda que limitada, foi crucial para a manipulação eficiente de dados estáticos e recursos dentro do projeto.

Adicionalmente, tecnologias secundárias foram incorporadas para atender requisitos de portabilidade e funcionalidade específicas. Swift foi utilizado para adaptar o projeto ao ecossistema iOS, garantindo que os usuários desta plataforma tivessem acesso pleno às funcionalidades do sistema. HTML foi empregado na criação de componentes da interface web, enquanto ShaderLab foi utilizado no processamento de imagens dentro do ambiente Unity, otimizando a renderização e a visualização de dados gráficos. Por fim, Matlab foi explorado na fase inicial para a elaboração de fórmulas estatísticas relacionadas ao comportamento dos blobs ao longo do tempo, embora tenha sido posteriormente substituído por soluções integradas mais eficientes.

3.2.1 Escolha da IA

Após uma extensa pesquisa bibliográfica e exploratória, a equipe identificou diversos modelos potenciais para o desenvolvimento da inteligência artificial (IA) do projeto. Dentre as opções consideradas, o modelo MobileNet, utilizando TensorFlow Lite, foi selecionado como a escolha ideal devido à sua eficiência em termos de recursos. A decisão de adotar o Modelo, é por conta de sua arquitetura de rede neural leve e otimizada para dispositivos móveis, alinha-se à necessidade de economizar recursos computacionais sem comprometer o desempenho da aplicação.

Além disso, a utilização do TensorFlow Lite facilita a implementação de modelos de IA em dispositivos com capacidades de hardware limitadas, permitindo a execução de inferências de forma rápida e eficaz. Essa escolha reflete uma abordagem pragmática para superar as limitações de hardware da equipe, garantindo que o desenvolvimento e a aplicação da IA não sejam prejudicados pela falta de recursos computacionais avançados.

4.1 Resultados Alcançados

Ao avaliar o projeto em seu estágio atual, pode-se constatar que foram alcançados resultados satisfatórios para o nível de aplicação pretendido, que é o de um protótipo de uma ideia de projeto com potencial para ser aplicada ao mundo real, transcendendo o ambiente simulado. Esses resultados são um testemunho do cuidadoso planejamento, da implementação rigorosa das várias metodologias de pesquisa, e da aplicação de tecnologias de inteligência artificial e simulação.

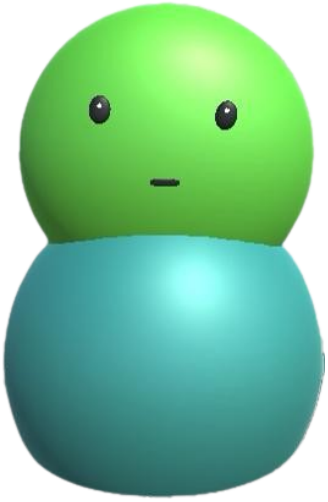
4.1.1 Ambiente Unity

A criação do ambiente Unity foi realizada a partir de materiais modelados em 3D pela própria equipe, onde se decidiu sobre o modelo dos seres e os nomes que eles iriam receber, sendo batizados de "blobs". Esses seres foram projetados para possuir suas próprias características distintas, cores e

comportamentos conforme suas reações. Foi estabelecido que os blobs teriam características faciais — como tamanho da boca, dos olhos e formatos — gerados proceduralmente, assegurando que eles seriam levemente diferentes uns dos outros.Além disso, foram definidos os seguintes comportamentos para os blobs: normal, triste, raiva, atento, dormindo, surpreso, feliz e a situação de ausência no dia. Cada comportamento corresponde a uma expressão facial e postura específica, refletindo o estado emocional e as reações dos blobs a diferentes situações dentro do ambiente de simulação. A modelagem e a animação desses comportamentos foram cuidadosamente elaboradas para oferecer uma representação visual intuitiva do estado interno de cada blob, enriquecendo a experiência de simulação com uma camada adicional de profundidade e realismo.

A diversidade de comportamentos e características faciais geradas proceduralmente permite uma ampla variação entre os blobs, garantindo que cada ser possui uma identidade única. Esse nível de detalhamento e personalização contribui significativamente para o realismo e a imersividade do ambiente Unity, proporcionando uma base sólida para a observação e análise do comportamento dos blobs em resposta às dinâmicas do ambiente simulado, segue um exemplo de blob conforme representado na figura IV.

Figura IV - Blob em estado Normal Renderizado

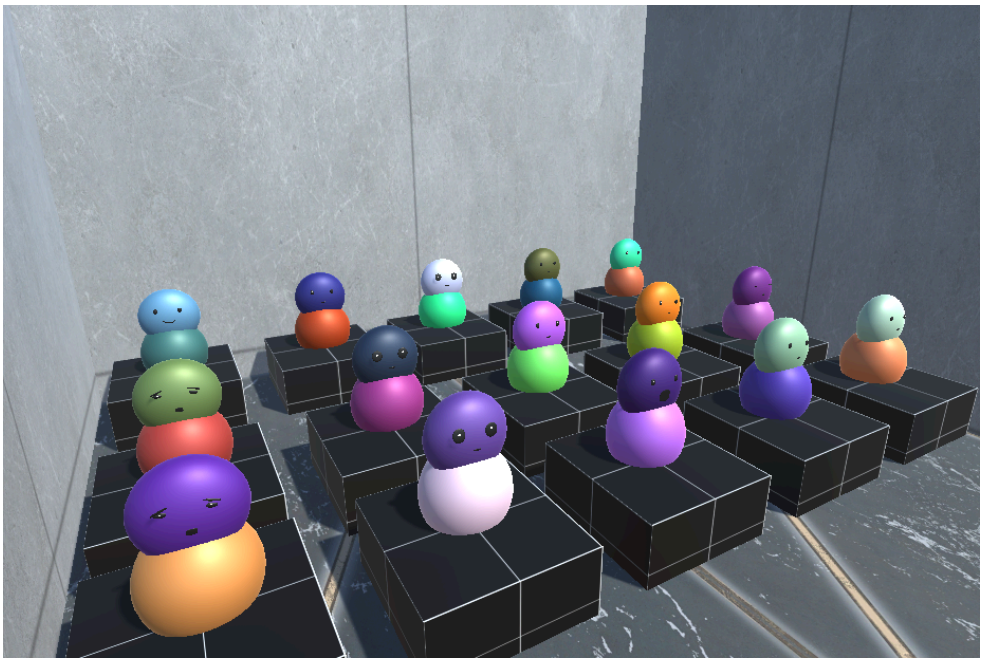


Fonte: Autoria Própria.

Foi estabelecido que a contagem de dias dentro da simulação duraria 3

segundos cada, permitindo que os comportamentos dos blobs mudassem ao longo dos dias. Essa dinâmica de tempo acelerado é crucial para observar as variações no comportamento dos blobs em um intervalo compacto, facilitando a análise de padrões e reações ao longo do tempo. Nesse contexto, a inteligência artificial desempenha um papel fundamental, pois é responsável por capturar imagens dos blobs em diferentes estados comportamentais para gerar o modelo de regressão linear, um fato importante sobre a questão do ambiente é que para fins de processamento na qual o aplicativo recebe vários usuários o ambiente não é renderizado o tempo todo, apenas quando é enviado uma requisição da IA para tirar uma foto do dia atual é onde o unity vai verificar os atributos e valores das texturas para assim realizar a criação da imagem e tirar a foto. a figura V retrata como o ambiente finalizado após renderização ficou.

Figura V - Ambiente Renderizado



Fonte: Autoria Própria

O Ambiente não renderizado é basicamente um JSON que contém todas as estruturas, cores tamanhos e coordenadas dos modelos gráficos e de textura de cada blob, e asset presente na imagem para o unity realizar a renderização.

4.1.2 Funcionamento da simulação estatística

Para a criação da simulação estatística, realizou-se uma ampla análise de fórmulas e métodos com o objetivo de criar um ambiente o mais próximo possível da realidade. Na primeira tentativa, era notável que os blobs possuíam tendência a estarem sempre mais atentos ou tristes, o que não era o ideal. Após muita reflexão pela equipe, decidiu-se por uma sequência de fórmulas matemáticas para atingir os valores esperados. Na primeira escala da fórmula, é definido, assim que o blob nasce, suas taxas de:

$$\begin{gathered}\text{Sejam } min = 0 \text{ e } max = 100. \\ \text{Para cada variável } x \in \{\text{laziness}, \text{commitment}, \text{interest}\}, \\ x = X_{n+1} = (aX_n + c) \mod m(min, max).\end{gathered}$$

Tendo como objetivo gerar valores aleatórios entre 0 e 100 para laziness (preguiça), commitment (comprometimento) e interest (interesse), onde a laziness representa a preguiça nativa do Blob, commitment indica o quão disposto ele está a manter o interesse e a não faltar no próximo dia. Em seguida, com os valores já estabelecidos, a taxa de laziness é diminuída a cada dia com base na seguinte fórmula:

$$\text{taxaPraDiminuirEmLaziness} = \frac{\text{commitment}}{2} + \left(\frac{\text{laziness}}{32 - \text{daysCount}}\right)$$

A cada dia, esta fórmula é utilizada para ajustar a preguiça do Blob, de modo que, mesmo que ele comece preguiçoso, se possuir um alto nível de comprometimento (um bom commitment), ele poderá superar essa preguiça através da força de vontade ao longo dos dias, apesar das dificuldades, tornando isso uma possibilidade viável.

Após essa etapa, é momento de analisar o commitment, que é a principal variável para definir o humor e comportamento do Blob. Se o commitment estiver baixo, o Blob estará triste; se estiver alto, estará atento. Essa fórmula opera com

dois estados possíveis. Se o commitment for maior que 45, a fórmula se comportará de maneira diferente; nesse caso, ela tentará diminuir ou aumentar menos conforme o estado. Isso assegura que um Blob não permaneça em um loop de aumento contínuo do commitment, pois o valor é reduzido à medida que aumenta e também é diretamente influenciado pela laziness, garantindo um equilíbrio para o Blob baseado unicamente no interesse atual que ele está sentindo. Assim, talvez ele ganhe commitment ou talvez ele perca. No entanto, é necessário ter dois estados possíveis para controlar o fluxo e evitar que o cálculo se transforme em uma acumulação excessiva, onde o Blob acabe com, por exemplo, 1 milhão de commitment ou -1 milhão de commitment. Essa fórmula serve como um mecanismo de balanceamento.

$$\text{commitment} > 45 =$$
$$x = \left(\frac{5}{1+\text{laziness}}\right) \cdot \left(\text{interest} \cdot \frac{\text{taxaPraDiminuirEmLaziness}}{100}\right), \text{estado} = 1$$

$$\text{commitment} < 45 =$$
$$x = \left(\frac{\text{laziness}}{10}\right) \cdot \left(\text{interest} \cdot \frac{\text{taxaPraDiminuirEmLaziness}}{1000}\right), \text{estado} = 0$$

Com essa abordagem, o commitment, que ditará o humor do Blob ao longo do dia, é atualizado e assegurado de estar sempre alinhado com a personalidade do Blob. Contudo, persiste um desafio em relação ao interesse, que pode condenar o Blob a uma situação desfavorável caso este seja muito baixo. Considerando que, em situações reais, o interesse de uma pessoa pode variar conforme o assunto, estabeleceu-se que a cada um número X de dias (definido aleatoriamente), uma fórmula é aplicada levando em conta a quantidade de dias transcorridos para definir um novo interesse.

Esta estratégia permite que o interesse do Blob seja dinâmico e reflita melhor a complexidade dos comportamentos e motivações humanas, garantindo que os Blobs não permaneçam presos em um estado de desinteresse constante. Ao invés disso, essa variabilidade do interesse possibilita que eles experimentem renovações periódicas em suas motivações, simulando a natural flutuação de interesses que ocorre em indivíduos reais. Essa metodologia não só enriquece a simulação, tornando-a mais realista, mas também introduz um elemento de

imprevisibilidade e complexidade na modelagem do comportamento dos Blobs. Garantindo, assim, um ambiente mais próximo da realidade e gerando também mais possibilidades imprevisíveis no ambiente, e, por último, para assegurar de fato que o ambiente possua outliers ou possíveis destaques, é realizada uma última variação dependendo da sorte.

A etapa final introduz um elemento de aleatoriedade adicional, refletindo as variações fortuitas que podem influenciar o comportamento e o desempenho dos indivíduos na vida real. Essa variação, baseada na sorte, simula eventos inesperados ou oportunidades únicas que podem alterar significativamente a trajetória de um Blob, para melhor ou para pior. Incorporando essa dimensão de incerteza, a simulação ganha em complexidade e realismo, aproximando-se ainda mais das nuances e imprevisibilidades da existência real. Além disso, a presença de outliers — aqueles Blobs que demonstram comportamentos ou resultados extremamente positivos ou negativos devido a fatores fortuitos — enriquece a dinâmica do ambiente simulado, oferecendo insights valiosos sobre como variações aleatórias podem impactar o desenvolvimento e a adaptação dos indivíduos em contextos diversos.

$$
\begin{aligned}
&\text{Dado } state, \text{ aplique as seguintes regras:}\\
&\text{Se } state = 2 \text{ ou } state = 1:\\
&\quad \text{Seja } randomChance = \text{GLC} \bmod 3\\
&\quad \text{Se } randomChance = 0 : laziness = laziness - 20\\
&\quad \text{Se } randomChance = 1 : blobInterest = blobInterest + 10\\
&\quad \text{Se } randomChance = 2 : \begin{cases} blobInterest = blobInterest - 30 \\ laziness = laziness + 1 \end{cases}\\
&\text{Se } state = 6:\\
&\quad \text{Seja } randomChance = \text{GLC} \bmod 3\\
&\quad \text{Se } randomChance = 2 : blobInterest = blobInterest + 50\\
&\text{Se } state = 7 \text{ ou } state = 4:\\
&\quad \text{Seja } randomChance = \text{GLC} \bmod 3\\
&\quad \text{Se } randomChance = 0 : laziness = laziness + 20\\
&\quad \text{Se } randomChance = 1 : blobInterest = blobInterest - 15\\
&\quad \text{Se } randomChance = 2 : \begin{cases} blobInterest = blobInterest + 30 \\ laziness = laziness + 1 \end{cases}
\end{aligned}
$$

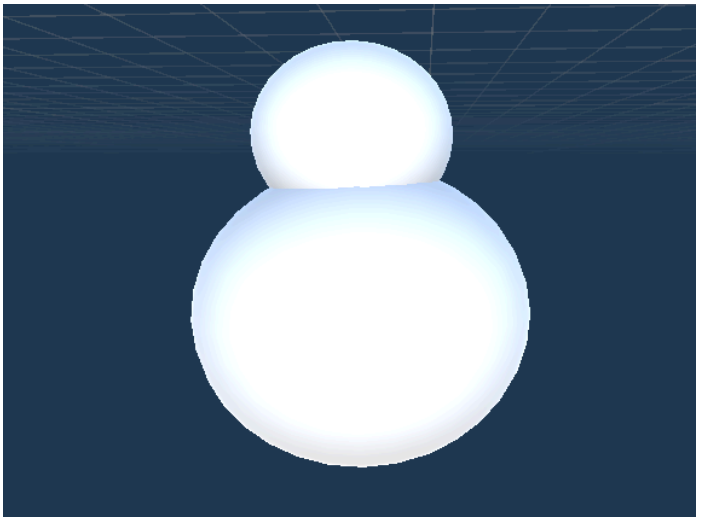
Neste cenário, tal tipo de situação é considerado uma anomalia no ambiente, apresentando uma probabilidade baixa de ocorrência (menor que 10%) para cada um desses casos. Ela provoca alterações diretas no comportamento do blob, o que pode simular o impacto de problemas pessoais que, de forma semelhante, podem ocorrer em ambientes humanos. A inclusão dessas

anomalias na simulação reconhece a existência e a influência de eventos raros ou extremos na vida dos indivíduos, refletindo como esses eventos podem desviar significativamente o comportamento ou o estado emocional de uma pessoa. Essas alterações diretas no comportamento do blob proporcionam uma camada adicional de realismo à simulação, destacando a importância de fatores externos ou internos inesperados na modelagem de sistemas dinâmicos e na compreensão da complexidade do comportamento individual.

4.1.3 Funcionamento da Geração Procedural

A partir do momento em que os blobs começaram a ser gerados e a se comportar de maneira satisfatória para a simulação, iniciou-se a implementação da geração procedural para os blobs. Nessa fase, embora cada blob possua exatamente a mesma forma básica, e tamanho de olhos, bocas e demais características faciais sejam uniformes, a técnica de geração procedural é aplicada para introduzir variações sutis e únicas em cada indivíduo. Isso é feito para garantir que, apesar de todos partirem de um "modelo pai", como ilustrado na figura V, eles apresentem características distintas que os tornem únicos dentro do ambiente de simulação.

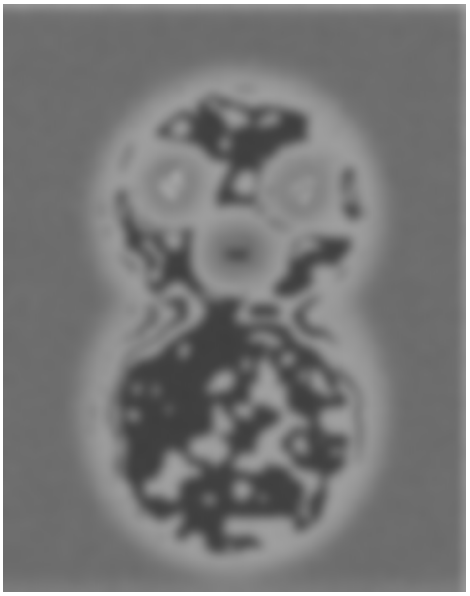
Figura VI - Modelo Pai



Fonte: Autoria Própria.

A partir deste modelo pai, são extraídas as características atuais, e, com base nisso, é gerada uma imagem de ruído base (Figura VII), esta imagem de ruído serve como um mapa de possibilidades, delineando as variações permitidas para cada característica dos blobs, como o tamanho dos olhos e da boca. O critério de que os tamanhos não podem variar mais de 10 pixels em relação ao modelo pai é um mecanismo de controle para garantir a consistência e a credibilidade visual dos blobs, mantendo-os reconhecíveis e alinhados ao design original.

Figura VII - Ruído de Imagem para Geração Procedural



Fonte: Autoria Própria

A imagem apresentada na Figura VII serve como uma representação didática, não refletindo com exatidão a imagem de ruído real empregada na geração procedural dos blobs no projeto. Para propósitos educacionais e de melhor compreensão, essa imagem foi submetida a um processo de upscale, partindo de suas dimensões originais de 64x128 pixels, além de ter sido aplicada

uma suavização. Essas modificações foram implementadas com o objetivo de facilitar a visualização e a interpretação da imagem por parte dos observadores, permitindo uma compreensão mais clara dos princípios subjacentes à geração procedural e como ela influencia a criação de características únicas para cada blob.

Após a criação da imagem de ruído inicial, é empregado um algoritmo de transformação que opera de maneira similar a um processo de convolução, com o objetivo de identificar áreas destinadas a características específicas e suas possibilidades dentro da imagem. Essa etapa é crucial para demarcar regiões que definirão as futuras características físicas dos blobs, como olhos, bocas e outras particularidades.

Uma vez completada a transformação, a imagem de ruído passa por uma metamorfose, resultando na imagem apresentada na Figura VIII. Esta representa o blob final, pronto para ser renderizado, indicando uma transição bem-sucedida das etapas conceituais de design para a materialização visual no ambiente Unity. A transição para a imagem de ruído final e a renderização subsequente são etapas cuidadosas, onde cada nova característica é meticulosamente verificada para garantir sua conformidade com os limites pré-estabelecidos de variação.

Figura VIII - Blob Pronto para ser Renderizado após algoritmo

Fonte: Autoria Propria

Caso uma característica - por exemplo, o tamanho de um olho ou de uma boca - esteja dentro do intervalo permitido de variação, ela é então integrada à imagem de ruído final. Este procedimento é exemplificado na Figura IX, onde é possível visualizar o resultado desse meticuloso processo de seleção e ajuste de

características, culminando na renderização do blob.

Este método de geração e verificação assegura que, embora cada blob exiba características físicas únicas, elas respeitam o padrão definido pelo modelo pai, garantindo assim uma uniformidade visual entre a população de blobs. Através deste processo, é possível manter a diversidade e individualidade de cada blob, ao mesmo tempo em que se preserva a coesão estética do conjunto, assegurando que a variabilidade entre os blobs enriqueça a simulação sem comprometer a integridade visual do projeto.

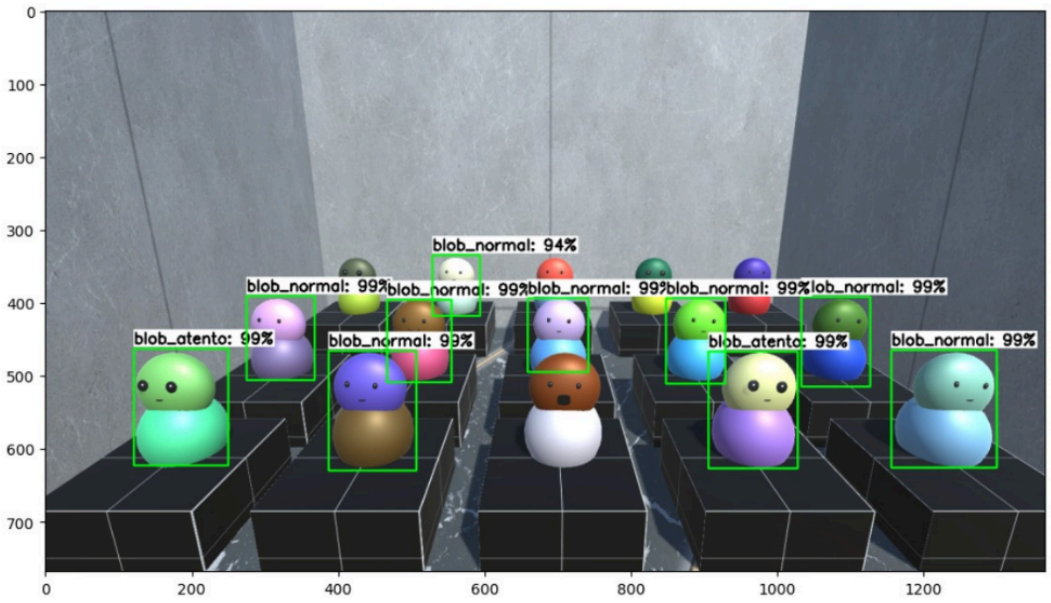
Figura IX - Blob procedural renderizado

Fonte: Autoria Própria.

4.1.4.1 Funcionamento da IA

A inteligência artificial (IA) no projeto foi projetada para funcionar tanto como um sistema de detecção de objetos quanto para a análise de expressões faciais. Seu principal objetivo é detectar a presença dos blobs dentro do ambiente simulado e, simultaneamente, identificar suas expressões faciais. Conforme visualizado na figura X.

Figura X - Funcionamento da IA



Fonte: Autoria própria

Utilizando A CNN MobileNet que possibilita esta abordagem bifuncional que realiza compreensão mais profunda e detalhada do comportamento dos blobs, captando não apenas sua localização, mas também seu estado emocional em momentos específicos. Ao detectar um blob, a IA analisa sua expressão facial, utilizando os modelos treinados para reconhecer as diferentes emoções e comportamentos previamente definidos, como felicidade, tristeza, raiva, surpresa, entre outros. Cada detecção e análise de expressão é cuidadosamente catalogada, com os dados sendo agregados a uma lista Esta lista contém informações cruciais sobre cada blob detectado, incluindo a identificação do blob, sua localização no ambiente, a expressão facial observada, e o timestamp do momento da detecção.

Vale ressaltar que a imagem utilizada durante o processo de desenvolvimento e treinamento da inteligência artificial (IA) não representa o funcionamento final do sistema. Eventuais erros de detecção e reconhecimento são meticulosamente corrigidos no modelo final, assegurando que a IA atinja um alto nível de precisão e eficiência na identificação e análise dos blobs. Este processo de aprimoramento contínuo é fundamental para garantir que a IA funcione conforme esperado, fornecendo resultados confiáveis e úteis para a simulação. Por questões de ética e privacidade, uma prática importante adotada no projeto é que, após a transformação das imagens capturadas em dados analíticos, essas imagens são prontamente excluídas do sistema. As imagens

são meramente ilustrativas para fins didáticos.

4.1.5 Funcionamento da Regressão Linear

A interação entre as diversas APIs e o modelo de inteligência artificial (IA) desempenha um papel fundamental na análise comportamental dos blobs. A segunda API tem a tarefa de enviar para o servidor web uma lista detalhada contendo os índices traduzidos de cada humor dos blobs, variando de 0 a 8, ou, em casos de ausência, marcando uma flag 8. Esses dados cruciais, originados da detecção precisa realizada pela IA, são coletados pela terceira API do ambiente Unity e encaminhados para o modelo analítico, onde são requisitados pela primeira API. O objetivo é transformar esses dados brutos em informações valiosas para o modelo de regressão linear, que sintetiza o engajamento diário dos blobs no ambiente virtual.

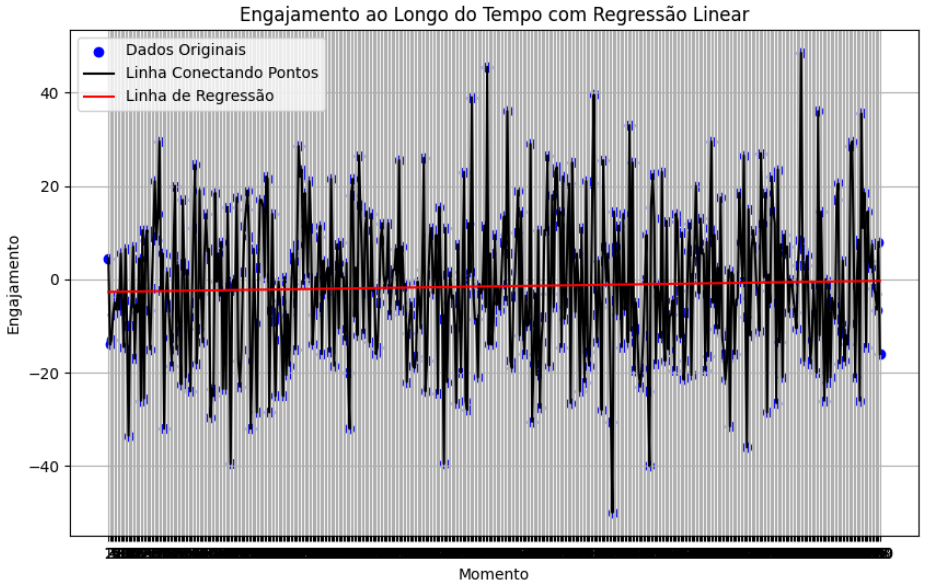
É feito um processo de transformação desses dados em um valor único de engajamento para cada dia. Inicialmente, é realizada a soma dos índices de humor de todos os blobs presentes, excluindo-se os ausentes. Essa soma é então convertida em um valor de engajamento que oscila entre -50 e +50, representando a dinâmica emocional coletiva dos blobs no respectivo dia. Cada ponto gerado nessa análise compõe a curva da regressão linear, representando visualmente o desempenho diário dentro do ambiente simulado.

A regressão linear é empregada com o propósito de avaliar o desempenho geral dos blobs, considerando as nuances de comportamento e variações emocionais manifestadas por suas expressões faciais. Essa metodologia de análise é atualizada periodicamente, a cada dez dias, proporcionando uma visão temporal aprofundada sobre o desenvolvimento e as transformações no comportamento dos blobs. Tal periodicidade é essencial para observar como interações específicas e eventos no ambiente influenciam os blobs ao longo do tempo.

Na Figura XI, é possível observar um exemplo concreto de uma regressão linear executada no dia 500 da simulação. Esse caso ilustra de maneira eficaz como os dados coletados e meticulosamente analisados ao longo do tempo são traduzidos em visualizações compreensíveis, oferecendo insights profundos sobre a conduta coletiva dos blobs. Através da aplicação da regressão linear, desenvolvedores e pesquisadores são capazes de identificar padrões

emergentes, estabelecer correlações significativas e explorar possíveis causalidades entre as variáveis observadas e o desempenho dos blobs. Essa abordagem não apenas facilita a compreensão dos princípios que regem o ambiente simulado, mas também destaca a aplicabilidade de técnicas avançadas de análise de dados na investigação de fenômenos comportamentais complexos.

Figura XI - Demonstração da Regressão Linear



Fonte: Autoria Própria

Através da regressão linear, os desenvolvedores e pesquisadores podem identificar padrões, correlações e potenciais causalidades entre as variáveis estudadas e o desempenho dos blobs, facilitando a compreensão dos mecanismos subjacentes que influenciam o ambiente simulado.

4.1.6 Funcionamento WEB e APIs

Este projeto multifacetado está estruturado em várias componentes principais: Web, Comunicação e Inteligência Artificial (IA), além do próprio Ambiente de Controle.

- Componente Web: A interface web foi desenvolvida utilizando Flutter, incorporando funcionalidades como sistema de autenticação e integração com o banco de dados Firebase, disponível no repositório "vt-web".

Comunicação:

- API Primária: Implementada em Go, a API primária ("vt-api") serve como o canal de comunicação direto com a interface web, desempenhando um papel crucial no funcionamento geral do projeto ao gerenciar requisições e facilitar a interação entre os componentes.

- API Secundária: Localizada no repositório "vt-model", esta API lida especificamente com o processamento de imagens. Utiliza o arquivo "api_communication.py" para capturar e transformar os dados antes de enviá-los à API primária. É importante notar que o "main.py" não aciona diretamente este arquivo; essa tarefa é realizada por uma API terciária, que automatiza a comunicação e a geração de gráficos de desempenho.

- API Terciária: Escrita em C# e mantida em um repositório privado, esta API tem a responsabilidade de gerar as imagens dos blobs e encaminhá-las para processamento pela IA. Além disso, recebe os dados processados de volta da API secundária. Uma de suas funções notáveis inclui a criação de novos ambientes Unity mediante requisições através do endpoint "/createNewAmbient{params1}/{params2}/{params3}", estabelecendo as condições necessárias para as operações das APIs subsequente.

Na Figura XII, é demonstrado o exemplo de uma interface dedicada ao usuário responsável pela gestão do ambiente virtual criado, evidenciando a funcionalidade e a praticidade do sistema. Esta interface é projetada para exibir informações atualizadas diariamente sobre os blobs, incluindo nomes, estados, humor e ocorrências de faltas, permitindo ao usuário um acompanhamento preciso e detalhado de cada entidade dentro do ambiente simulado. Uma das características mais notáveis desta interface é a capacidade de o usuário selecionar datas específicas para análise. Esta funcionalidade proporciona uma flexibilidade significativa, possibilitando a inspeção detalhada do desempenho dos blobs em qualquer dia escolhido. Tal recurso é especialmente valioso para avaliações de rendimento diário, possibilitando ao usuário identificar tendências, avaliar impactos de modificações no ambiente ou investigar eventos específicos

que influenciam o comportamento dos blobs.

A Figura XII apresenta a interface desenvolvida especificamente para facilitar a gestão do ambiente virtual pelos usuários, demonstrando a eficácia e funcionalidade do sistema projetado. Esta interface foi cuidadosamente elaborada para exibir um conjunto abrangente de informações sobre os blobs, atualizadas em um ciclo diário. Dados como nomes, estados emocionais, humor e registros de presença são disponibilizados, permitindo um monitoramento detalhado e contínuo de cada blob dentro do ambiente simulado.

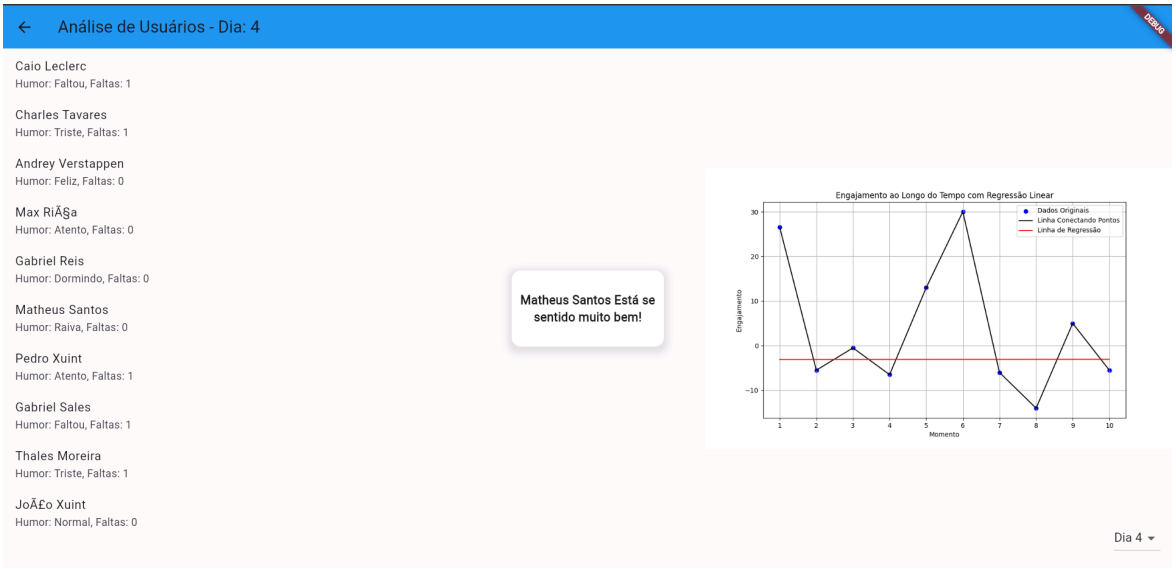
Uma funcionalidade destacada desta interface é a capacidade de seleção de datas específicas pelo usuário para análise detalhada. Esta opção oferece uma flexibilidade sem precedentes na exploração dos dados, habilitando o usuário a realizar uma inspeção minuciosa do comportamento e desempenho dos blobs em dias específicos. A relevância dessa característica não pode ser subestimada, pois facilita avaliações precisas de rendimento diário, assim como permite a identificação de padrões comportamentais, avaliação dos impactos decorrentes de alterações ambientais, e a investigação de eventos particulares que possam afetar significativamente o comportamento dos blobs.

Além disso, a interface é enriquecida com a atualização periódica de um gráfico de regressão linear, realizada a cada 10 dias, proporcionando uma análise longitudinal do desempenho dos blobs. Essa atualização gráfica é complementada por análises profundas geradas pela interação entre a segunda API e o modelo de inteligência artificial operando continuamente em segundo plano. A cada nova imagem processada, essa interação não apenas transforma os dados, mas também avalia as características e o bem-estar dos blobs. A análise considera se, com base nos registros dos últimos dias, os blobs têm experienciado períodos de maior ou menor produtividade, bem-estar ou dificuldades. Requisições contendo insights e atualizações sobre o estado dos blobs são enviadas a cada 5 dias, oferecendo uma perspectiva dinâmica e atualizada sobre o desempenho e as condições dos blobs dentro do ambiente simulado.

O gráfico de regressão linear, juntamente com as análises derivadas da interação da API com o modelo de IA, fornece uma visão compreensiva do desempenho geral dos blobs ao longo do tempo. Essa funcionalidade não

apenas facilita a identificação de tendências e padrões comportamentais, mas também possibilita a avaliação criteriosa da eficácia de intervenções realizadas no ambiente virtual. Além disso, contribui significativamente para a compreensão das dinâmicas complexas que regem o comportamento dos blobs, oferecendo insights valiosos para a otimização do ambiente simulado e a promoção de um entendimento mais profundo das interações sociais e emocionais virtuais.

Figura XII - Exemplo da Interface do Aplicativo



Fonte: Autoria Própria.

A integração dessas funcionalidades na interface do usuário reforça a usabilidade e a eficiência do sistema, permitindo uma gestão e um monitoramento eficazes do ambiente simulado. Facilitando a interação com os dados coletados, esta interface não apenas melhora a experiência do usuário, mas também enriquece as possibilidades de pesquisa e experimentação dentro do projeto.

5. Conclusão

O projeto de simulação dos blobs, desenvolvido no ambiente Unity e enriquecido com o uso inovador de inteligência artificial, geração procedural, e análises estatísticas avançadas, representa um marco significativo na exploração de sistemas dinâmicos e comportamentos simulados. Através da integração bem-sucedida de diversas tecnologias e metodologias de pesquisa, o projeto não

apenas atingiu seus objetivos iniciais, mas também abriu caminhos para futuras aplicações no mundo real, transcendendo suas origens como um protótipo de conceito.

Os resultados satisfatórios alcançados refletem a capacidade da equipe de superar desafios técnicos e teóricos, demonstrando a viabilidade de simular comportamentos complexos e interações sociais em um ambiente virtual. A precisão na modelagem das expressões faciais dos blobs, o realismo de suas interações, e a capacidade de análise comportamental por meio de regressão linear forneceram insights valiosos sobre os mecanismos subjacentes que governam o comportamento social e emocional, tanto em ambientes virtuais quanto, potencialmente, no mundo real.

A escolha estratégica de utilizar MobileNet com TensorFlow Lite para a criação do modelo de IA, juntamente com a eficácia das APIs de comunicação desenvolvidas, assegurou a eficiência e a sustentabilidade do projeto, mesmo diante de limitações de recursos. Além disso, a prática ética de converter imagens em dados e excluí-las subsequentemente reforça o compromisso da equipe com os princípios de responsabilidade e privacidade no desenvolvimento tecnológico.

Durante o projeto, uma série de lições valiosas foi aprendida no contexto de processamento de imagens, especialmente no que diz respeito à implementação e otimização de técnicas de inteligência artificial para reconhecimento e análise de expressões faciais em um ambiente simulado. Essas lições não apenas enriqueceram a compreensão técnica da equipe sobre o processamento de imagens, mas também proporcionaram insights sobre as melhores práticas para a criação de modelos de IA eficientes e responsáveis.

Durante o desenvolvimento do projeto, além das valiosas lições aprendidas no contexto de processamento de imagens, a equipe também adquiriu uma ampla gama de conhecimentos em desenvolvimento de software. A necessidade de utilizar diversas linguagens de programação e frameworks, incluindo Go, Flutter para desenvolvimento web, C# e C++ para o desenvolvimento no Unity, e Python com TensorFlow para o processamento de dados e inteligência artificial, proporcionou uma experiência de aprendizado rica e diversificada.

5. Referências

CHÁVEZ, G. **Fundamentos** . 2013. decom.ufo. Disponível em: <http://www.decom.ufop.br/guillermo/BCC326/slides/Processamento-de-Imagens-Fundamentos.pdf>. Acesso em: 29 jan. 2024.

Mateas, M., & Stern, A. **Procedural content generation techniques for games**. In Proceedings of the 2005 ACM SIGGRAPH symposium on Video games (pp. 247-254). ACM.: URL Mateas & Stern, 2005

Unity Technologies[1]. **UnityWebRequest**. Unity Manual. (2023). Disponível em: <https://docs.unity3d.com/Manual/UnityWebRequest.html/>. Acesso em: 8 mar. 2024. [1]

OLIVEIRA, R. **Complete Guide to Procedural Level Generation in Unity – Part 2**. (2023). Disponível em: <https://gamedevacademy.org/complete-guide-to-procedural-level-generation-in-unity-part-2/>. Acesso em: 8 mar. 2024.

SHAN, Y. **School of Arts Master's Programme in Game Design and Production A procedural character generation system**. [s.l: s.n.]. (2021). Disponível em: <https://aaltodoc.aalto.fi/server/api/core/bitstreams/9d5677f0-d8b8-49d0-9429-9a692310e12a/content>. Acesso em: 8 mar. 2024.

Unity Technologies[2]. (2023). **Creating Environments**. Unity Manual. (2023). Disponível em: <https://docs.unity3d.com/Manual/CreatingEnvironments.html/>. Acesso em: 8 mar. 2024. [2]

SANTOS, B.(2021) **O que é Unity?.** (2022)**.** Disponível em: <https://antlia.com.br/artigos/o-que-e-unity/>. Acesso em: 8 mar. 2024.

ARVALDO, N.; IGNACIO, L. **Texturas procedurales**. [s.l.] Universitat de València, 2009. Disponível em: <https://www.uv.es/mperezm/proyectos/curso_13_14/presentaciones/Neira_Alvar

ado__Luis_Ignacio_TexProceduralGPU.pdf>. Acesso em: 8 mar. 2024.

QUILEZ, I. **2D dynamic clouds - 2005**. (2005). Disponível em: <https://iquilezles.org/articles/dynclouds/>. Acesso em: 8 mar. 2024.

CASTRO, A.; DE OLIVEIRA, S. **SIMULAÇÃO COMPUTACIONAL E ANÁLISE ESTATÍSTICA APLICADAS AO ESTUDO DE DIFERENTES ASPECTOS DA DINÂMICA POPULACIONAL DE PRAGAS DE IMPORTÂNCIA ECONÔMICA -O BICHO-MINEIRO DO CAFEEIRO E O ÁCARO RAJADO**. (2010).. Disponível em: <http://repositorio.ufla.br/jspui/bitstream/1/3866/1/TESE_Simula%C3%A7%C3%A3o%20computacional%20e%20an%C3%A1lise%20estat%C3%ADstica%20aplicadas%20ao%20estudo%20de%20diferentes%20aspectos%20da%20din%C3%A2mica%20populacional%20de%20pragas%20....pdf>. Acesso em: 8 mar. 2024.

CHEIN, F. **Introdução aos modelos de regressão linear Metodologias COLEÇÃO**. ENAP, 2019. Disponível em: <https://repositorio.enap.gov.br/bitstream/1/4788/1/Livro_Regress%C3%A3o%20Linear.pdf>. Acesso em: 8 mar. 2024.

BEN-GAL, I. Outlier detection. In: MAIMON, O.; ROCKACH, L. (Ed.). Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers. 2nd. ed. [S.l.]: Kluwer Academic Publishers, 2005. cap. 1. Citado 2 vezes nas páginas 15 e 26.

COPPIN, B. **INTELIGENCIA ARTIFICIAL - 1ªED.(2010) - Ben Coppin - Livro**. [s.l.] LTC, 2010.

FERNEDA, E. **Redes neurais e sua aplicação em sistemas de recuperação de informação**. USP. 2006. Disponível em: <https://www.scielo.br/j/ci/a/SQ9myjZWLxnyXfstXMgCdcH/?format=pdf&lang=pt>. Acesso em: 8 mar. 2024.

RODRIGUES, D. **DEEP LEARNING E REDES NEURAIS CONVOLUCIONAIS: RECONHECIMENTO AUTOMÁTICO DE CARACTERES EM PLACAS DE LICENCIAMENTO AUTOMOTIVO**. João Pessoa - Paraíba: UFPA, 2018. Disponível em:

<https://repositorio.ufpb.br/jspui/bitstream/123456789/15606/1/DAR20052019.pdf>. Acesso em: 8 mar. 2024.

PINTO, G.; OSHIRO, H.; REIS, H.; DAL, R.; FILHO, M.; CLEMENTE, R.; DE SOUZA, T. **Application of MobileNet Convolutional Neural Network for Classification of Pediatric Images of Chest X-rays Aplicação da Rede Neural Convolucional MobileNet para a classificação de imagens pediátricas de raio-x de tórax**. Curitiba, Paraná: Conbepro, PPGEP, UTFPR, 2020. Disponível em: <https://aprepro.org.br/conbrepro/2020/anais/arquivos/09262020_180940_5f6fb12091cdd.pdf>. Acesso em: 9 mar. 2024.

Python Software Foundation. **Python 3.9.1 documentation**. 2020. Disponível em: https://docs.python.org/pt-br/3/tutorial/. Acesso em: 8 de março de 2024.

TensorFlow Team. **TensorFlow documentation**. TensorFlow. 2023. Disponível em: https://www.tensorflow.org/. Acesso em: 8 de março de 2024.

ALBERTO, M. **O que é Flutter? O Framework do Iniciante ao Avançado**. Alura. 2023. Disponível em: <https://www.alura.com.br/artigos/flutter#:~:text=O%20Flutter%20utiliza%20o%20Dart>. Acesso em: 9 mar. 2024.

LIMA, C. **O que é e como começar com Go (Golang)?.** TreinaWeb. 2021. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-e-como-comecar-com-golang>. Acesso em: 9 mar. 2024.

FERNANDES, A. **O que é API? Entenda de uma maneira simples**. Vertigo Tecnologia. 2018. Disponível em: <https://vertigo.com.br/o-que-e-api-entenda-de-uma-maneira-simples/>. Acesso em: 8 mar. 2024.

FERREIRA, G. **HTTP: desmistificando o protocolo por trás da Web**. Alura. 2023. Disponível em: <https://www.alura.com.br/artigos/desmistificando-o-protocolo-http-parte-1>. Acesso em: 8 mar. 2024.

VIDA UNIVERSITARIA. **Conheça os tipos de metodologia de pesquisa que você pode usar no seu TCC**. 2020. Disponível em: <https://www.universia.net/br/actualidad/vida-universitaria/conheca-os-tipos-metodologia-pesquisa-que-voce-pode-usar-seu-tcc-1166813.html>. Acesso em: 9 mar. 2024.