UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUCAS JAENISCH LOPES

AVALIAÇÃO ARTÍSTICA E TÉCNICA NO DESENVOLVIMENTO DE UM JOGO COM GERAÇÃO PROCEDURAL DE CONTEÚDO

CHAPECÓ
2021

LUCAS JAENISCH LOPES

AVALIAÇÃO ARTÍSTICA E TÉCNICA NO DESENVOLVIMENTO DE UM JOGO COM GERAÇÃO PROCEDURAL DE CONTEÚDO

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Prof. Dr. Fernando Bevilacqua

CHAPECÓ
2021

Lopes, Lucas Jaenisch

Avaliação Artística e Técnica no Desenvolvimento de um Jogo com Geração Procedural de Conteúdo / Lucas Jaenisch Lopes. – 2021.
55 f.: il.

Orientador: Prof. Dr. Fernando Bevilacqua.
Trabalho de conclusão de curso (graduação) – Universidade Federal da Fronteira Sul, curso de Ciência da Computação, Chapecó, SC, 2021.

1. Geração Procedural de Conteúdo. 2. Godot. 3. Arte. 4. Jogos. I. Bevilacqua, Prof. Dr. Fernando, orientador. II. Universidade Federal da Fronteira Sul. III. Título.

E-mail: lucasirmaododaniel@gmail.com

LUCAS JAENISCH LOPES

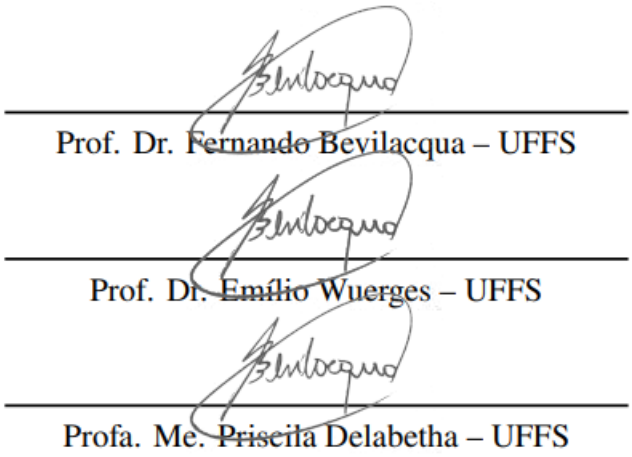
AVALIAÇÃO ARTÍSTICA E TÉCNICA NO DESENVOLVIMENTO DE UM JOGO COM GERAÇÃO PROCEDURAL DE CONTEÚDO

Trabalho de conclusão de curso apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Fernando Bevilacqua

Este trabalho de conclusão de curso foi defendido e aprovado pela banca avaliadora em: 20/05/2021.

BANCA AVALIADORA



Prof. Dr. Fernando Bevilacqua – UFFS

Prof. Dr. Emílio Wuerges – UFFS

Profa. Me. Priscila Delabetha – UFFS

RESUMO

A *Geração Procedural de Conteúdo* aplicada ao design de jogos eletrônicos permite a criação de produtos robustos e de maior qualidade, empodera desenvolvedores e proporciona melhor recepção de jogadores. No entanto essa tecnologia aplicada no desenvolvimento de jogos não possui apenas vantangens. Neste contexto o estudo buscou avaliar os benefícios e desafios no desenvolvimento de um jogo eletrônico com variação dos personagens de forma técnica e artística.

Palavras-chave: Geração Procedural de Conteúdo. Godot. Arte. Jogos.

ABSTRACT

Procedural Content Generation applied to game design allows the creation of robust and higher quality products, empowers developers and provides better reception from players. However this tecnology applied in game development does not only have advantages. In this context the study sought to evaluate the pros and cons of developing an electronic game with variations in the characters in a technical and artistic form.

Keywords: Procedural Content Generation. *Godot*. Games. Art.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

1.1 APRESENTAÇÃO

Jogadores procuram em jogos diversão, engajamento, e experiências únicas que muitas vezes extrapolam suas realidades. Essas experiências, por vezes, os mantém repetindo por inúmeras horas as mesmas atividades dentro de um jogo. A quantidade de tempo investida jogando e a frequência com a qual jogam depende de inúmeros fatores. Como, por exemplo, a quantidade de conteúdo comunicado pelo jogo. A Figura 1 mostra uma captura de tela do jogo desenvolvido.

Figura 1 – Captura de tela do jogo desenvolvido neste trabalho, com o jogador sendo perseguido por inimigos gerados de forma procedural.

Suponha-se um jogo de pintar casas onde se pode apenas utilizar um modelo de casa e um número limitado de cores: Os desfechos possíveis de casas a serem criadas rapidamente seriam alcançados pelo jogador devido ao pequeno alcance de possibilidades. Se existisse um maior leque de parâmetros, por exemplo, mais tipos de construções (casas, prédios, mansões) e fosse extendida a paleta de cores para pintar, as possibilidades de casas únicas a serem criadas são maiores em relação a versão anterior apresentada. Assim como as chances de um jogador ser estimulado a jogar mais vezes é aumentada, já que ele poderá ter maior chance de criar novas e únicas casas ao invés de repetir o padrões semelhantes que podem causar tédio.

Nesse contexto, jogos eletrônicos precisam da junção de diversos conteúdos para serem criados, sendo eles os desenhos e modelos digitais, animações, efeitos de som, trilha sonora,

códigos de programação, personagens, níveis, itens e mais. Para cada um desses conteúdos existirem é necessário investimento de capital financeiro e humano.

Quando se produz um jogo, o tempo e dinheiro presentes em um projeto podem ser suficentes para seu desenvolvimento. Entretanto é possível não existir recursos para pagar todos os gastos que lançar um produto completo como esse requer. É necessário investimento na sua distribuição, licenciamento, propaganda, testagem, e todas áreas além de seu funcionamento básico.

Para então reduzir tempo e dinheiro necessários na produção de um jogo eletrônico, assim como aumentar diversão e engajamento, que a Geração Procedural de Conteúdo (GPC) é utilizada. Ela permite que algoritmos criem novos conteúdos para jogos usando recursos pré existentes e faça modificações de acordo com o que os designers procuram, por exemplo, um modelo de personagem que possa ter inúmeras vestimentas (VLACHOS, 2010). Outro exemplo é o desenho de um recife que pode ser criado diferente do outro toda vez que necessário, evitando repetição (KORN et al., 2017) e mantendo um padrão de qualidade que não quebre os as diretrizes estéticas definidas.

Esse processo automatizado empodera os desenvolvedores, pois eles agora não precisarão iterar manualmente suas criações, o que exige tempo de trabalho, já que os algoritmos farão isso. Consequentemente há uma redução no tempo e no custo necessário para criar tais conteúdos, possibilitando o investimento nas outras áreas que envolvem o desenvolvimento de um jogo e que antes acabavam sem recurso.

A GPC é capaz de ir além, aumentando o limite da imaginação humana (AMATO, 2017) pois os algoritmos criam variações de personagens, níveis, itens, acabam por inspirar designers a desenvolverem novos jogos e melhorarem os já existentes. Um jogo que antes possuía apenas um número pequeno estático de níveis poderia se tornar chato e entendiante rapidamente. Através da utilização de GPC, podem-se gerar níveis novos e únicos em tempo real aumentando muitas vezes a rejogabilidade, diversão e engajamento de jogadores.

Aumentar a frequência com que um jogo é consumido através da variação do conteúdo dentro dele é de grande interesse para desenvolvedores (CONNOR; GREIG; KRUSE, 2017). Isso porque seu produto se torna um ambiente único de investigação e descoberta para os jogadores, que agora irão experimentar os limites dos jogos até onde os parâmetros que a GPC controla. Isso instiga e engaja jogadores a cada novo jogo. Essas experências de jogo únicas criadas pela GPC são os momentos em que jogadores poderão recordar e compartilhar com outros para comparar suas vivências.

Nesse contexto, o presente trabalho focou no desenvolvimento de um jogo com GPC aplicado em seus personagens. A questão da pesquisa contemplou elementos como: O que é necessário para começar geração procedural de conteúdo? Como projetar dentro de um jogo? Que limitações, impedimentos e dificuldades podem surgir? A maioria dos estudos apontam direções positivas para o uso e desenvolvimento de jogos utilizando GPC, mas nenhum dos quais foram reunidos para o estudo avaliaram GPC do ponto de vista técnica e artística.

2 OBJETIVOS

2.1 OBJETIVOS GERAIS

Avaliar o desenvolvimento de um jogo eletrônico com variação visual de personagens usando geração procedural de conteúdo do ponto de vista técnico e artístico.

2.2 OBJETIVOS ESPECÍFICOS

- Revisar na literatura conceitos de geração modular em jogos;
- Desenvolver um jogo onde o jogador e os inimigos tenham variação na aparência.
- Avaliação técnica e artística.

3 JUSTIFICATIVA

Um jogo eletrônico possui diversas qualidades que precisam ser atingidas para que este tenha uma chance de sobreviver em um mercado que é cada vez mais competitivo. Por exemplo, se um jogo é divertido, ele atrai jogadores e é aclamado nas mídias sociais. Ele tem para si grandes qualidades que o dão chance de se tornar um sucesso de mercado. No entanto, para isso se tornar possível, é necessário trabalho considerável e investimento de seus desenvolvedores afim de alcançar esse nível de qualidade em seu produto.

Para produzirem melhores jogos e alcançarem melhores resultados de vendas, desenvolvedores buscam por estudos que testem e comprovem a eficácia de novas técnicas e métodos de desenvolvimento. A análise empírica pode revelar as possibilidades de comportamento do mercado ao receber um produto com uma nova tecnologia nele embutida e assim descobrir se o investimento é vantajoso, ao invés de especular um resultado teórico.

A GPC ganha cada vez mais oportunidades de aparecer dentro dos jogos eletrônicos, sejam estes produzidos por desenvolvedores grandes ou indepedentes. Resultados positivos de estudos empíricos e a rentabilidade que a Geração Procedural de Conteúdo (GPC) proporciona aos desenvolvedores mais segurança em seu uso. Os grandes desenvolvedores, ou seja, as grandes empresas, geralmente formadas por inúmeros funcionários e investidores envolvidos nos negócios, possuem o financiamento necessário para desenvolvimento de jogos e tem maior interesse em aumentar a frequência com a qual produzem e lançam novos títulos para competir no mercado. Os desenvolvedores indepedentes, que costumam ser pequenos grupos de pessoas ou até um único indivíduo por trás do desenvolvimento completo de um jogo, fazem o máximo para criar um produto de qualidade com poucos recursos. Sendo assim, geralmente os recursos poupados em uma área são investidos na comercialização e aperfeiçamento do produto, além de pagarem suas contas.

Ambos ganham significativamente quando existem a possibilidade de direcionarem os investimentos a outras partes do desenvolvimento de seus jogos. Isso será refletido em menos falhas, maior propaganda, e qualidade de trabalho para funcionários. Consequentemente, serão criados jogos ainda mais robustos e completos.

O ambiente de desenvolvimento do jogo eletrônico para o estudo é de código livre *Godot*. O próprio trabalho resultou em recursos que desenvolvedores podem usar. Dentro desses elementos, estão códigos para conceber a arte técnica necessária que combine atlas de texturas e renderize personagens 2D complexos com variações de aparência, com suporte para animações, mecânicas de tiro, geração de personagens não jogaveis (PNJ's), inteligência artificial, assim como artes de personagens e cenários. Todos estes recursos desenvolvidos no jogo estão disponíveis online e, de forma indireta, são resultados do presente trabalho.

Este estudo colocou em foco o desenvolvimento de um jogo utilizando GPC. Com ele foram obtidos novos dados que podem ser utilizados para pesquisas futuras e além disso servir como caso de estudo para desenvolvedores quando se tratar de implementar variação de

personagens em jogos. A revisão bibliográfica conduzida para esse trabalho mostra que a GPC aplicada a cenários e níveis produz uma melhor recepção de jogadores aos jogos que utilizam de GPC. No entanto exige conhecimento técnico e artístico para representar e projetar elementos visuais que muitas vezes não é satisfeito por apenas um desenvolvedor.

A percepção dos jogadores a GPC com personagens é subjetiva sendo suas interpretações abertas a especulação e estudos empíricos que sustentem análises deste temas são bem-vindas e auxiliam na pesquisa na área de jogos digitais.

Jogos que implementam GPC podem se beneficiar ao longo de todo seu tempo de vida pois são capazes de gerar conteúdo novo até que seus jogadores cansem de jogar. Os mesmos sistemas que são criados para estes jogos podem ser reutilizados e ampliados para futuros trabalhos, sendo necessário um entendimento, conhecimento e também documentação de um sistema e suas regras para criação para que isso aconteça. Também é possível que tais elementos se tornem obsoletos devido a mudanças de escopo ou da direção de desenvolvimento, sendo necessário uma avaliação meticulosa sobre a necessidade da implementação de GPC no momento em que se projeta um jogo, afim de evitar trabalho que pode ser desperdiçado.

Por esses questionamentos que este estudo serve como um ponto de vista dos inúmeros entendimentos que são requeridos para avaliar um projeto com Geração Procedural de Conteúdo.

4 REVISÃO BIBLIOGRÁFICA

4.1 GODOT

A *Godot* é um motor de jogo de código aberto publicado no âmbito da licença MIT. Ela é desenvolvida pela comunidade do *Godot* Engine e usada em várias empresas da América Latina antes de ter se tornado código aberto e lançada para o público.

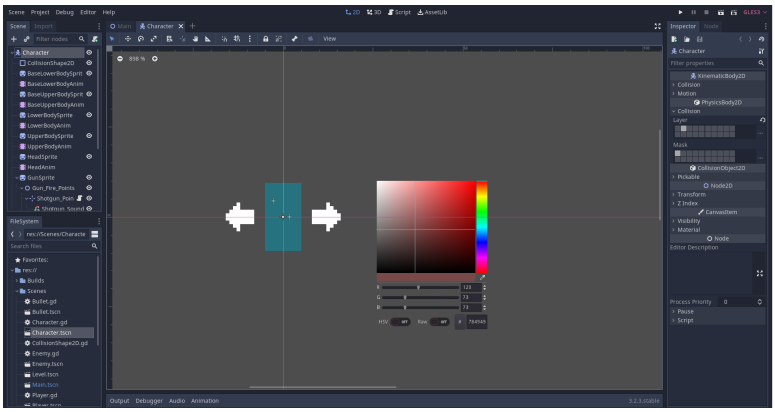


Figura 2 – Interface de desenvolvimento na *Godot*.

Godot fornece uma grande quantidade de ferramentas prontas, então é possível desenvolver um jogo sem reinventar a roda, agilizando o desenvolvimento.

4.2 RENDERIZAÇÃO DE FERIDAS

Vlachos (2010) mostra como infectados do jogo *Left 4 Dead 2 (2009)* são criados para serem visualmente diferentes um dos outros em comparação ao título anterior *Left 4 Dead (2008)*. Eles possuem uma resposta gráfica aos ferimentos que os jogadores causam aos mortos-vivos em combate, além de terem um ganho de performance. Os infectados possuem um modelo único masculino e feminino que são vestidos com inúmeros adereços. Assim como para mudar a aparência pessoas trocam de roupa, cortam o cabelo, o mesmo é feito para estes infectados. Variando-se uma quantidade limitada de adereços, texturas e modelos 3D que representem cortes de cabelo, roupas, pigmentações, é possível criar inúmeras variações entre estes infectados no jogo em tempo real. A Figura 3 mostra infectados gerados através destas variações.

No contexto do jogo *Left 4 Dead 2*, jogadores passam a maior parte do tempo atirando nos infectados. A melhoria de mostrar uma variação gráfica maior do impacto que os jogadores tem em combate cria uma experiência mais imersiva. Isso porque jogadores percebem que o modo de enfrentar os infectados gera ferimentos em regiões específicas do modelo de acordo com o armamento usado. A Figura 4 mostra infectados após combate e a variação alcançada.

Figura 3 – Infectados após servem vestidos pelo algoritmo. Figura reproduzida de Vlachos (2010).

Figura 4 – Demonstração da variação nos ferimentos em infectados. Figura reproduzida de Vlachos (2010).

4.3 GPC POSSIBILITANDO NOVAS EXPERIÊNCIAS DE JOGO

Smith et al. (2011) mostra o jogo *Rathenn* que utiliza significativamente de GPC para gerar novas experiências jogáveis. Sendo um jogo plataforma 2D, *Rathenn* cria conteúdo de acordo como o jogador percorre o jogo, sendo coletando moedas, derrotando inimigos, evitando obstáculos ou não. Toda essa informação é usada para direcionar a geração das próximas partes do jogo permitindo que o jogador faça seu próprio caminho conforme joga. A Figura 5 mostra uma captura de tela do jogo sendo executado, ilustrando um dos cenários vivenciados pelo jogador.

O estudo foca nos parâmetros que *Rathenn* usa como pilares para GPC do jogo. O espaço é dedicado a mostrar a criação de jogos que seguem o desenvolvimento de elementos procedurais. Esse é, também, um dos norteadores da pesquisa do presente trabalho.

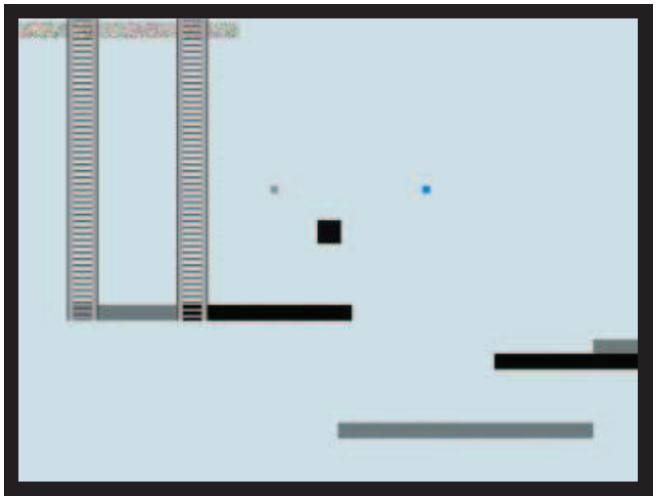


Figura 5 – Captura de tela do jogo *Rathenn* (2011) que foi utilizado no estudo. Figura reproduzida de Smith et al. (2011).

4.4 AVALIANDO E DEFININDO A EXPERIÊNCIA DE IMERSÃO EM JOGOS

Jennett et al. (2008) mostra como é possível construir e avaliar a experiência de imersão de jogadores em jogos usando questionários no formato de escalas de Likert. A Figura 6 contém um questionário usado em um dos jogos testados.

I did not feel any emotional attachment to the game.
SD D N A SA

I was interested in seeing how the game's events would progress.
SD D N A SA

It did not interest me to know what would happen next in the game.
SD D N A SA

I was in suspense about whether I would win or lose the game.
SD D N A SA

Figura 6 – Exemplo de Questionário com Escala Likert. Figura reproduzida de Jennett et al. (2008).

A utilização de questionários é uma das formas mais comuns de obtenção de informações sobre a interação de jogadores com jogos digitais. Embora elas sejam respondidas de forma subjetiva, os resultados fornecem um panorama da percepção de jogadores dos mais variados elementos do jogo.

4.5 CRIATIVIDADE COMPUTACIONAL EM GERAÇÃO PROCEDURAL DE CONTEÚDO

Barreto, Cardoso e Roque (2014) diz que Criatividade Computacional (CC) é uma área de pesquisa que, entre outras investigações, pesquisa modelos criativos aplicados a produção de artefatos. Apresentando como GPC poderia se beneficiar de modelos criativos, conclui

que é interessante introduzir CC à GPC para aprimorar a experiência do jogador. Quando se introduz criatividade computacional ao conteúdo gerado proceduralmente, jogos podem melhorar sua experiência geral, além de que GPC pode ser usada para empoderar designers, como ferramentas de iniciativa mista. Aponta-se como seria interessante explorar como um paradigma de iniciativa mista pode estimular a criatividade de um designer. De fato, dando-se as ferramentas e habilidade de engajar em comportamentos criativos por meio de modelos criativos subjacentes, o trablho explorou o quanto isso pode estimular e prover para produção de artefatos criativos.

4.6 PROCEDURAL ZELDA: UMA GPC DE AMBIENTE PARA PESQUISA DE EXPERIÊNCIA DE JOGADOR

Heijne e Bakkes (2017) apresentam a maneira como a recriação de *The Legend of Zelda: A Link to the Past* com o design procedural pode ser utilizado em jogos na geração de fases, combate e quebra-cabeças procedurais aos gráficos e jogabilidade de *A Link to the Past* em *Procedural Zelda*. A Figura 7 representa a geração de fases no ambiente.

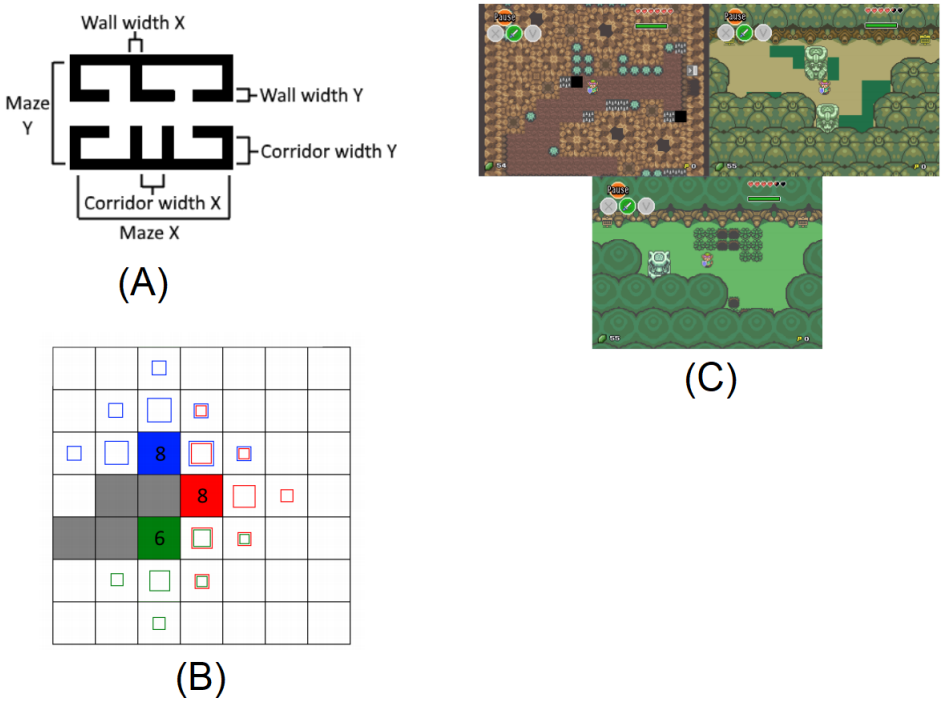


Figura 7 – Demonstração da geração procedural do mapa em Procedural Zelda. (A) mostra os parâmetros de geração. (B) visualização em forma de tabuleiro dos elementos do mapa. (C) o desenho final do mapa que o jogador percorre. Figura reproduzida de Heijne e Bakkes (2017).

O estudo não possui os resultados do teste com a experiência dos jogadores em *Procedural Zelda*, mas serve como exemplo de quão amplos podem ser os conteúdos e parâmetros que variam dentro de um jogo. Por exemplo, há variação de gráficos dentro de uma fase, dos inimigos que a

populam ou dos quebra-cabeças que na fase estão. Assim uma narrativa diferente é gerada para cada vez que uma nova campanha no jogo é iniciada.

4.7 GERAÇÃO PROCEDURAL DE CONTEÚDO NA INDÚSTRIA DE JOGOS

Amato (2017), no livro Procedural content generation in the game industry (2017) (Geração de Conteúdo Procedural na Indústria de Jogos), contribui para um apanhado histórico do uso de GPC em jogos. Um dos exemplos é o jogo *Diablo (1996)* no qual as áreas do mapa e masmorras tem seu desenho aleatório gerado em tempo de jogo utilizando de gráficos pré construídos na perspectiva isométrica. Nesse caso, inimigos que são derrotados derrubam armamentos com elementos aleatórios atribuídos. A Figura 8 mostra uma masmorra do jogo em questão.

Figura 8 – Captura de tela do jogo *Diablo (1996)*. Figura reproduzida de Amato (2017).

Desde 1996 as ferramentas comerciais já faziam uso de GPC, tornando-as estabelecidas, mesmo que ainda confinadas a jogos narrativos com interpretação de personagem como RPG's. Iniciava-se o nascimento da era moderna da GPC com o foco em reduzir ainda mais custos e tempo necessários para criação de conteúdo para jogos.

4.8 TRABALHOS RELACIONADOS

4.8.1 Geração Procedural de Conteúdo de Adereços para Jogos: Um Estudo nos Efeitos da Experiência de Usuário

Korn et al. (2017) utilizam de um jogo de exploração com barco por corais para testar a experiência de usuário. O jogo possui duas versões em uma os corais são criados de forma procedural, e em outra, manual. A Figura 9 mostra uma captura de tela do jogo, mostrando todos os elementos visuais compondo a atmosfera do jogo. A Figura 10 destaca a comparação de um coral feito de forma manual com outro procedural.

Figura 9 – Captura da tela do jogo, com barco e corais, utilizado no teste de avaliação de jogaderes no estudo. Figura reproduzida de Korn et al. (2017).

Jogadores que participaram do teste experimentaram as duas versões do jogo de forma aleatória em sessões de 10-15 minutos. Após cada sessão de jogo eram feitas perguntas usando-se uma escala de Likert de 5 pontos. Usuário pontuaram de 1 a 5 palavras chaves do tipo "realista", "bonito". Os resultados obtidos mostraram que os corais gerados de forma procedural alcançaram maior aceitação e percepção de qualidade em relação aos corais gerados manualmente pelos jogadores.

4.8.2 Avaliando como o GameFeel é Influenciado Pela Aceleração e Desaceleração do Avatar do Jogador

Dahl e Kraus (2015) avalia diferentes percepções de jogadores dentro de um jogo plataforma 2D onde se chuta uma bola. No jogo, as variáveis de aceleração e desaceleração são

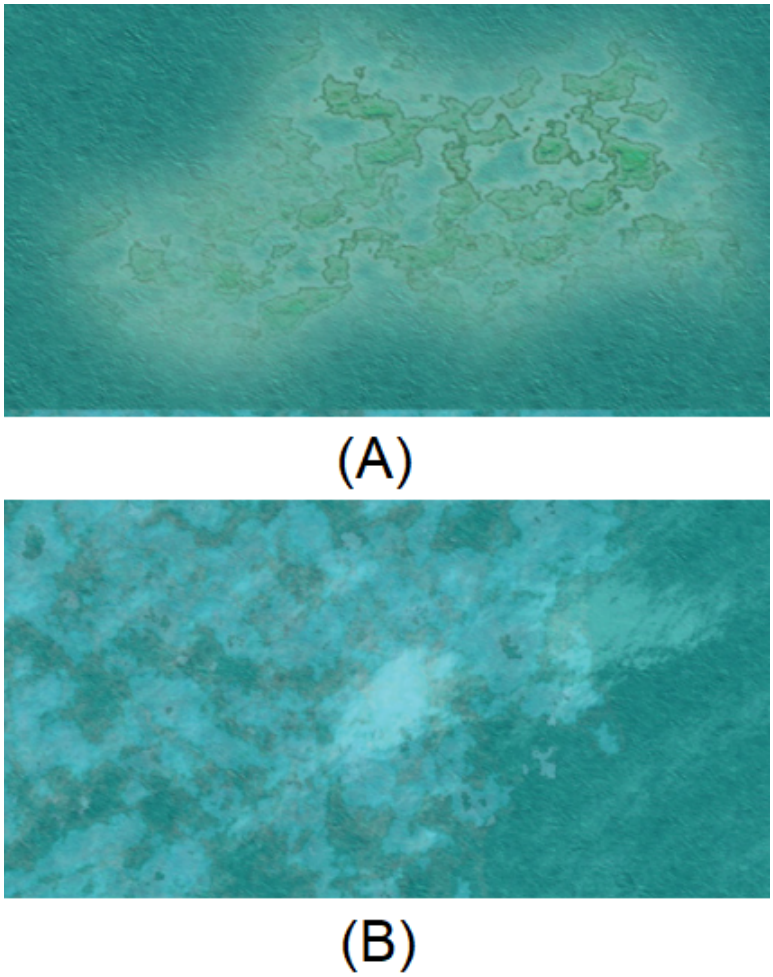


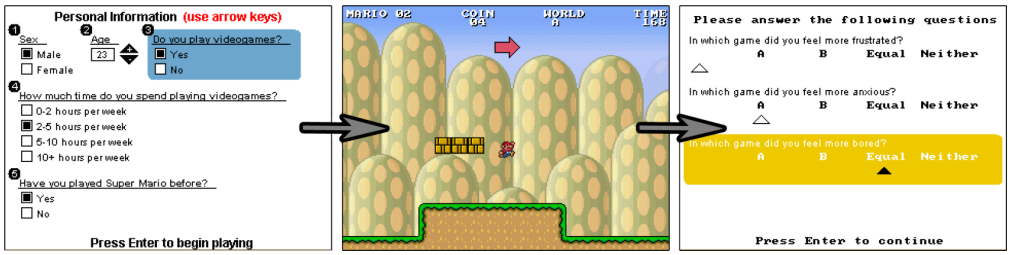
Figura 10 – Comparação entre Coral criado por um artista e o outro procedural. (A) criado por artista. (B) criado proceduramelmente. Figura reproduzida de Korn et al. (2017).

alteradas por fase ao longo de 3 fases.

O estudo contempla uma coleta de dados separada em 3 etapas, conforme ilustrado na Figura 11:

- O primeiro questionário começa antes do jogo, sendo um questionário demográfico para perguntar nome, sexo, idade, localização.
- O segundo questionário possui duas partes e ocorre na metade do jogo.
  - Na primeira parte, participantes utilizam as próprias palavras para descrever o jogo.
  - Na segunda parte, participantes recebem um leque de 5 palavras pré definidas e avaliam com uma escala Likert de 7 pontos . Esse leque pode possuir palavras chaves como "Realista", "Entediante"e dessa forma variar entre o alcance de 1 á 7.
- O terceiro questionário consiste nas respostas de quão difícil, frustrante, responsivo foi de controlar a bola assim como o quanto gostaram. Caso esquecessem, foi disponibilizado um botão para voltar a controlar a bola.

Após as quatro rodadas de questionários, os participantes eram submetidos a um formulário online perguntando quais parâmetros eram alterados a cada round. Além disso, ele podiam descrever como é a percepção do jogo em geral, assim como a percepção de outros 6 jogos de plataforma.



(A) (B) (C)

Figura 11 – Demonstração do fluxo de captação dos dados utilizado no estudo. (A) captura de dados demográficos de um participante. (B) o jogo é testado pelo participante. (C) É feito o questionário de experiência. Figura reproduzida de Dahl e Kraus (2015).

4.8.3 Avaliando o Impacto da Geração de Conteúdo Procedural na Imersão de Jogos

Connor, Greig e Kruse (2017) comparam duas versões do mesmo jogo onde em um as fases são criadas por um designer humano enquanto no outro elas são criadas proceduralmente. Em um jogo de tiro com perspectiva cima-baixo, o jogador deve navegar para progredir ao próximo nível. A Figura 12 mostra uma captura de tela do jogo.

Figura 12 – Captura de tela do jogo de tiro com cenário gerado proceduralmente utilizado no estudo. Figura reproduzida de Connor, Greig e Kruse (2017).

O conceito de imersão é usado para medir a qualidade, usando uma escala de Likert de 10 pontos. Foram testados 20 participantes, sendo que cada um jogou apenas uma versão

do jogo. Com a dificuldade de alguns para navegar pelo jogo, foram feitos alguns cortes onde ficaram 9 jogadores com a versão GPC e 7 com a versão humana.

4.8.4 Agentes de Computador contra Avatares: Respostas a Personagens de Jogo Interativo com Controlados por Computador ou outro Jogador

Lim e Reeves (2010) testam duas hipóteses e três perguntas de pesquisa utilizando da Análise de Variância (ANOVA) para avaliar a importância de um ou mais fatores. Junto da medição fisiológica de jogadores, por exemplo, Batimentos Cardíacos, Pressão Sanguínea, Resposta Galvânica da Pele que acontece durante o tempo de jogo, fez-se um teste comportamental baseado no Manequim de Auto Avaliação ao fim da sessão. A Figura 13 demonstra o teste em questão, que é outra ferramenta comum para medição dos níveis de agitação e valência de jogadores.

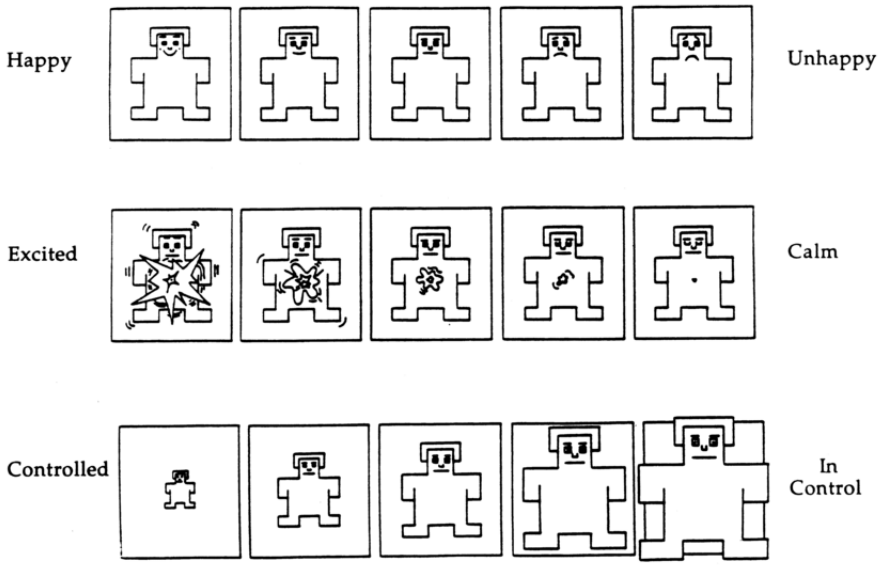


Figura 13 – Exemplo de teste com Manequim de Auto Avaliação.

Foram coletados os dados fisiológicos transmitidos pelo corpo do jogador durante a sessão e sua experiência emocional com o teste de Auto Avaliação. Juntos e combinados na Análise de Variância, analisa-se como estes fatores influenciam nos gráficos gerados. A Figura 14 contém as comparações de interação do jogador com outro jogador e com computador.

O que ganha destaque é a captura dos dados fisiológicos de jogadores para análises tão subjetivas quanto os níveis de interação deste grupo com avatares controlados por outros ou com uma máquina.

4.8.5 Modelando Experiência de Jogador no Super Mario Bros

Yannakakis e Togelius (2011) utilizam o jogo *Infinite Mario Bros* para medir a experiência do jogador. Nesse contexto, há um controle dos parâmetros dentro do jogo, como o número de

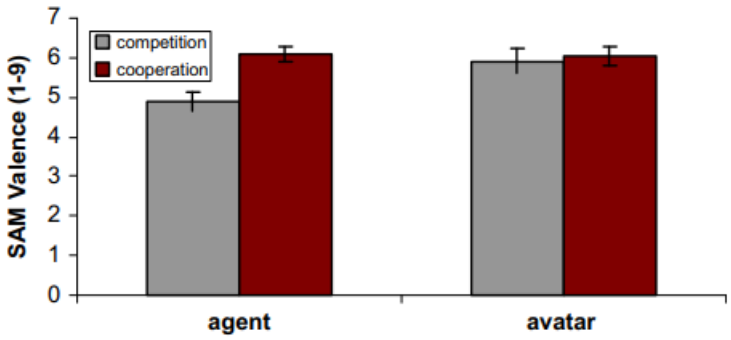


Figura 14 – Gráfico do desempenho de interação entre Computador (cinza) e Jogador (Vermelho) gerados com respostas de um teste com manequim de auto avaliação 13. Figura reproduzida de Lim e Reeves (2010).

objetos que constroem a fase, obstáculos, plataformas, buracos e também as ações do jogador como o número de pulos, mortes, itens coletados. Aplicando-se um questionário ao fim das sessões, constroem-se gráficos que mostram a diversão, dificuldade e frustração do jogador. Os resultados são mostrados na Figura 15.

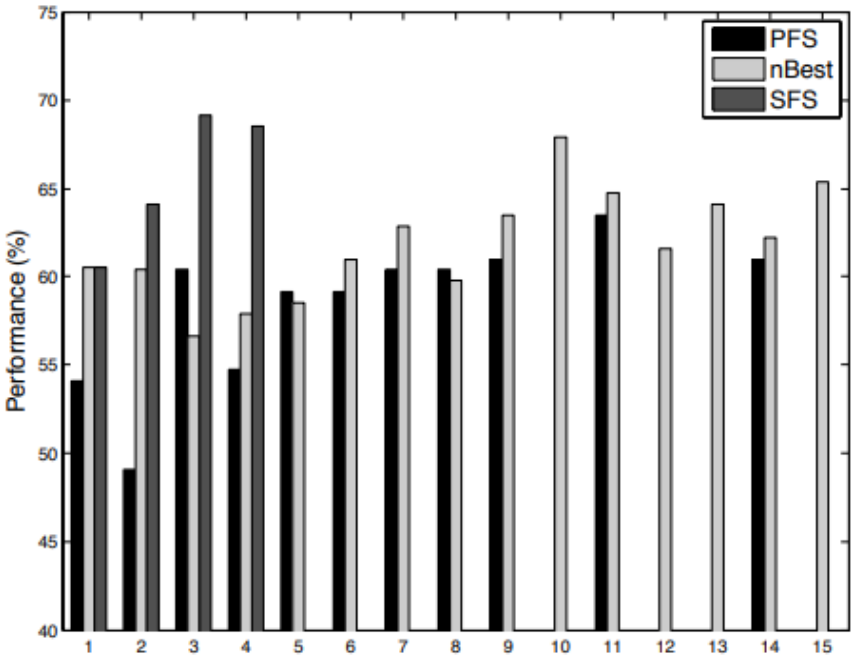


Figura 15 – Gráfico gerado da diversão do jogador após computado parâmetros. Figura reproduzida de Yannakakis e Togelius (2011).

5 IMPLEMENTAÇÃO

O presente trabalho descreve os passos para construção de um jogo no gênero shooter (jogo de tiro), utilizando Geração Procedural de Conteúdo no motor de jogos Godot. O maior foco deste trabalho foi em gerar avatares (entidades que representam visualmente o jogador e inimigos) dentro do jogo e que sejam compostos por elementos estéticos personalizáveis. Foi necessário também que a geração procedural de avatares tivesse capacidade de gerar não apenas uma figura estática de uma única pose de animação, mas também um sistema capaz de representar o avatar em inúmeras poses, segurando diferentes tipos de armas. Adicionalmente, que houvesse possibilidade de, no futuro, aumentar o leque de animações e conteúdo de personalização. Este capítulo descreve o processo de implementação bem como os desafios e abordagens de cada etapa na construção do sistema de criação procedural de conteúdo do jogo.

Em um jogo do estilo pixel art, está presente a simplicidade para representar qualquer objeto nesse nível baixo de fidelidade gráfica sejam personagens, objetos ou ambiente. Estes são criados rapidamente comparados a um estilo que exige maior qualidade e fidelidade gráfica como são alguns jogos 3D. Portanto quando este estilo retrô de jogos é aplicado, o trabalho para que uma animação e efeitos em geral sejam interpretados é menor. Isso porque o nível de detalhe necessário para representar os elementos em tela com os pixels é menor, deixando grande parte da interpretação do que está acontecendo na tela para a imaginação de quem está vendo. Um dos objetivos desse trabalho é usar desse baixo nível de fidelidade gráfica para acelerar a criação de conteúdo, como as roupas e elementos de aparência dos avatares para que o sistema os organize em populações temáticas e gere estes inimigos de cada tema com chances pré determinadas de aparecer.

A primeira etapa consistiu em criar o conceito artístico que representasse personagens capazes de segurarem armas dentro de um estilo pixel art no padrão de 32x32 pixels. Este é um tamanho clássico de texturas e sprites de jogos antigos como os dos sistemas *Super Nintendo* ou *Sega Genesis* que utilizavam esse padrão com o poder de processamento gráfico da época. Também para desenhar um personagem que possua proporções realistas do corpo, alguns elementos devem ficar de fora nesse nível baixo de resolução, como detalhes de rosto e mãos. Isso acontece pois não existe espaço digital em tela que represente estes elementos. Sendo estes os requesitos e limitações artísticas, originou-se o primeiro avatar do jogo. A Figura 16 mostra o rascunho do avatar.

Foi necessário projetar como o sistema conseguiria gerar avatares aleatórios animáveis usando um número fixo de conteúdo pré gerado de roupas e elementos de aparência. A modularização do avatar é a chave para atingir tal objetivo. Para modularizar o avatar, é necessário dividir este em partes, como se fosse recortado em peças de um quebra cabeça, que juntas formam uma imagem só. No caso do projeto, as partes junstas representam o avatar do jogador ou inimigos.

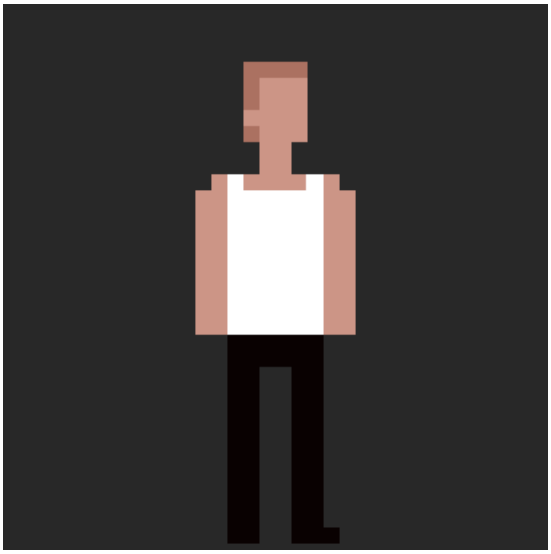


Figura 16 – Rascunho do primeiro avatar criado para o jogo

5.1 MODULARIZAÇÃO DO AVATAR

5.1.1 Separação das Partes do Avatar

O avatar desmontado de um grande bloco de 1024 pixels para um formato quadricular de 32 pixels de altura por 32 pixels de largura permite que sejam feitos recortes em outras imagens. Isso possibilita que essas sejam agrupadas posteriormente em uma imagem só do avatar a partir dos elementos que influenciaram o número de partes que foram recortadas. Um avatar pode ter listados o nível de detalhe, quantas armas e ferramentas este avatar utilizará, quantas telas de animação serão usadas para simular movimento e vida do avatar, tempo, trabalho, e nível de personalização.

Assim que os pesos de cada um desses elementos foram colocados sobre o projeto, dividiu-se em 6 partes principais um avatar dentro do jogo. O personagem foi composto pela combinação das imagens da parte de baixo do corpo (da cintura até os pés), de cima do corpo (da cintura até a cabeça), cabeça sendo acessórios faciais, cabelos, máscaras, roupas da parte de cima do corpo da cintura até a cabeça, roupas da parte de baixo do corpo da cintura até os pés e um equipamento. A Figura 17 mostra o avatar separado de forma modular nessas 6 partes que este é composto.

Após esta convenção ser criada, pode-se combinar cada uma dessas camadas de imagens uma sobre a outra respeitando-se a seguinte ordem dos 6 elementos que compõe o avatar: primeiro a parte de cima do corpo, parte de baixo do corpo, acessório da cabeça, acessório da parte de cima do corpo, acessório da parte de baixo do corpo, e arma. Dependendo do jeito que um motor de jogos desenha texturas, esta sobreposição pode não funcionar, ou ter um resultado não desejado. Por exemplo, o corpo pode ficar na frente da arma. A Figura 18 mostra o avatar combinado das 6 partes modulares que o compõe. É importante também deixar claro que cada

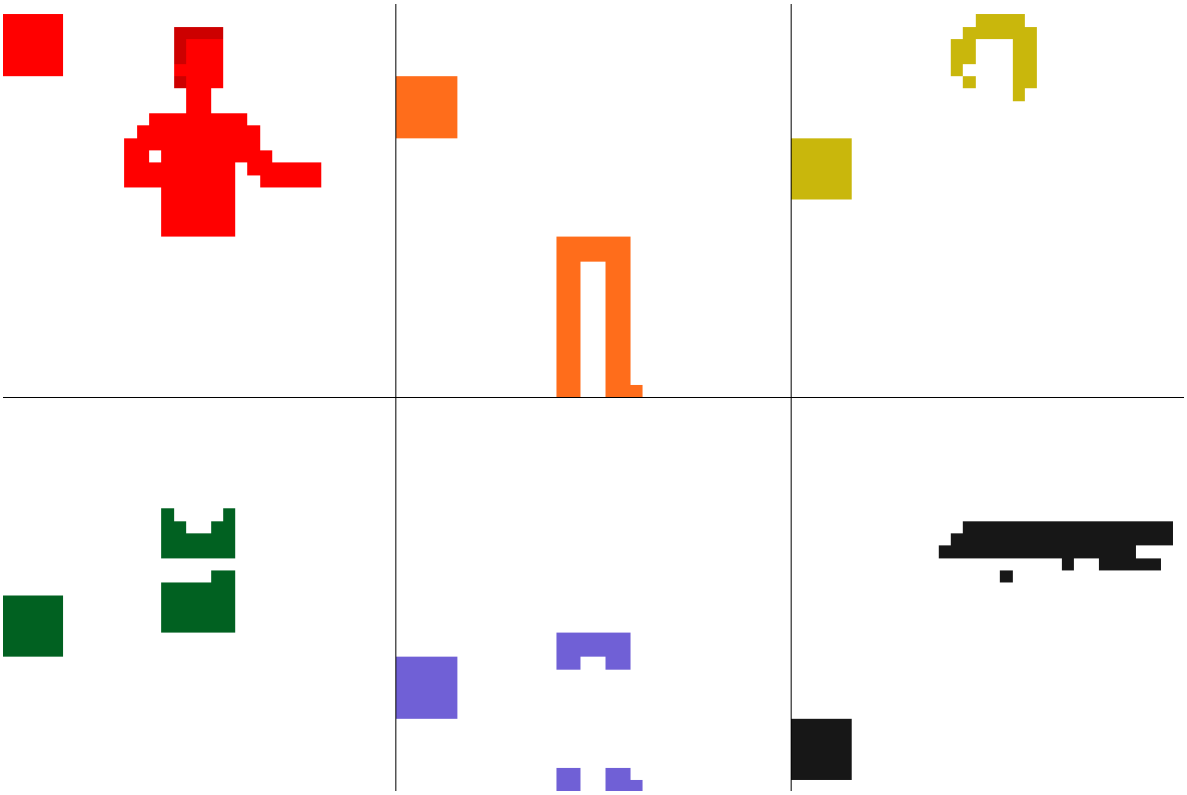


Figura 17 – Avatar do jogo separado em 6 partes com cada parte tendo uma cor correspondente para cada uma. (VERMELHO) parte de cima do corpo. (LARANJA) parte de baixo do corpo. (AMARELO) acessório da cabeça. (VERDE) acessório da parte de cima do corpo. (AZUL) acessório da parte de baixo do corpo. (PRETO) a arma.

acessório criado para personalizar avatares deve ter em sua textura apenas o componente que ele deve representar, junto de transparência da imagem. Essa é um componente comum em imagens do formato *png* para que as texturas dessa forma também não sobreponham as outras.

5.1.2 Combinação de Diferentes Elementos do Avatar

Com o avatar carregado de forma modular, pode-se trocar os elementos que o compõem da forma desejada. Para isso, trocam-se as texturas utilizadas em cada uma das 6 partes fazer um avatar com outra aparência. A Figura 19 mostra o mesmo avatar, porém combinado com outra variação de texturas para acessório da cabeça, acessório da parte de cima e acessório da parte de baixo.

Restou determinar a organização dos componentes que criam avatares dentro do projeto. Isso foi realizado com uma separação em pastas que correspondem a cada uma das 6 partes criadas para formar o avatar. Guardou-se todos os conteúdos e acessórios de cada parte em sua respectiva pasta, cabelos e chapéus no acessório da cabeça, por exemplo. Deixou-se para o sistema buscar nestas os arquivos de textura que são possíveis de carregar para o motor de jogo montar um avatar.

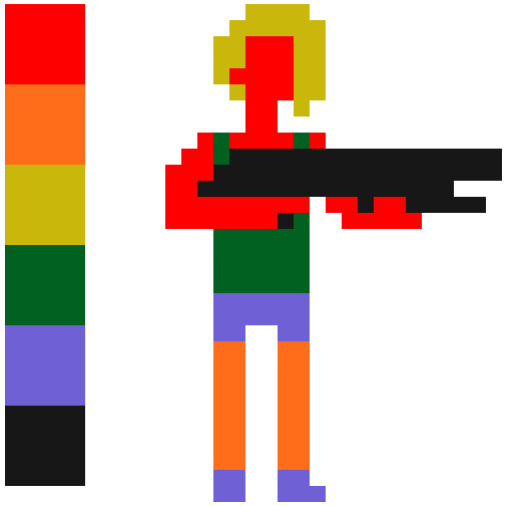


Figura 18 – Avatar do jogo combinado em 6 partes com cada parte tendo uma cor correspondente para cada uma. (VERMELHO) parte de cima do corpo. (LARANJA) parte de baixo do corpo. (AMARELO) acessório da cabeça. (VERDE) acessório da parte de cima do corpo. (AZUL) acessório da parte de baixo do corpo. (PRETO) a arma.

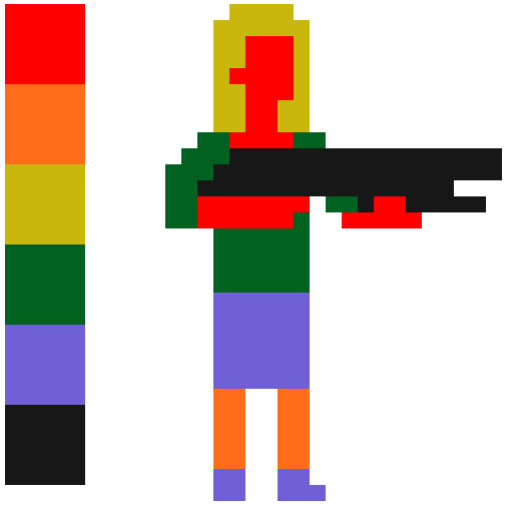


Figura 19 – Variação do vatar do jogo combinado em 6 partes tendo uma cor correspondente para cada uma. (VERMELHO) parte de cima do corpo. (LARANJA) parte de baixo do corpo. (AMARELO) acessório da cabeça. (VERDE) acessório da parte de cima do corpo. (AZUL) acessório da parte de baixo do corpo. (PRETO) a arma.

5.1.3 Modificação de Cores do Avatar

A variação do sistema é capaz de criar avatares diferentes tendo apenas que escolher texturas para cada parte do corpo. É de interesse aumentar esse poder de criação dando ao sistema a habilidade de adicionar cores a cada uma dessas partes. Isso torna possível a troca de cores de cada elemento e a adição de mais personalidade a cada avatar.

Essa permutação visual é baseada no valor *modulate* (modular) da cor de uma textura dentro do motor *Godot* que aplica uma multiplicação da cor da textura atual por outra. O

resultado matemático disso utilizando uma cor neutra na textura como tons de branco e cinza claro por um vermelho, resultará num vermelho quase idêntico ou bem similar ao tom escolhido. A Figura 20 representa como este processo ocorre visualmente.

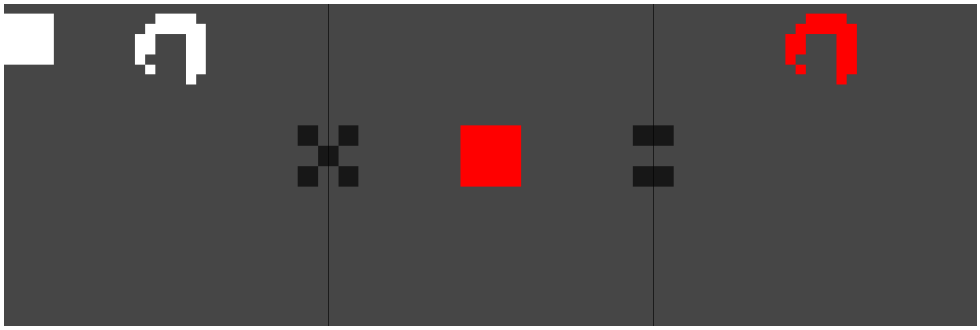


Figura 20 – Representação da multiplicação de uma textura neutra na cor branca por uma cor vermelha a qual se deseja pintar o acessório da cabeça, resultadando no acessório em vermelho.

O resultado é um sistema capaz de combinar inúmeros acessórios com cores diferentes para um avatar, sendo possível alterar em tempo real também avatares como a da Figura 19 e alterar o valor de suas cores. A Figura 21 mostra o mesmo avatar, porém colorido com outros valores de cor para todos os elementos

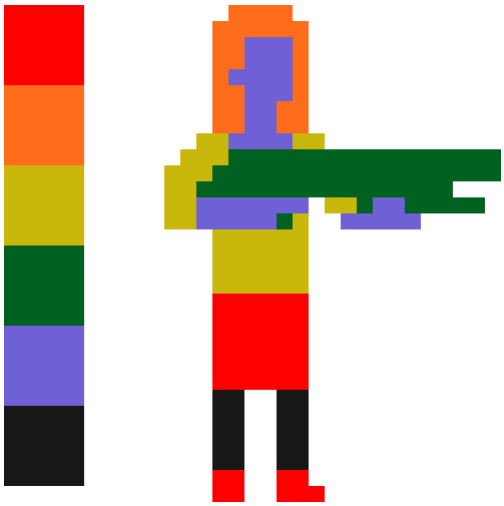


Figura 21 – Variação do avatar do jogo combinado em 6 partes tendo uma cor correspondente para cada uma. (AZUL) parte de cima do corpo. (PRETO) parte de baixo do corpo. (LARANJA) acessório da cabeça. (AMARELO) acessório da parte de cima do corpo. (VERMELHO) acessório da parte de baixo do corpo. (VERDE) a arma.

Um avatar ao ser criado não possuirá cores nem acessórios, apenas texturas da parte de cima e de baixo do corpo em branco aguardando que o jogador ou o sistema aplique as alterações visuais neste. A Figura 22 mostra como um avatar aparece no momento que é carregado, à espera de acessórios que o customizem antes de ser colocado no jogo.

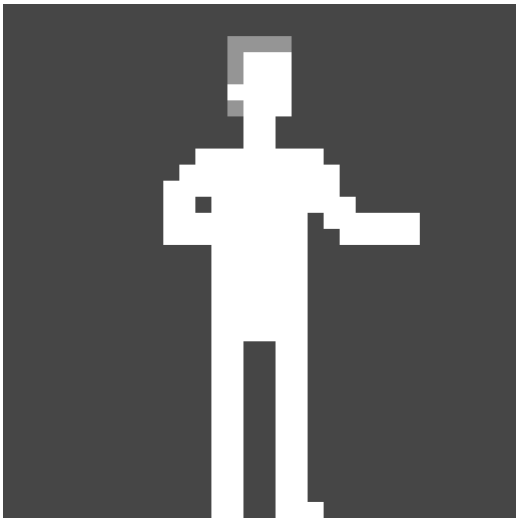


Figura 22 – Representação do Avatar antes de receber qualquer modificação ou alteração.

Quando a criação de avatares modulares está feita, eles conseguem ter acessórios e cores, porém faltam animações para uma maior qualidade gráfica e resposta visual para o jogo e jogadores.

5.2 ANIMAÇÃO DE COMPONENTES MODULARES

5.2.1 Primeiros Passos

As ações do jogador e inimigos que devem ser representados em um jogo para informar i que está acontecendo. Sendo assim, o avatar deve ser capaz de mostrar o que está fazendo sem contar explicitamente, apenas visualmente, caso ele esteja atirando, andando, trocando de arma, segurando uma arma ou coletando itens. Para isso utiliza-se a animação para comunicar sem falar ao jogador o que se passa dentro do jogo.

A animação tem esse papel de contar o que acontece com os elementos que estão na tela além de dar resposta ao jogador de suas ações. O quão elaboradas estas animações devem ser depende do nível de fidelidade da representação ser alta ou baixa. E tendo em vista a direção em estilo retrô de pixel art do presente trabalho, este nível pode ser baixo sem comprometer muito a qualidade. Isso significa que poucas telas de animação foram necessárias, e pouco retrabalho ocorreu por conta das poses similares que o avatar executa.

Foi importante nesse momento durante a implementação ter as mecânicas do jogo estipuladas e dentro de um escopo. Isso porque uma maior alteração de direção do projeto poderia ter comprometido o trabalho inteiro ao tentar representar algo que fosse distante do foco proposto.

Tendo como objetivo um jogo do gênero de tiro com a visão de cima e pra baixo com inimigos que atacam o jogador de todos os lados, uma movimentação que correspondesse a oito direções seria suficiente. Para comunicar isso visualmente em uma animação, não se fez

necessário simular as pernas do avatar nestas oito direções, apenas quatro mostram o que o avatar está fazendo e para onde ele está indo.

Com quatro direções para representar, não necessariamente significa que foram necessários quatro desenhos para cada direção já que dependendo da ação que o avatar executa, uma tela já transmite o que ele está fazendo. Por exemplo, estar parado, mostrando apenas o desenho das pernas paradas. No entanto para convencer que o avatar está se movimentando foi necessário um pouco mais de detalhes. Usou-se duas telas de animação para cada perna, somando quatro telas ao total possibilitou-se o movimento do avatar para frente e para trás. A Figura 23 mostra como são as pernas do avatar em cada 4 telas na animação de caminhada.

Figura 23 – Atlas de textura da animação das pernas do avatar andando para frente e para trás do avatar.

Para poupar trabalho na *Godot* em ter que escrever no código cada uma das telas de animação foi utilizado um Atlas de textura. Esta é uma convenção que permite matematicamente posicionar uma imagem em uma matriz de animação. A Figura 23 mostra como uma matriz de 1 linha por 4 de colunas é disposta.

Essa técnica ajudou significativamente, pois as animações são agrupadas em partes do personagem e sincronizadas no código. Dessa forma foi possível criar um atlas com todas as animações necessárias das pernas. O processo é ilustrado na Figura 24, que mostra as animações dessa forma.

É interessante apontar que não foi necessário redesenhar a animação das pernas andando para esquerda, pois no *Godot* foi possível inverter a textura no eixo x, como num espelho. Dessa forma, faz-se ir para esquerda com o mesmo desenho. Ganha-se bastante em tempo já que novos acessórios desenhados seguiam esse padrão, não precisavam ter mais uma linha de desenho feita. O único problema que isso proporciona é a questão de detalhe, visto que a animação de caminhada para esquerda será sempre idêntica a animação da direita. Dessa forma, assimetrias de animação são inconsistentes. Imagine, por exemplo, um personagem com uma perna de pau aonde essa perna muda de posição toda vez que é invertida.

É importante apontar que na necessidade de uma nova animação basta adicionar uma linha nova de desenhos no atlas, sendo assim, caso fosse adicionado mais 4 animações no atlas atual, ele se tornaria uma matriz 8x4. No momento que uma incrementação dessas é feita, é necessário que os parâmetros de leitura do atlas de textura sejam alterados em código ou no motor da *Godot*, já que a adição de uma nova linha ou coluna de animação pode causar que uma

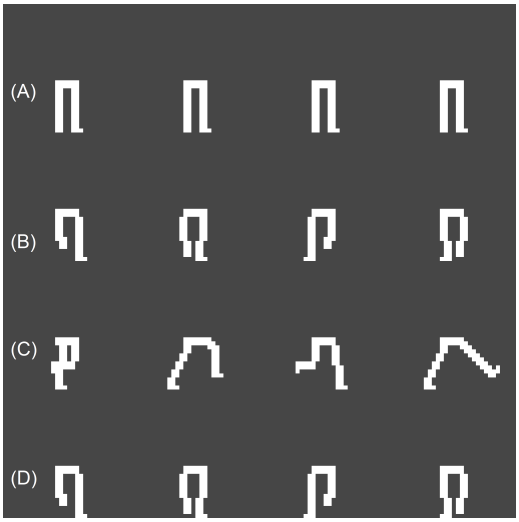


Figura 24 – Atlas de textura das animações das pernas do avatar. (A) parado. (B) Andando para frente. (C) andando para direita. (D) andando para trás.

animação não seja acessada por estar numa coluna além do valor que o controlador consegue acessar. Na Figura 24 caso uma quinta linha (E) de animação fosse adicionada esta nunca seria acessada pela nova entrada, tornando-se uma matriz de tamanho 5x4 e não mais 4x4.

Neste momento as pernas do avatar podem se mover independentes do resto do corpo, se o avatar por acaso estiver executando uma animação que seja para direita, as pernas podem interindependentemente irem para a esquerda, graças a combinação modular das texturas que foi feita na seção 5.1.

5.2.2 Mira e Complexidade de Animações

O presento trabalho requer que o avatar do jogador seja capaz de atirar nos inimigos que o atacarem, para isso no planejamento foi definido um arsenal de armas que pode ser utilizado. Tendo que inimigos venham de todos os lados para atacar, o jogador tem a possibilidade de atirar para apenas quatro lados: cima, baixo, esquerda e direita. Limitar o jogador a utilizar apenas essas quatro direções adiciona dificuldade e também diminuí o trabalho necessário para desenhar e representar aonde ele está mirando. Caso 8 direções de mira ainda fossem desejadas, seria possível apenas fazer os projéteis da arma sairem na posição mais próxima da direção que o jogador estivesse mirando. Por questão de fidelidade da representação visual, manteve-se quatro direções de mira. Isso porque, na parte de cima do corpo do avatar, este pode segurar mais de um tipo de arma por vez, no total sendo três armas mais a opção de não segurar nenhuma.

Começando da animação em que o avatar não segura nenhuma arma, são necessárias quatro telas a fim de representá-lo olhando para as quatro ou três, pois utilizando da inversão da textura no eixo x foi possível usar do desenho olhando da direita para representar o avatar olhando para esquerda, totalizando 3 telas. A Figura 25 mostra a matriz 1x3 do atlas de textura da parte de cima do corpo.

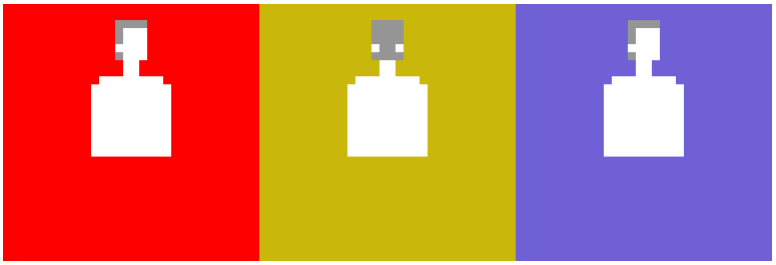


Figura 25 – Atlas de textura da animação da parte de cima do corpo. (VERMELHO) virado para direita. (AMARELO) virado para trás. (AZUL) virado para frente.

É possível reparar que a tela do avatar virado para direita e para frente são idênticos, contudo este padrão é necessário para que sejam adicionados detalhes com o intuito de que as armas e partes do corpo sejam devidamente representadas.

O avatar apenas estático não possui muito apreço ou vida, por isso foi adicionado mais uma tela para cada uma das direções dele respirando. Removeu-se apenas uma linha de pixels do torso e desceu-se a parte de cima do corpo, totalizando agora 6 telas, pois foi acrescentado 1 tela pra cada uma que já existia. A Figura 26 mostra a matriz 1x3 do atlas de textura da parte de cima do corpo.

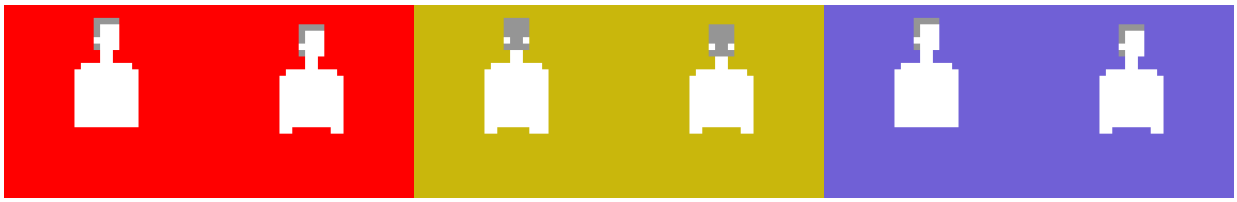


Figura 26 – Atlas de textura da animação da parte de cima do corpo agora com mais uma tela de animação para simular respiração do avatar. (VERMELHO) virado para direita. (AMARELO) virado para trás. (AZUL) virado para frente.

Foi adicionado então mais detalhe para animação do avatar que é capaz de mirar para os quatro lados independente da animação das pernas. Nesse contexto, faltam as poses de animação para este conseguir segurar as armas apropriadamente. Tendo três armas para adicionar, foram mais 3 linhas de animação adicionados ao atlas. A Figura 27 mostra o atlas de textura completo da parte de cima do corpo. Totalizando 24 telas de animação.

Aproveitando tal solução, foi possível determinar os atlas de cada arma, sendo elas uma matriz 1x6. A Figura 28 mostra o atlas da espingarda.

A partir deste padrão, outras armas seguem o mesmo modo de construção. Caso novas fossem adicionadas, basta criar duas telas de animação para direção para direita, atrás e frente respectivamente.

As animações que restam a ser implementadas são da cabeça e dos acessórios do corpo. A cabeça por ser uma parte tão centralizada em perspectiva com o corpo do avatar não exige muito detalhe. Desse modo optou-se por usar apenas uma tela que mostrasse a cabeça para trás e outra para frente. A tela que mostra o rosto para frente também é utilizada para representar

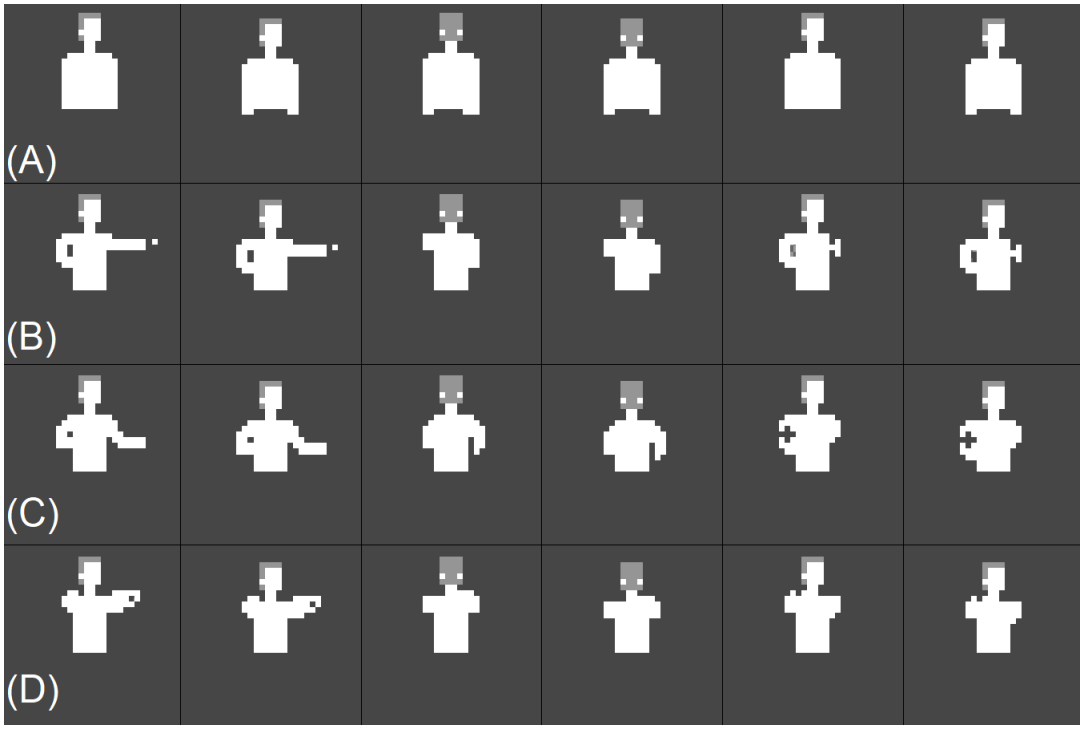


Figura 27 – Atlas de textura da animação da parte de cima completo com todas animações. Linha (A) desarmado. Linha (B) segurando metralhadora. Linha (C) segurando espingarda. Linha (D) segurando pistola.

Figura 28 – Atlas de textura de animação da espingarda utilizada no jogo. (VERMELHO) virada pra direita. (AMARELO) Virada pra trás. (AZUL) virada para frente.

o rosto olhando para direita e esquerda. O movimento de respiração do torso é sincronizado também nessa textura da cabeça, adicionando mais uma tela, totalizando um atlas de textura no formato 2X2. A Figura 29 mostra o atlas as animações da cabeça.

Por fim, implementou-se a vestimenta do avatar. Para isso foi reutilizado o mesmo padrão dos atlas criados para as pernas e o torso do avatar. Apenas agora representar camiseta, calças e roupas no geral. a Figura 30 mostra o atlas das animações de uma regata.

Nota-se que em algumas telas de (B), (C) e (D) da Figura 30, existem vácuos de pixel no meio da roupa. Isso acontece para representar o braço que estaria em cima da camiseta, mas como a textura fica em baixo, é necessário remover alguns pixels para que o braço seja representado seguindo o padrão que foi feito. Uma solução seria modularizar os braços do tronco e colocar estes em cima da textura do corpo e dos acessórios da parte de cima do corpo e depois a textura da arma.

O trabalho até então se resumiu a criar padrões de criação com o mínimo de trabalho

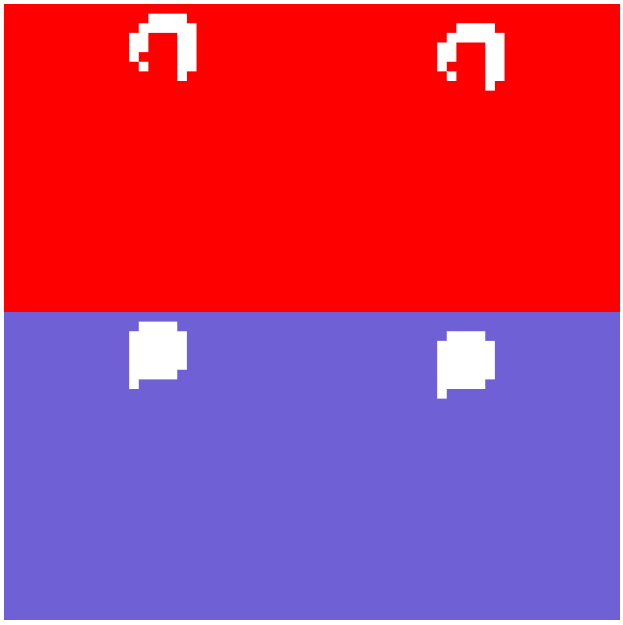


Figura 29 – Atlas de textura de animação da espingarda utilizada no jogo. (VERMELHO) virada pra frente, esta que também é reutilizada para direita e esquerda. (AZUL) virada para trás.

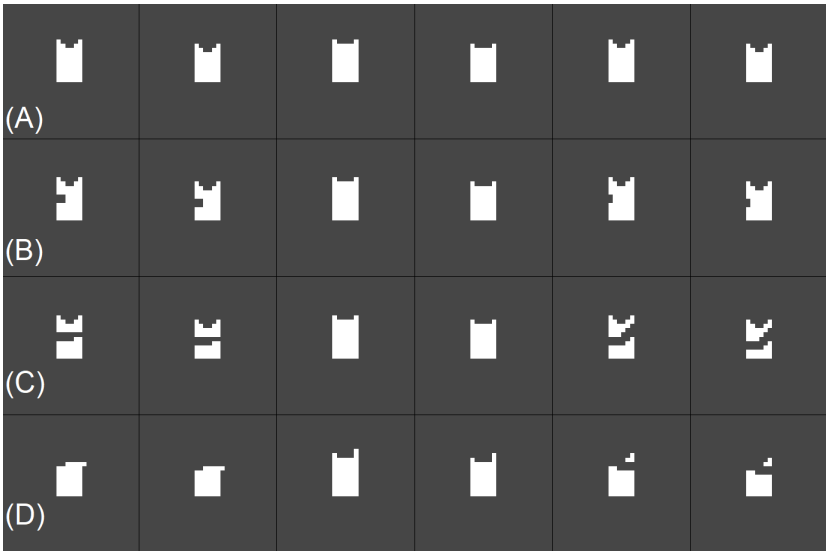


Figura 30 – Atlas de textura da animação da parte de cima completa com todas animações para uma regata. Linha (A) desarmado. Linha (B) segurando metralhadora. Linha (C) segurando espingarda. Linha (D) segurando pistola.

e o máximo de reaproveitamento. Adicionar mais cinco roupas para a parte de cima do avatar significa desenhar mais 24 telas por roupa, totalizando 120 telas. Ressaltando que o estilo ser em pixel art com padrão de 32x32 pixels, constata-se que o trabalho total de novas cinco roupas da parte de cima seria de desenhar 122.880 unidades de pixels na tela.

Muito parecido com o que acontece com as roupas da parte de baixo do avatar, o padrão das pernas também é reutilizado para desenhar qualquer acessório que possa vestir. A Figura 31 mostra o atlas das animações de uma bermuda.

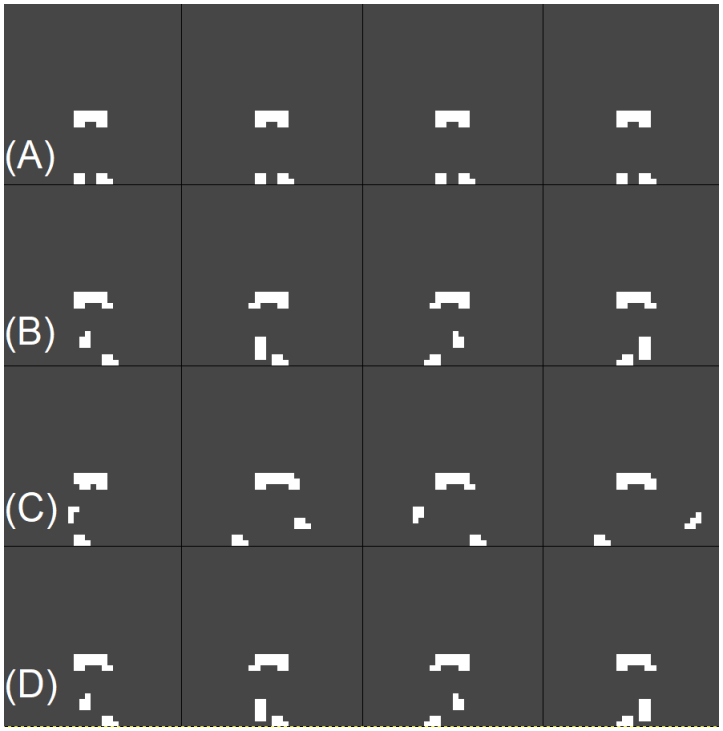


Figura 31 – Atlas de textura da animação da parte de cima completa com todas animações para uma regata. Linha (A) parado. Linha (B) andando para frente. Linha (C) andando para direita. Linha (D) andando para trás.

Para animar propriamente o avatar foi necessário atribuir variáveis que guardassem as direções de movimentação e mira, definidas para quatro direções como as animações foram projetadas. O modo como foi implementado neste trabalho foi verificar qual a última direção que os botões de movimentação e mira estavam e colocar o respectivo atlas para repetir os quadros de animação da ação que está sendo feita. Toda vez que uma arma é trocada do inventário do jogador, é disparado o valor correspondente da arma com a linha de animação do atlas da parte superior do corpo. Conforme ilustrado na Figura 30, cada arma possui uma linha de animação e para cada direção que essa arma aponta um par de colunas correspondentes para animar.

5.3 GERAÇÃO PROCEDURAL DE INIMIGOS

Utilizando o motor de jogos da *Godot* foi possível montar a maquete para um jogo de tiro existir, a movimentação do jogador, selecionar e usar armas, criar inimigos que perseguem e causam dano ao jogador, enfrentamento dos inimigos utilizando armas, um sistema de pontos que recompensa o jogador a cada inimigo derrotado e criação de itens para recuperar quantidade de vida e munição que o jogador possui. A Figura 32 mostra uma captura de tela do jogo.

Figura 32 – Captura de tela do jogador enfrentando inimigos que se aproximam para atacar.

5.3.1 Personalização do Jogador

Além das mecânicas de tiro, pontuação, o presente trabalho tem um foco especial da modificação dos avatares. As primeiras versões do jogo eram apenas o avatar do jogador que andava de um lado para o outro na tela sem arma alguma, com apenas os acessórios do corpo. A Figura 33 mostra o rascunho de como seria a tela de modificação do jogador.

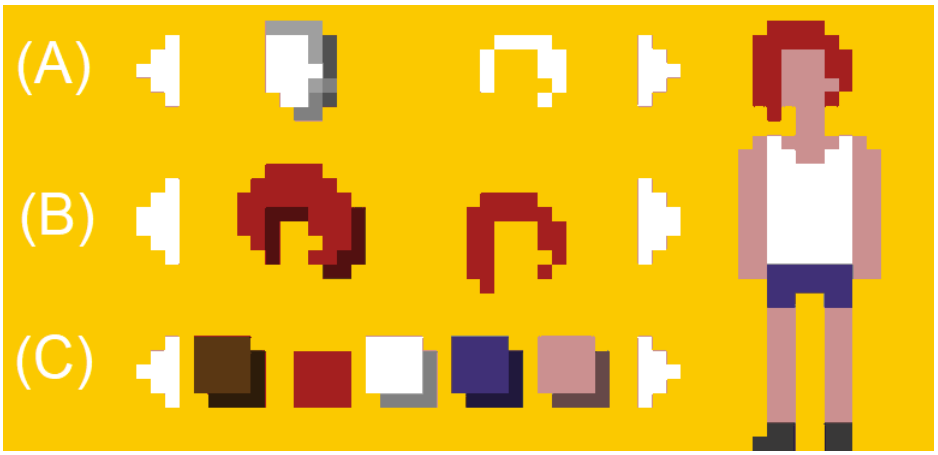


Figura 33 – Rascunho do que seria o sistema de Guarda Roupa do jogador, este que permite modificar o persoangem visualmente, em um menu em que o jogador clica nos botões para escolher qual parte do corpo modificar, acessórios e cores. (A) mostra a parte do corpo selecionada para personalizar. (B) mostra acessórios da parte selecionada para escolher. (C) cores que podem ser aplicadas ao acessórios.

A ideia original também era de permitir que jogadores pudessem acessar as pastas onde texturas do jogo ficam salvas e adicionarem novos acessórios para personalizar seus personagens.

Dessa forma a implementação do sistema de guarda roupa para o jogador começou em apenas listar todos os arquivos do tipo png que existiam nas pastas de texturas para personalizar o avatar e salvar no próprio jogador esta lista de materiais carregados na memória. Isto mostrou-se muito custoso visto que depois que se criava mais avatares do tipo jogador, e todos eles carregavam, muita memória era utilizada para armazenar as texturas para uso.

Por menores que sejam as texturas em pixel art, isso ainda não evitava ajustes de performance, como o problema de todas as texturas serem carregadas por cada avatar. Essa dificuldade foi resolvida mais tarde fazendo a cena principal parar carregar os níveis e salvar uma referência para as texturas dos avatares. Cada entidade carrega na memória apenas o que está desenhando na tela do seu avatar, e não todos os acessórios os quais poderia usar.

Uma abordagem semelhante foi utilizada para gerar os inimigos, onde a cena principal encarregada de salvar as texturas do jogo também determina e cria essas entidades.

5.3.2 Populações de Inimigos

Os jogos *Left 4 Dead 1 & 2* utilizam de um controlador chamado de *diretor* (VLACHOS, 2010) para criar os zumbis do jogo que atacam o jogador. Uma abordagem semelhante foi implementada no jogo deste trabalho. Um controlador é responsável por decidir que tipo de inimigo criar, o controle dele é definido por parâmetros de chance de popular o mapa com inimigos de acordo com porcentagens pré estipuladas. Esse controle é exercido seguindo uma regra de criação com populações e quem faz parte delas. Uma população é algo temático como policiais, trabalhadores de construção, enfermeiros, trabalhadores que possuem um uniforme distinto geralmente. Cada uma delas tem uma chance percentual de ser gerada, sendo possível que uns apareçam mais do que outros. Por exemplo, um indivíduo da população de policiais tem 50% de ser gerada, construção tem 25% e enfermeiros 25%. Isso significa ao longo do tempo de um ciclo de geração, haverão 2 policiais pra cada par de enfermeiro e construtor serem criados.

É possível subdividir populações também, colocando chances de cada construtor ter o uniforme cinza e não azul. Isso acaba trazendo mais raridade pra determinadas gerações que possuem baixas chances de ocorrerem.

Os inimigos comuns em Left 4 Dead são os mesmos em jogabilidade tendo de diferença a aparência entre eles. O que acaba sendo reproduzido e expandido nesse trabalho é a implementação de um controlador para populações de inimigos, que dependendo de sua aparência tem atributos modificados de outros.

5.3.3 Seleção das Populações do Jogo

Depois de desenhar alguns acessórios para popular o jogo, roupas, máscaras, cabelos, foi possível fazer uma seleção de populações que pudessem se relacionar. As populações foram nomeadas como:

- Civis, 80% de chance de ser gerado.
- Felpudos, 10% de chance de ser gerado.
- Carreta Furacão, 10% de chance de ser gerado.

Somando as três populações, obtem-se 100% de criação de alguma delas. A população de *Civil* é composta por avatares que usam acessórios comuns do dia a dia, principalmente acessórios da cabeça, como penteados, bonés, chapéus. A Figura 34 mostra um exemplo de civil gerado dentro do jogo.

Figura 34 – Avatar de civil de cabelo longo gerado dentro do jogo, possui roupas de coloração azul acizentado e veste um macacão.

Esta população é a que tem maior chance de ter um membro criado, com uma porcentagem de 80% estipulada dentro do jogo. Estes inimigos são os que possuem maior velocidade, mas tem menos dano e pontos de vida.

A população de *Felpudos* é composta por avatares que possuem cabeças de animais, como patos, raposas, ursos, e possuem roupas comuns do dia a dia. A Figura 35 mostra um exemplo de felpudos gerado dentro do jogo.

Esta população tem menor chance de ter um membro criado, com uma porcentagem de 10% estipulado dentro do jogo. Estes inimigos possuem menor velocidade, mas ganham mais pontos de vida e maior dano.

A população de *Carreta Furacão* é composta por membros que usam máscaras de personagens famosos da televisão brasileira. A Figura 36 mostra um exemplo de carreta furacão gerado dentro do jogo.

O controle foi feito colocando-se todos os itens do que pode ser acessório da cabeça de avatares em uma lista para cada uma das populações se diferenciarem. Exceto pelo acessório da cabeça do avatar o resto montado pelo controlador é decidido independe da população selecionada. A Figura 37 demonstra um exemplo de cada item de um avatar ter sido escolhido.

Figura 35 – Avatar de felpudo cabeça de rato gerado dentro do jogo. Um exemplo de felpudo gerado dentro do jogo.

Figura 36 – Avatar de Carreta Furacão com cabeça de personagem infantil antigo. Um exemplo de carreta furacão gerado dentro do jogo.

5.3.4 Aplicação de Probabilidade Acumulada

Para atribuir chances de algo ser escolhido no jogo, a probabilidade acumulada foi uma solução para o problema. Apenas selecionar elementos aleatórios para o avatar não é suficiente. É essencial para que o conceito das populações seja aplicado que o número de avatares inimigos gerados seja proporcional a porcentagem estipulada para cada população. Assim foi testado gerar mais de dez mil avatares entre três populações com a distribuição de 50%, 25% e 25% obtendo o número de avatares total distribuídos de acordo com as chances de sua população.

A probabilidade acumulada exerce o seguinte papel neste trabalho: dada uma lista de objetos que possuem chances distintas de serem selecionados, suas chances são somadas de forma que formem algo inteiro, 1 ou 100% e um número aleatório é escolhido para esse intervalo que começa de 0 até 1. As probabilidades distintas são respeitadas dessa forma, por que cada uma delas acaba por ocupar um espaço dentro desse intervalo de 0 até 1. Por exemplo,

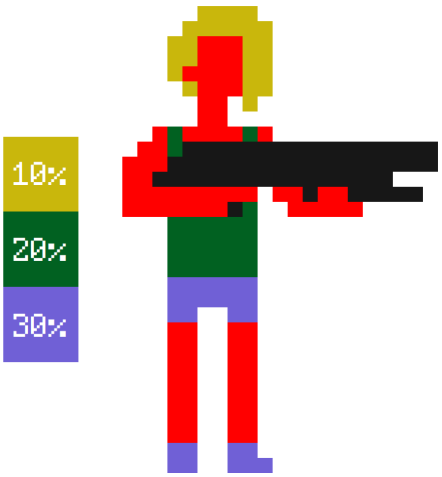


Figura 37 – Avatar gerado com seus acessórios pintados da cor de sua chance correspondente de serem escolhidos, por exemplo nesse caso, 10% do acessório da cabeça ser um cabelo de franja, 20% de vestirem regata e 30% de vestirem bermuda.

distribuições de 50%, 40% e 10% são representadas na probabilidade acumulada como 0.5, 0.9 e 1, pois 50% começa de 0 até 0.5, 40% dos maiores que 0.5 até 0.9 e 10% dos maiores que 0.9 até 1 e somadas formam um valor inteiro de 100% ou 1. A Figura 38 demonstra isso.

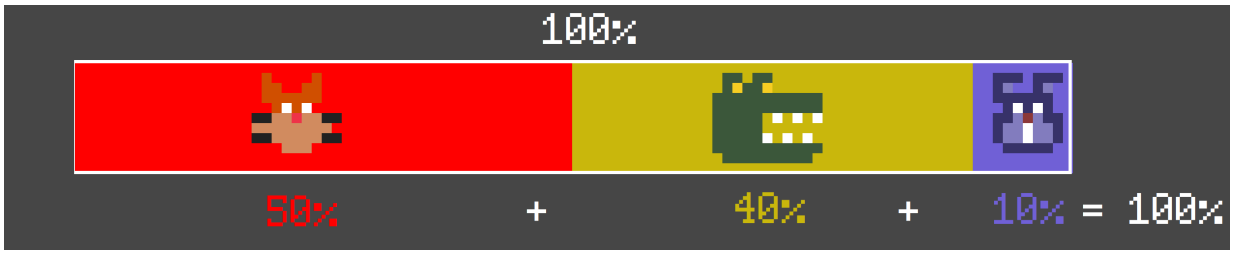


Figura 38 – Demonstração Visual da Probabilidade Acumulada, onde considerando uma imagem que corresponda a 100%, as chances de serem gato ocupam metade da imagem, 50%, as chances de serem jacaré ocupam 40% e as chances de resultar em coelho são muito pequenas, apenas 10% da imagem.

Assim a probabilidade de um número aleatório cair no intervalo de 0 até 0.5 são metade das respostas que o resultado pode oferecer, sendo que este intervalo pertence ao acessório de gato, o escolhido será ele, e não os demais que participam.

No controlador existe uma lista dos nomes de populações as chances correspondentes delas serem escolhidas. A Figura 38 mostra isso dentro do código.

O mesmo acontece para os acessórios das cabeças que compõe determinada população. A Figura 40 mostra a lista de acessórios que fazem parte da população civil com as chances e os respectivos caminhos dentro do projeto para carregar a textura.

Para todos os acessórios e a população de um avatar a solução é a mesma: utilizou-se da probabilidade acumulada, mudando-se apenas os componentes dentro das listas e as chances que tem de serem escolhidos. A Figura 41 representa a árvore de decisão que acaba sendo tomada no momento de escolher um acessório de uma avatar.

```
150  #chance of each populatio to spawn
151  var population_chance = [
152      0.1,
153      0.2,
154      1
155  ]
156  var population_name = [
157      "furry",
158      "carreta",
159      "common"
160      ]
```

Figura 39 – Lista das chances acumuladas das populações e lista dos nomes correspondentes.

Antes de implementar a solução para a geração dos inimigos, os avatares recebiam apenas valores aleatórios para cada acessório que correspondiam a uma posição da lista de itens lidos dentro de uma pasta das partes do avatar. As gerações então não tinham controle de cor, ou padrão de repetição entre os acessórios ou de populações.

```
192  #chance of each common head asset to spawn
193  var common_chance = [
194      0.1,
195      0.15,
196      0.20,
197      0.25,
198      0.30,
199      0.35,
200      0.40,
201      0.45,
202      0.50,
203      0.55,
204      0.60,
205      0.70,
206      0.80,
207      0.90,
208      1
209  ]
210  var common_name = [
211      "res://Textures/Head/big_curvy.png",
212      "res://Textures/Head/excentric.png",
213      "res://Textures/Head/fringe.png",
214      "res://Textures/Head/horns.png",
215      "res://Textures/Head/long.png",
216      "res://Textures/Head/long_cute.png",
217      "res://Textures/Head/long_fringe.png",
218      "res://Textures/Head/none.png",
219      "res://Textures/Head/ponny_tail.png",
220      "res://Textures/Head/power.png",
221      "res://Textures/Head/sombrero.png",
222      "res://Textures/Head/star.png",
223      "res://Textures/Head/venom.png",
224      "res://Textures/Head/emo.png",
225      "res://Textures/Head/cap.png",
226      "res://Textures/Head/short_and_simple.png",
227      "res://Textures/Head/talk_show.png",
228  ]
```

Figura 40 – Lista das chances acumuladas de acessórios e lista dos caminhos das texturas dentro do projeto da população de civil.

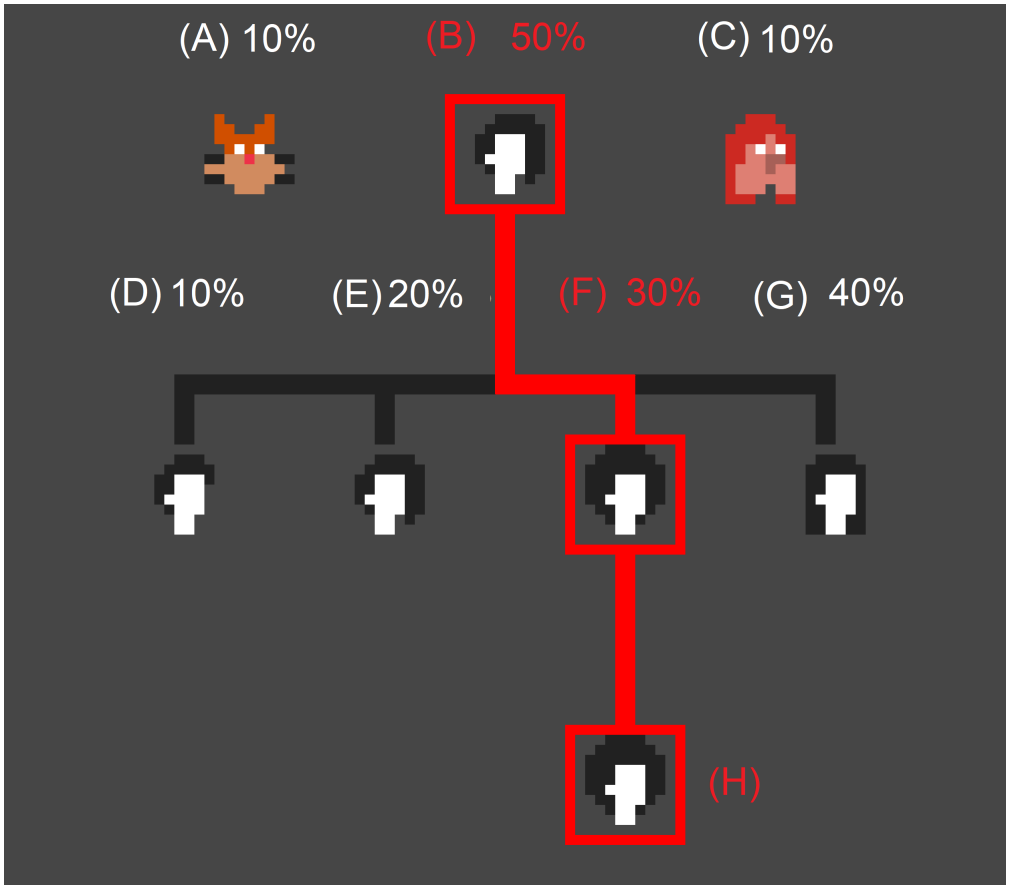


Figura 41 – Árvore com o processo de escolha do acessório da cabeça de um avatar. Populações correspondem aos itens (A), (B), (C), onde (B), em vermelho, é o escolhido, a seguir as opções de acessório da cabeça da população escolhidas são (D), (E), (F), (G), onde (F), em vermelho, é o escolhido, e o acessório final resulta no item (H).

6 RESULTADOS

O jogo desenvolvido é capaz de criar um avatar composto por uma parte do corpo inteira da mesma cor, 30 acessórios de cabeça, 6 acessórios da parte de baixo do corpo, 7 acessórios da parte de cima do corpo, 3 armas, com direito a 48 cores da grade de seleção disponível que podem alterar todas essas partes a não ser o armamento, totalizando 418.037.760 avatares únicos que podem ser gerados.

O número de combinações talvez garanta que pouquíssimos avatares enfrentados sejam vistos novamente em novas jogatinas, mantendo a experiência mecânica que pode ser repetitiva mais visualmente refrescante.

Os números também podem criar uma falsa impressão de que todo inimigo será novo, o que pode não ser o caso já que a maior parte da variação acontece pela extensão que as cores alterando acessórios do avatar podem causar. Atributos como a silhueta, forma, tamanho, até mesmo olhos que alguns acessórios da cabeça de avatar acabam por chamar mais atenção dentro do jogo, o que pode resultar na fácil identificação de acessórios já vistos mas apenas recoloridos.

A etapa de projeto do jogo necessitou muito planejamento, além de conhecimento técnico e artístico, tanto para representar o trabalho que se desejou fazer assim como para executa-lo. Equipes pequenas e principalmente iniciantes podem ter bastante dificuldade de reproduzir este trabalho assim como não ter tempo ou recurso suficientes para desenvolver.

Um cuidado durante este tipo de projeto é conseguir medir o preciosismo necessário para evitar dívidas técnicas no código, ter que refazer partes do projeto por falta de planejamento futuro, ao mesmo tempo em que ele pode ficar estagnado por não achar uma solução que responda todas as demandas.

Se um gráfico de desenvolvimento do jogo utilizando a geração procedural neste caso fosse desenhado, onde y corresponde ao desenvolvimento real de código e a um produto mínimo viável (MVP) e x representa o tempo, ele tenderia a zero em y por longo tempo até que começa ter forma. Sempre que uma tecnologia nova for implementada em um trabalho é crucial que se pergunte se aquilo realmente é importante, se vai mais somar que tirar do projeto.

Dependendo de como é definido o escopo do trabalho e a necessidade de alteração dele no meio do desenvolvimento, é possível que muitos elementos se percam. O trabalho para projetar e criar um jogo com geração procedural de conteúdo demandará refatoração do código ou até mesmo que este seja descartado por conta das mudanças. Em algum momento foi gasto tempo considerável para alinhar armas que fossem de corpo a corpo. Contudo isso foi retirado afim de terminar o trabalho dentro do prazo.

Observando a execução do jogo em plataformas diferentes, constata-se que em algumas delas a geração procedural não funcionava de forma correta sendo a mesma versão do trabalho utilizado em todos os casos. Inimigos repetiriam até 4 vezes indiscriminadamente e não em uma sessão de jogo apenas, mas também em até diferentes máquinas.

O jogo proporciona variabilidade mesmo com os possíveis defeitos de projeto. Deixar de

se preocupar com listas de personagens criados manualmente e deixar que o sistema gere automaticamente é refrescante devido à criatividade que a máquina consegue aplicar nas variações. Algumas delas, inclusive, que "saltam aos olhos"por serem tão raras e diferentes da maioria. São estes momentos que o sistema com GPC pode proporcionar um sentimento de recompensa.

A lógica para projetar e montar avatares, assim como organizar as animações para serem estendidas para inúmeros contextos, como ambientes com construções modulares, criação de criaturas não é limitado ao ambiente 2D pixel art.

Ao trabalhar com *Godot*, foi cometido um erro significante que tomou muito tempo de desenvolvimento. Um dos objetivos era adicionar um arquivo editável de texto que pudesse ter uma lista dos acessórios com atributos nele. Não obtendo sucesso com essa implementação, decidiu-se por usar listas de vetores dentro da cena principal para guardar os detalhes de geração do controlador.

Criar acessórios seguindo o padrão para cada parte do corpo do avatar é relativamente rápido devido à escala e baixa fidelidade que a arte precisa ter para represntar. No entanto, dependendo de como o projeto pode crescer em quantidades de animações por parte ou quantidades de parte até, o processo seria muito mais extenso.

No momento em que a GPC começou a criar seus primeiros avatares aleatórios, lendo arquivos de pasta, ao ser exportado do *Godot* para outra plataforma o jogo não era capaz de encontrar mais nenhuma textura ou arquivo de imagem. Se não fossem fóruns e blogs enriquecidos de debate e dúvidas respondidas pela comunidade que utiliza *Godot*, o desenvolvimento desse trabalho estaria comprometido. O motor *Godot* pode no momento deste trabalho não ser tão popular ou robusto quanto outras do mercado, mas sua simplicidade e até mesmo descaso com alguns recursos mais populares ajudam quem desenvolve a manter o foco em partes mais importantes dentro de um jogo.

7 CONCLUSÃO

Esse projeto apresentou um jogo implementado com a ideia de visualizar um avatar de forma modular e disso criar inúmeras variações com geração procedural de conteúdo no contexto de um jogo de tiro, com perspectiva de cima pra baixo, no estilo pixel art. As etapas descritas neste trabalho possibilitaram a construção de um jogo usando essa tecnologia e mostraram os benefícios e desafios que este tipo de trabalho pode proporcionar. Devido à pandemia do Coronavírus, o projeto que tinha intenção de testar a percepção de jogadores a variabilidade de inimigos foi adaptado e reduzido para ser trabalhado em menor tempo e exigindo menos envolvimento social. Tomou-se, então, como foco principal a avaliação dos elementos técnicos e artísticos para projetar este tipo de trabalho.

Também foi vislumbrada a possibilidade de reaproveitamento desse sistema para futuros projetos, tendo em vista que o sistema é capaz de receber novas animações e acessórios para alimentar o jogo e criar novos personagens.

Por fim, desenvolver um modelo de geração procedural de conteúdo para jogos 2D exige do programador e do artista que trabalhem juntos para conceber o sistema dentro de um jogo. A união das duas áreas possibilita construir sistemas GPC capazes de criarem possibilidades que os desenvolvedores não tinham imaginado. O jogo desenvolvido neste trabalho também contava com inimigos que usavam armas assim como as do jogador, sendo que o avatar já é capaz de representar isso visualmente. No entanto não foi possível em tempo hábil desenvolver a autonomia dos inimigos. O trabalho também tinha interesse em realizar testes com jogadores em dois grupos sendo um jogando uma versão do jogo com maior variabilidade que a do outro. Para assim avaliar o impacto dessa variação dentro da percepção dos jogadores, o que resultou em uma lacuna para trabalho futuro.

7.1 TRABALHOS FUTUROS

Para futuros projetos, existem algumas propostas e tentativas válidas a serem feitas com base em modificar não apenas aparência de avatares mas seu comportamento também. Explorar a geração da aparência depender de fatores não apenas pré determinados como porcentagens junto de números aleatórios, mas algum tipo de respostas de interações do jogador dentro do jogo.

Da mesma forma, aplicar esse modelo de personalização em elementos não tão interativos em um jogo como são os personagens ou inimigos e estender a modularização para ambientes, cenários e até mesmo quebras cabeças que impedem o progresso de jogadores.

REFERÊNCIAS

AMATO, Alba. Procedural content generation in the game industry. In: GAME Dynamics. [S.l.]: Springer, 2017. p. 15–25.

BARRETO, Nuno; CARDOSO, Amılcar; ROQUE, Licınio. Computational creativity in procedural content generation: A state of the art survey. In: PROCEEDINGS of the 2014 conference of science and art of video games. [S.l.: s.n.], 2014.

CONNOR, Andy M; GREIG, Taura J; KRUSE, Jan. Evaluating the impact of procedurally generated content on game immersion. **The Computer Games Journal**, Springer, v. 6, n. 4, p. 209–225, 2017.

DAHL, Gustav; KRAUS, Martin. Measuring how game feel is influenced by the player avatar's acceleration and deceleration: using a 2D platformer to describe players' perception of controls in videogames. In: PROCEEDINGS of the 19th International Academic Mindtrek Conference. [S.l.: s.n.], 2015. p. 41–46.

HEIJNE, Norbert; BAKKES, Sander. Procedural zelda: A pcg environment for player experience research. In: PROCEEDINGS of the 12th International Conference on the Foundations of Digital Games. [S.l.: s.n.], 2017. p. 1–10.

JENNETT, Charlene et al. Measuring and defining the experience of immersion in games. **International journal of human-computer studies**, Elsevier, v. 66, n. 9, p. 641–661, 2008.

KORN, Oliver et al. Procedural content generation for game props? a study on the effects on user experience. **Computers in Entertainment (CIE)**, ACM New York, NY, USA, v. 15, n. 2, p. 1–15, 2017.

LIM, Sohye; REEVES, Byron. Computer agents versus avatars: Responses to interactive game characters controlled by a computer or other player. **International Journal of Human-Computer Studies**, Elsevier, v. 68, n. 1-2, p. 57–68, 2010.

SMITH, Gillian et al. PCG-based game design: enabling new play experiences through procedural content generation. In: PROCEEDINGS of the 2nd International Workshop on Procedural Content Generation in Games. [S.l.: s.n.], 2011. p. 1–4.

VLACHOS, Alex. **Rendering Wounds in Left 4 Dead 2**. [S.l.], 2010. p. 1–29.

YANNAKAKIS, Georgios N; TOGELIUS, Julian. Experience-driven procedural content generation. **IEEE Transactions on Affective Computing**, IEEE, v. 2, n. 3, p. 147–161, 2011.