Agenor Furigo Neto

DESENVOLVIMENTO DE PERSONAGEM PARA JOGOS ELETRÔNICOS 3D: CRIAÇÃO E CONSTRUÇÃO ADAPTADAS AO CONTEXTO NACIONAL DE PRODUÇÃO

Projeto de Conclusão de Curso do Departamento de Expressão Gráfica, Centro de Comunicação e Expressão da Universidade Federal de Santa Catarina, como requisito parcial para a obtenção do grau de bacharel em Design.

Orientador: Prof. Dr. William Machado de Andrade

Florianópolis
2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Furigo Neto, Agenor
DESENVOLVIMENTO DE PERSONAGEM PARA JOGOS ELETRÔNICOS 3D : CRIAÇÃO E CONSTRUÇÃO ADAPTADAS AO CONTEXTO NACIONAL DE PRODUÇÃO / Agenor Furigo Neto ; orientador, William Machado de Andrade, 2017.
109 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro de Comunicação e Expressão, Graduação em Design, Florianópolis, 2017.

Inclui referências.

1. Design. 2. Jogos Eletrônicos. 3. Modelagem 3D. 4. Otimização. 5. Personagem para jogos. I. Machado de Andrade, William. II. Universidade Federal de Santa Catarina. Graduação em Design. III. Título.

Agenor Furigo Neto

DESENVOLVIMENTO DE PERSONAGEM PARA JOGOS ELETRÔNICOS 3D: CRIAÇÃO E CONSTRUÇÃO ADAPTADAS AO CONTEXTO NACIONAL DE PRODUÇÃO

Esta monografia foi julgada adequada para obtenção do Título de "Bacharel em Design", e aprovado a em sua forma final pelo Curso de Design da Universidade Federal de Santa Catarina.

Florianópolis, 20 de novembro de 2017.

Profª. Marília Matos Gonçalves, Dr.ª
Coordenador do Curso

Banca Examinadora:

Prof. William Machado de Andrade, Dr.
Orientador
Universidade Federal de Santa Catarina

Prof.ª Monica Stein, Dr.ª
Universidade Federal de Santa Catarina

Prof. Flávio Andaló, Me.
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Aos meus pais, Agenor e Neusa, que me trouxeram a este mundo e, com seu amor incondicional, me criaram, me apoiaram e me ensinaram sobre todos os aspectos da vida, além de serem a maior força que eu poderia ter tido na difícil decisão de vivenciar esta segunda graduação.

À minha irmã Karina, por nunca deixar de estar ao meu lado e por todos os dias me mostrar como é importante ser incansável na busca do aperfeiçoamento pessoal.

Aos meus tios, tias, primos, primas e avós por me proporcionarem grandes momentos de alegria e pelo grande apoio.

A Thaïs e Lara que, nestes mais de dez anos de profunda amizade, nunca deixaram de me apoiar, aconselhar e compartilhar tanto bons e maus momentos, sempre com a mesma intensidade.

Aos meus amigos de infância, Bruno, Gustavo, Gustavinho, Luiz, Eduardo, Arthur e Guilherme, por sempre me trazerem grande alegria mesmo nos piores momentos, participando comigo de cada passo dessa caminhada até aqui.

A Ingrid, que navega comigo desde o começo do curso de Design e foi uma inspiração para continuar avançando nesta graduação, além de ter sido extremamente prestativa e compreensiva nos momentos que mais precisei.

A Nayara, sempre trazendo muitas risadas e alegrias, me ensinando também sobre persistência e trabalho duro, andando lado a lado comigo neste longo percurso da graduação.

A Amanda, com quem aprendi muito e que, com sua gentileza, leveza e força, me reergueu no momento que mais precisei, além da grande parceria no trabalho, sempre disposta a ensinar e a aprender.

A Boi e Rodrigo, por terem sido professores não oficiais neste curso, mostrando uma insaciável sede de conhecimento, além de disposição e paciência para dividi-lo.

A Bruna e Diogo, por terem me mostrado que grandes desafios podem ser encarados com leveza, gentileza, confiança e tranquilidade.

Ao professor William, por sua grande dedicação como professor, além da orientação e confiança, tornando este trabalho possível.

Ao professor Flávio, também por sua grande dedicação e disposição em ajudar nos momentos mais complicados,

À professora Mônica, por nunca desistir e por toda a ajuda e carinho que me deu durante o curso.

A todos os outros colegas, amigos e professores que contribuíram de alguma forma para a realização deste trabalho.

RESUMO

A área de jogos eletrônicos está entre os maiores mercados de entretenimento do mundo, movimentando muitos empregos e tecnologias diferentes, além de ser parte das atividades sociais de uma grande parcela da população mundial. Tendo em vista a importância deste mercado e seu pouco desenvolvimento no Brasil, buscou-se criar um personagem para jogos 3D voltado para os consoles de mesa atuais (2017), levando em conta as limitações existentes nessas plataformas. Para isso, utilizou-se uma metodologia já aplicada em grandes empresas, passando pelas fases de conceituação e pré-produção de jogos. O personagem foi modelado e preparado para a animação no programa 3DS Max, utilizando escaneamento facial como ferramenta para a modelagem de sua face. Suas texturas foram construídas no programa Adobe Photoshop e suas animações foram feitas por captura de movimento, sendo refinadas posteriormente no *software* Motion Builder. Com as animações prontas, o personagem foi colocado na *engine* de jogos Unreal Engine onde verificou-se que ele se encontrava dentro das especificações para um personagem de jogos para consoles, mostrando a viabilidade da produção deste tipo de personagem com as limitações e ferramentas apresentadas.

Palavras-chave: jogos digitais, vídeo games, jogos eletrônicos, personagens, modelagem 3D, animação 3D, *Game Design*,

ABSTRACT

Video games are among the largest entertainment industries in the world, generating many jobs, helping in the development of many different technologies while also being part of the social activities of a great portion of the world's population. Considering the importance of this market and its subpar development in Brazil, the main goal was to create a 3D character for current video game consoles (2017), taking into account the limitations of these platforms. To achieve this, a methodology already used in the industry was utilized, going through the Concept and Pre-Production phases of a regular video game development. The character model and animation skeleton were created using the software 3DS Max, utilizing 3D scanning as a tool for modeling its face. Its textures were done with the program Adobe Photoshop and its animations were created through motion capture, being refined later in the software Motion Builder. In the end, the character was imported to the Unreal Engine game engine, where it was confirmed that it could perform within the specifications of a modern game, showing that the production of this kind of character is viable with the limitations and tools presented.

Keywords: Digital games, vídeo games, eletronic games, 3D characters, 3D modeling, 3D animation, Game Design.

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

Este trabalho, iniciado no primeiro semestre do ano de 2017, é a etapa final do processo de desenvolvimento de um Projeto de Conclusão de Curso de Design da Universidade Federal de Santa Catarina.

O projeto visa aprimorar as competências do presente autor na área de animação e modelagem 3D, que eram partes integrais do currículo de Design na época do desenvolvimento do trabalho, produzindo um personagem para um jogo digital 3D.

Jogos digitais são produtos que uma grande parcela da população brasileira tem contato, pois estão presentes em computadores, celulares, videogames e outros equipamentos eletrônicos que são utilizados com frequência.

Este trabalho ajudará no entendimento dos desafios e problemas encontrados na área de jogos, que estão em crescimento no Brasil, movimentando mais de um bilhão e seiscentos milhões de dólares somente no ano de 2016 no país (MOGNON, 2017). Apesar do crescimento, desenvolvedores locais ainda têm dificuldades, ainda recorrendo com frequência à produção de jogos puramente educativos ou de cunho publicitário (MOGNON, 2017).

O completo estudo da criação de um personagem para um console pode ajudar a preencher as lacunas que ainda existem para o mercado brasileiro alcançar o patamar dos grandes desenvolvedores como os Estados Unidos, Europa e Japão.

Este documento visa mostrar todas as etapas do processo de desenvolvimento deste trabalho, demostrando as pesquisas e metodologias utilizadas para a construção do mesmo, sendo todas as etapas divididas em cinco capítulos.

No primeiro capítulo, chamado de Introdução, serão apresentados o objetivo principal do projeto bem como seus objetivos específicos associados. Também será apresentada uma justificativa demonstrando como os jogos são importantes nos âmbitos social e econômico. Por fim, será feita uma delimitação do projeto, mostrando quais áreas não serão exploradas no decorrer do desenvolvimento do mesmo.

No segundo capítulo, chamado de Revisão Bibliográfica, serão mostrados estudos essenciais para alcançar os objetivos do trabalho, apresentando primeiramente os principais problemas no desenvolvimento de um personagem 3D para um jogo digital e, em seguida, descrevendo como é produzido um jogo na indústria.

No terceiro capítulo, chamado de Metodologia, será mostrado um modo de produção de jogos que será utilizado no decorrer da etapa de desenvolvimento do trabalho, para a criação do personagem 3D.

No quarto capítulo, chamado de Desenvolvimento, são descritas todas as etapas do processo de construção do personagem, desde sua concepção até sua implementação em uma *engine* de jogos.

No quinto capítulo, chamado de Conclusão, são feitas as considerações finais, discutindo os objetivos concluídos, conhecimentos adquiridos e sugestões de trabalhos futuros.

Ao final são apresentadas as referências bibliográficas utilizadas para a escrita deste documento para que qualquer interessado possa ter acesso às informações.

1.1 OBJETIVO

O objetivo deste projeto é produzir um personagem para um jogo digital 3D voltado para os consoles de mesa atuais considerando as limitações impostas pela renderização em tempo real presentes nestas plataformas.

Para atingir esta meta, são necessários os seguintes objetivos específicos:

- Apontar limitações impostas;
- Planejar a produção com os recursos disponíveis;
- Descrever o processo de criação do personagem resultante.

Para cada objetivo específico serão consideradas as limitações dos consoles de forma que a implementação do personagem resulte em algo jogável em tempo real nestas plataformas.

1.2 JUSTIFICATIVA

Para justificar a construção de um personagem para um jogo 3D primeiramente é necessário demonstrar a importância econômica dos jogos digitais comerciais nos dias atuais uma vez que um personagem é uma parte do desenvolvimento deste tipo de produto.

Jogos Digitais são hoje um mercado consolidado uma vez que eles já existem comercialmente há mais de 40 anos com o advento dos primeiros consoles e expandiram-se para computadores e celulares no processo. Em 2012, as receitas do mercado global ultrapassaram 67

bilhões de dólares somente em vendas de *hardware*, software e jogos *mobile*. As vendas de conteúdos extras dentro de jogos somam mais 14,8 bilhões de dólares a este valor (MARCHAND e HENNIG-THURAU, 2013).

Comparando estes dados com outras áreas do entretenimento, o mercado de jogos possuía na época arrecadação global maior que o mercado de música ($16,5 bilhões) e o de livros ($69,4 bilhões) e era próximo ao mercado de filmes ($85 bilhões) (MARCHAND e HENNIG-THURAU, 2013). Em 2016 o mercado global de jogos teve arrecadação direta de 101,1 bilhões de dólares (MCDONALD, 2017).

A arrecadação do mercado global de jogos já é mais que duas vezes e meia maior do que a arrecadação das bilheterias de cinema em 2016 em todo o mundo ($38,6 bilhões) (MPAA, 2016).

A previsão de crescimento do mercado até 2020 é de 6,2% anualmente (MCDONALD, 2017) sendo este valor maior do que que os mercados de música (3,2%), TV e filmes (0,5%) e livros (2,9%) (TAKAHASHI, 2016). Alguns jogos possuem arrecadações maiores do que os de filmes consagrados pelo público geral. Como exemplo, o jogo *Call of Duty: Modern Warfare 3* arrecadou 1 bilhão de dólares em 16 dias, chegando mais rapidamente nestes valores em um dia a menos do que o aclamado filme Avatar do diretor James Cameron (WAUGH, 2011).

Tendo em vista estes valores e suas similaridades com o cinema, a indústria de jogos é responsável pela geração de diferentes postos de trabalho em diferentes áreas. Dentro desse ramo atuam profissionais como programadores, engenheiros, designers, artistas, atores, animadores, roteiristas, músicos, entre outros.

No âmbito social, os jogos digitais, combinados com o crescimento da internet, geraram novos tipos de interações sociais. Jogos multijogadores foram desenvolvidos e se expandiram através de comunidades locais e na internet e, com o avanço da tecnologia, jogos *online* foram criados, onde é possível conversar, compartilhar conquistas, criar conteúdo e interagir com redes sociais comuns, tudo isso com pessoas de diferentes locais e nacionalidades (MARCHAND e HENNIG-THURAU, 2013).

Hoje, todo este processo culmina com os *eSports*, representados principalmente por jogos competitivos como *League of Legends*, *Counter-Strike* e *Street Fighter*. As comunidades em torno desses jogos cresceram até ganharem público o suficiente para serem assistidos da mesma forma que esportes tradicionais sendo inclusive transmitidos pela televisão ao vivo (HOOD, 2016). *League of Legends*, por exemplo, gerou uma arrecadação 1,6 bilhões de dólares em 2015 (MUELLER, 2016).

Em conjunto com os *eSports*, diversos canais do Youtube e do Twitch cresceram transmitindo jogos digitais, gerando outro tipo de entretenimento e hoje existem diversas pessoas se sustentando com este tipo de atividade (KERR, 2015). Ao redor dos jogos houve o desenvolvimento de diversos tipos de mercados associados (MARCHAND e HENNIG-THURAU, 2013).

Outro dado que mostra a penetração dos jogos na sociedade é que nos Estados Unidos 42% da população é considerada jogadora e 4 em cada 5 casas possuem um console de videogame (CHIKHANI,2015).

Portanto, tendo em vista as relevâncias econômicas e sociais apresentadas e o crescimento da indústria que continua acontecendo, o desenvolvimento de um personagem pensando na sua adequação às limitações das tecnologias hoje empregadas é algo justificado. Além disso, a conceituação e criação através da modelagem 3D contempla os conhecimentos adquiridos no curso de Design desenvolvendo as competências do aluno.

1.3 DELIMITAÇÃO DO PROJETO

Sendo produzido dentro do curso de Design, este projeto não abordará nenhuma implementação relacionada somente à área de programação como: Inteligência Artificial, simulação de Física, Manipulação de colisões e eventos, linguagens de programação e lógica. Além disso, não haverá nenhuma mudança ou otimização no código fonte de qualquer *engine* de jogos utilizada e nem serão produzidos algoritmos que desempenhem qualquer função de uma *engine*.

Este projeto também não se preocupará com a implementação do *gameplay* do jogo uma vez que, apesar de estar associado com o *Design*, também tange a área de programação.

Apesar de serem áreas correlacionadas ao desenvolvimento de personagens dentro da produção de jogos, também não estão no escopo deste projeto o Design de Níveis (*Level Design*) e a modelagem de cenários já que são partes amplas com complexidade suficiente para serem avaliadas em outros trabalhos independentes deste. As áreas de Roteiro e Cinematografia também não serão desenvolvidas pelos mesmos motivos.

Por fim, animações faciais não serão realizadas dentro deste projeto com o intuito de permitir a viabilidade do mesmo dentro do prazo estipulado.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo abordará os aspectos que impactam a performance da renderização 3D em um jogo digital em tempo real, mostrando os fatores que um artista ou *designer* deve se preocupar ao produzir tanto personagens como cenários para este tipo de mídia. Dessa forma, haverá a compreensão de como será alcançado o objetivo proposto neste trabalho.

Primeiramente é necessário compreender a execução de um jogo em um *hardware* preparado para este tipo de processamento. Consoles são computadores e como tais possuem uma Unidade Central de Processamento (CPU) e uma Unidade de Processamento Gráfico (GPU). Estas duas partes possuem uma memória volátil dedicada chamada de memória RAM (*Random Acess Memory)* no caso da CPU e VRAM (*Video Random Acess Memory*) no caso da GPU (TRÜMPLER, 2015).

Durante a execução de um jogo a GPU é primariamente responsável por repetidamente traduzir as informações tridimensionais de geometria e texturas recebidas da CPU em uma imagem bidimensional na tela do equipamento onde o jogo é executado (GHAZI, 2012).

A GPU é capaz de realizar simples cálculos aritméticos paralelamente de maneira muito mais eficiente que uma CPU. Cálculos de posições de diferentes vértices de um objeto 3D e diferentes *pixels* em uma tela são inerentemente independentes, podendo ser computados paralelamente (BEETS, 2013).

Alguns cálculos de física também podem ser realizados por uma GPU. A empresa Nvidia, por exemplo, provê meios de aproveitar-se das características de processamento paralelo de suas GPUs para realizar cálculos como movimento da malha de tecidos, cabelo e partículas de fumaça (NORRIS, 2016).

Fora da área de jogos, mas ainda por sua capacidade de processamento paralelo, hoje em dia GPUs também são utilizadas para treinamento de redes neurais na área de inteligência artificial (SHAIKH, 2017), entre outros problemas matemáticos.

Retornando para os jogos digitais, a CPU é responsável principalmente por organizar as informações de geometria e textura e passá-las para a GPU (TRÜMPLER, 2015) além de executar algumas tarefas como detecção de colisão entre personagens, salvar e carregar o jogo, cálculos relacionados à vida do personagem, itens, Inteligência Artificial e Física (PETKOV, 2017).

Após o processamento de todos estes dados por ambas as partes, a GPU gera uma imagem estática, chamada de *frame* (quadro) que é colocada na tela (GHAZI, 2012). Durante um jogo vários destes *frames* são gerados por segundo, dando a ilusão de movimento da cena, que caracteriza a renderização em tempo real.

2.1 FATORES QUE AFETAM O DESEMPENHO

Um jogo do estilo proposto deve renderizar seus quadros em tempo real, respondendo rapidamente aos comandos executados pelo jogador. A taxa de quadros por segundo determina o quão fluida é a animação na tela. Em geral, desenvolvedores de jogos em consoles tentam atingir ou 30 ou 60 *frames* por segundo (COPELAND, 2014).

Estes valores são comuns pois é padrão na indústria que TVs ou monitores atualizem a sua tela 60 vezes por segundo (60Hz) sendo 30 e 60 divisores deste número (GREENWALD, 2014). Um número diferente de um divisor de 60 pode causar um efeito chamado de *judder*, que é quando o ser humano percebe pequenos travamentos na imagem resultantes da repetição de frames ser inconstante na tela, este efeito é comum em filmes que rodam a 24 *frames* por segundo (ROUSE, 2010).

Para alcançar a fluidez, os artistas e *designers* devem se preocupar com alguns aspectos que impactam no desempenho, estes fatores serão discutidos nos tópicos a seguir.

2.1.1 *Draw Calls*

Segundo TRÜMPLER (2015) *Draw Call* é um comando da CPU para a GPU renderizar uma malha (*mesh*) na tela. A GPU, já dispondo das geometrias, texturas e estados da malha em sua memória interna (VRAM), recebe a mensagem e utiliza as informações que dispõe para gerar os pixels correspondentes àquela geometria na tela. Uma *Draw Call* é necessária para cada material em uma geometria, outras fontes como sombras, efeitos e tipos diferentes de iluminação que recaem sobre a mesma geometria geram *Draw Calls* adicionais (TRUMPLER, 2015).

Este processo deve ser considerado durante a produção de um jogo pois é custoso computacionalmente, uma vez que a GPU é capaz de renderizar pequenas malhas muito mais rapidamente do que a CPU é capaz de mandar instruções para renderizá-las. (TRÜMPLER, 2015). Portanto, renderizar diversos pequenos conjuntos de triângulos é mais

custoso computacionalmente do que renderizar grandes conjuntos dos mesmos de uma única vez.

O manual da *engine* Unity3D, por exemplo, recomenda pelo menos 1500 triângulos por malha (UNITY USER MANUAL, 2017.2) enquanto a *engine* Unreal recomenda pelo menos 300 triângulos por malha (UNREAL ENGINE, 2017).

Tendo em vista o custo computacional, parte do desenvolvimento de jogos é otimizar o número de *Draw Calls*. Cada *engine* de jogos implementa certos tipos de otimização para economizar no número de *Draw Calls*, mas de maneira geral é necessário pelo menos uma *Draw Call* para cada material em uma geometria (CRYENGINE, 2016; UNITY USER MANUAL, 2017.2; UNREAL ENGINE, 2017). O manual da *engine* de jogos CryEngine, por exemplo, explica que em seu *software* cada malha com um material necessita de dois *Draw Calls* sendo que sombras e iluminações diferentes se somam a este valor (CRYENGINE, 2016).

No caso da CryEngine, há a recomendação que o número de *Draw Calls* para os consoles Playstation 3 e Xbox 360 não ultrapassem o número de dois mil por *frame* (CRYENGINE, 2016). Michaud (2015) aponta que fabricantes de placas gráficas estimam que com computadores e consoles de ponta na época da publicação (2015) é possível chegar a um número entre dez mil e vinte mil *Draw Calls* por *frame* caso a otimização seja bem trabalhada. Em contraposição com os fabricantes, Gesota (2016) aponta um número entre quinhentas e cinco mil *Draw Calls* por *frame* no atual estágio de desenvolvimento da maioria dos jogos em computadores considerados modernos na data do artigo (2016).

2.1.2 Geometria

Existem limites para a manipulação e renderização de vértices em tempo real pela GPU. Isto significa que o artista ou *designer* deve considerar o número de polígonos e vértices ao produzir seu personagens, objetos e terrenos. O manual da *engine* Unity, por exemplo, sugere ao desenvolvedor de jogos não ultrapassar cem mil vértices totais para um jogo *mobile*. Um computador considerado moderno no tempo corrente é capaz de lidar com milhões de vértices; contudo, é prudente manter este número sob controle (UNITY USER MANUAL, 2017.2). Ainda segundo UNITY USER MANUAL (2017.2) é sugerido manter o número de vértices abaixo de três milhões em um jogo de computador.

Para os consoles, CRYENGINE (2016) sugere não ultrapassar 1 milhão de triângulos em uma cena completa para o Xbox 360 atingir 60

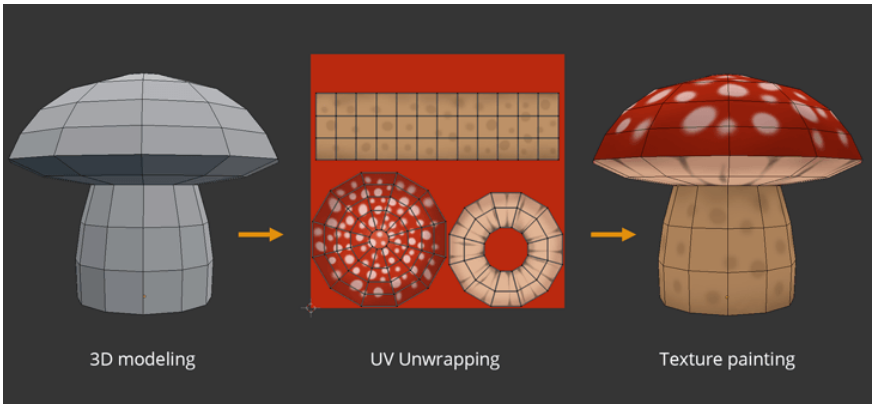
frames por segundo. Consoles e computadores considerados modernos no presente tempo deste trabalho (2017) são capazes de gerar mais triângulos por *frame* para manter a mesma taxa de quadros por segundo. A quantidade de triângulos e vértices por *frame* também depende da taxa de quadros por segundo que se deseja alcançar, uma vez que a 30 *frames* por segundo a geometria será renderizada menos vezes por segundo na tela do que se estivesse a 60 *frames* por segundo e, portanto, podem-se utilizar mais polígonos e vértices.

Manuais das *engines* apontam também para a preocupação com os UV *Maps* que são mapas bidimensionais que são aplicados sobre um modelo tridimensional e onde cada ponto no mapa 2D corresponde a um ponto na malha 3D, possibilitando que uma imagem bidimensional possa ser utilizada como textura do objeto tridimensional. Divisões de mapas na mesma malha são chamados de costuras (*seams*) (JAMES, 2012).

No centro da Figura 1 há um exemplo de UV *Map*. A malha do cogumelo é dividida em várias partes onde são aplicadas texturas diferentes a partir de uma única imagem bidimensional. As regiões de contato entre cada textura são as costuras.

.

Figura 1 – Exemplo de UV Map.



Fonte: Muttaquien (2017).

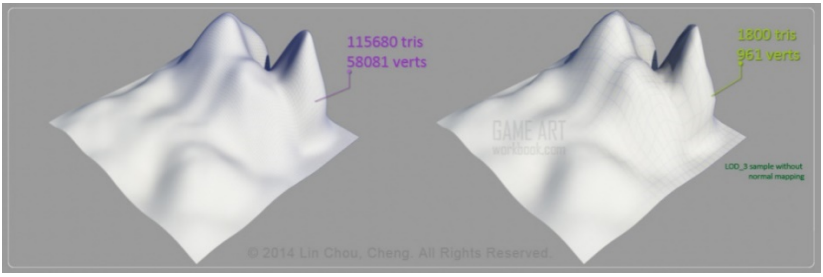
As três *engines* estudadas sugerem criar a menor quantidade possível de costuras em um UV *Map*. Nas *engines* os vértices nas costuras são duplicados para posterior processamento, aumentando o número de vértices total, diminuindo o desempenho (CRYENGINE, 2016; UNITY, 2017; UNREAL ENGINE, 2017).

Outras técnicas de otimização relacionadas à geometria surgem devido à natureza dos jogos 3D simularem espaços tridimensionais em uma tela bidimensional. Na natureza os seres humanos enxergam menos detalhes em objetos distantes do que próximos. Da mesma forma, objetos simulados como distantes do observador em um jogo podem possuir menos detalhes.

Algumas técnicas citadas por todos os manuais para redução dos detalhes são *Level of Detail* (LOD) e *Culling*. LOD se refere a uma técnica que consiste em reduzir a complexidade da geometria e texturas de um objeto quando este se encontra a uma certa distância simulada do observador (CHENG, 2015).

A Figura 2 ilustra esta técnica, o terreno da esquerda possui 115680 triângulos e 58081 vértices enquanto o terreno da esquerda possui 1800 triângulos e 961 vértices, a diferença visual entre os dois é praticamente imperceptível devido à distância simulada do observador com o terreno.

Figura 2 – Exemplo de LOD.



Fonte: Cheng (2015).

A técnica de *Culling* é a rejeição de um objeto de qualquer tipo que não contribui com a renderização da imagem final (CRYENGINE, 2016). O objeto pode estar, por exemplo, escondido atrás de outros objetos do terreno ou ainda estar a uma distância simulada muito grande do observador. Neste caso, a técnica de *Culling* retira o processamento de toda a geometria do objeto fazendo com que ele efetivamente não produza *Draw Calls* e que seus vértices e polígonos não sejam computados pela GPU (CRYENGINE, 2016).

A Figura 3 mostra uma tabela retirada do manual da *engine* CryEngine exemplificando uma distribuição comum de triângulos e *Draw Calls* em uma cena de forma que a mesma fique dentro das recomendações deles para os consoles Playstation 3 e Xbox 360.

Figura 3 – Exemplo de distribuição de triângulos e *Draw Calls* em uma cena.

Object	Triangles	Drawcalls
Particles	2000	100
Decals	20000	80
Roads	30000	120
Veg	40000	200
Terrain	50000	80
FP Character	75000	40
Entities	300000	320
Brushes	600000	1000
TOTAL	**1042000**	**1940**

Fonte: CRYENGINE (2016).

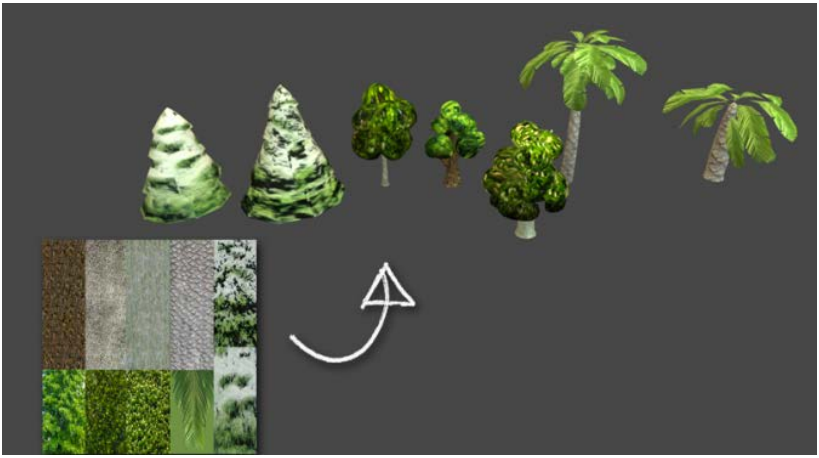
2.1.3 Texturas

As texturas de uma geometria são outro fator que impactam a performance, cada *engine* possui estratégias para a compressão e geração de texturas de baixa resolução. A *engine* Unity 3D possui a opção de compressão ao importá-las para a *engine*, além de possuir uma função chamada de *Texture Mipmaps* que habilita texturas em baixa resolução para triângulos menores no modelo (UNITY USER MANUAL, 2017.2).

Segundo Jie et al. (2011) os mapas de texturas devem ser retângulos ou quadrados e preferencialmente ter resoluções das laterais proporcionais a potências de 2, como 4, 256, 1024, etc. Caso o tamanho não seja uma potência de 2, o custo computacional será levemente superior. Não se recomenda o tamanho de uma lateral ser maior do que 2048 pixels (JIE et al., 2011; CRYENGINE, 2016).

Para aumentar a performance, pode-se utilizar algo chamado de Atlas de Textura, que é uma coleção de diferentes texturas em uma única imagem, permitindo que a GPU possa renderizar objetos separados, mas com a mesma textura, em uma mesma *Draw Call* (SCIUTTERI, 2016). Um exemplo da aplicação desta técnica pode ser visto na Figura 4. Com apenas uma imagem de textura é possível em uma única *Draw Call* renderizar a textura de todas as árvores diferentes.

Figura 4 – Exemplo de Atlas de Textura.



Fonte: Hogan (2014).

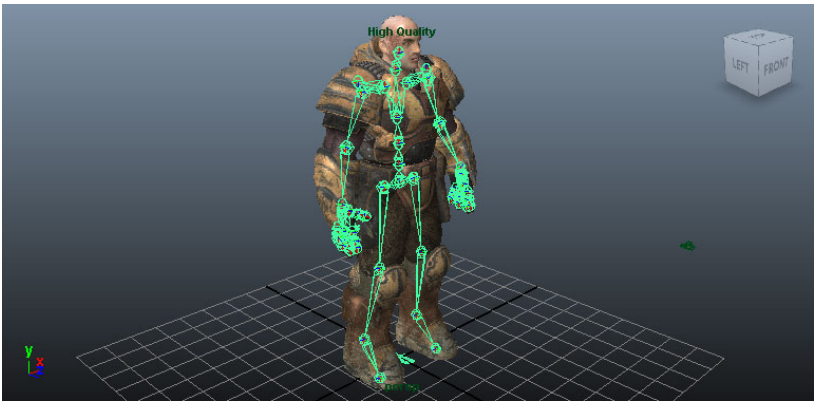
Há, contudo, desvantagens ao utilizar-se de Atlas de Texturas. Gerar e replicar um mesmo padrão de textura em uma superfície (*texture tiling*) não é possível com este método. Além disso, como os objetos diferentes com a mesma textura são tratados como um só é necessário ser cuidadoso ao escolher quais objetos agrupar no Atlas, uma vez que é possível que a GPU carregue texturas que não estarão na cena caso o Atlas espalhe textura em objetos presentes em uma área muito grande (IVANOV, 2006).

2.1.4 *Skinned Meshes*

Outro fator relacionado à performance de um jogo são as malhas que possuem uma estrutura chamada de ossos ou *bones* que são utilizados para a animação dos vértices, chamados de *Skinned Meshes* (UNITY 3D, 2017). Um esqueleto de animação é composto de diversos *bones*, conectados através de uma estrutura hierárquica, sendo os pontos de conexão entre cada um dos *bones* chamados de juntas que possuem uma posição e rotação no espaço 3D (VENUTA, 2014). As juntas estão conectadas a um ou mais vértices da malha do personagem ou objeto que será animado. Ao movimentar-se uma junta os vértices da malha influenciados por ela também se movimentam de acordo (VENUTA, 2014).

A Figura 5 mostra um exemplo de personagem modelado com seu esqueleto de animação. Os triângulos verdes internos ao personagem são seus bones e os círculos são as juntas. Cada junta está conectada a um ou mais vértices da malha do personagem possibilitando a animação do mesmo.

Figura 5 – Personagem modelado com seu esqueleto de animação.



Fonte: Simpson (2014).

Ressalta-se que apesar de parecer uma estrutura replicando os ossos de um ser humano, não necessariamente os *bones* representam ossos reais, podendo ser utilizados inclusive para animar o rosto ou cabelo

de algum personagem ou ainda qualquer outro objeto que possua animação sendo sua contraparte real inanimada ou não.

No caso da *engine* Unity 3D, sugere-se a utilização de algo entre quinze e sessenta *bones* para um personagem, sendo trinta um valor considerado bom para um personagem de jogo de computador (UNITY USER MANUAL, 2017.2). Um jogo *mobile* deve possuir menos que trinta *bones* por personagem. Além disso, um vértice não deve ser influenciado por mais do que quatro *bones* (UNITY USER MANUAL, 2017.2).

Além dos fatores apresentados até aqui, diversos outros elementos causam impacto no desempenho de um jogo digital, tais como a simulação de física, inteligência artificial, entre outros. Estas áreas, no entanto, não estão relacionadas ao artista ou ao *Designer* e, portanto, estão fora do escopo deste projeto.

2.2 EXEMPLOS DE JOGOS DESENVOLVIDOS NA INDÚSTRIA

Dados os principais pontos que afetam o desempenho de um jogo tendo em vista a perspectiva do artista, é possível agora observar alguns jogos produzidos pela indústria de forma que se tenha ideia dos parâmetros e recursos necessários para a produção de um jogo de alta complexidade. São considerados jogos de alta complexidade ou triple A aqueles cujo orçamento ultrapassa 1 milhão de dólares e são planejados para vender acima de um milhão de cópias para terem um lucro várias vezes acima do custo de produção (KOTAKU, 2014; SCHULTZ, 2017).

O site Kotaku em um artigo de 2014 mostra o custo de diversos jogos considerados *triple* A durante a história dos videogames. Entre os anos de 1990 e 2000 os jogos mais complexos custavam em média entre um e quatro milhões de dólares para produzir, com algumas exceções como o jogo Shenmue (1999) que custou quarenta e sete milhões de dólares (KOTAKU, 2014).

Entre os anos 2000 e 2010 o custo para a produção de jogos desse porte aumentou de cerca de quatro milhões para valores entre dez e trinta milhões de dólares, alguns jogos dessa época como *Call of Duty: Modern Warfare 2* (2009) e *World of Warcraft* (2004) chegaram a custar duzentos milhões de dólares. Após 2010 jogos com custo entre trinta e cem milhões de dólares são comuns, com jogos custando acima de cem milhões de dólares aparecendo esporadicamente (KOTAKU, 2014).

Após avaliar os aspectos de custo, pode-se inferir o que é possível realizar neste projeto dado que ele possui menos recursos que uma grande

softhouse. Para isso é necessário também visualizar algumas características de produção de alguns destes jogos.

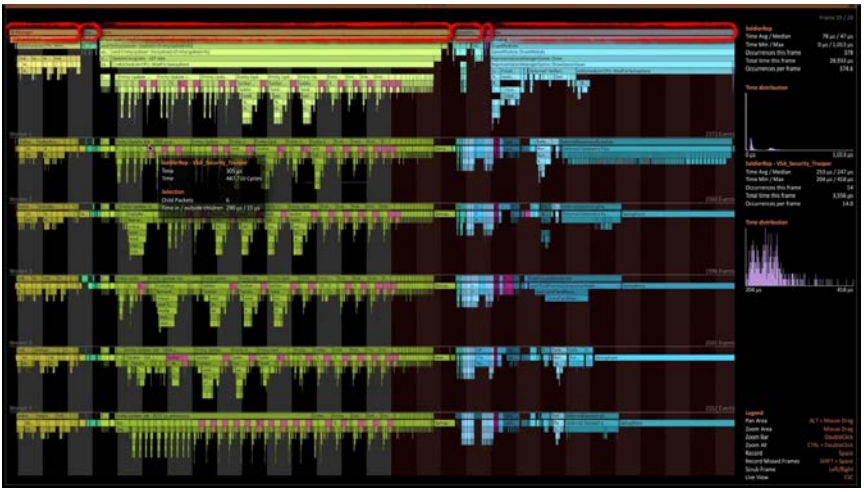
Um primeiro exemplo é o jogo *Killzone Shadow Fall* que foi um dos jogos de lançamento do Playstation 4, produzido pela desenvolvedora Guerrilla Games. Em fevereiro de 2013 a sua equipe possuía cento e cinquenta integrantes e o ciclo de produção era estimado em dois anos e meio (CIFALDI, 2013). Em 2013 houve uma apresentação feita por Michal Valient, um dos líderes do desenvolvimento do jogo, mostrando os feitos técnicos alcançados no desenvolvimento do mesmo (VALIENT, 2013). Os dados apresentados a seguir serão desta apresentação.

Um personagem de *Killzone Shadow Fall* possuía aproximadamente quarenta mil polígonos em seu maior LOD, havendo mais seis níveis de detalhe com menor número de polígonos. Isso representa um aumento de 400% no número de polígonos com relação ao último jogo da desenvolvedora o Playstation 3. Além disso, o personagem possuía seis mapas de textura quadradas com 2048 pixels de lado, sendo as texturas também quatro vezes maior do que o jogo no Playstation 3. Vários vértices dos personagens eram influenciados por oito juntas, que é duas vezes o valor recomendado pela *engine* Unity3D, por exemplo.

O jogo também apresentou um aumento no uso de iluminação volumétrica (referir-se ao item 2.1.4), com praticamente todas as luzes possuindo esta capacidade. Apresentou-se também mapas de reflexos e efeitos de profundidade de campo utilizados em larga escala.

Com relação à carga sobre a CPU, a demo do jogo possuía sessenta personagens com inteligência artificial, cerca de oito mil e duzentos objetos que interagiam com a física, além de quinhentos sistemas de partículas, entre diversos outros aspectos. A Figura 6 mostra a ferramenta de CPU Profiler criada pelos desenvolvedores para acompanhar os tempos de execução de diversas tarefas do processador durante um *frame*. É possível ver uma grande quantidade de tarefas executadas pela CPU, muitas em paralelo, mostrando um elevado grau de complexidade na execução deste *frame*.

Figura 6 – CPU *profiler* de *frame* demonstração do jogo *Killzone Shadow Fall*.



Fonte: Valient (2013).

Com todas essas características, a versão de demonstração apresentada funcionava no console Playstation 4 a uma taxa de 30 *frames* por segundo em uma resolução de 1920x1080 pixels. A Figura 7 mostra o exemplo de um *frame* do jogo para que se possa visualizar o tipo de produção que está sendo tratada.

Figura 7 – Tela do jogo Killzone Shadow Fall.

Fonte: Owen (2013).

Para efeito de comparação, o jogo *Ryse: Son of Rome*, foi apresentado também em 2013, mas para o console Xbox One, feito pela desenvolvedora Crytek através da CryEngine. Um personagem é feito de mais de 150 mil triângulos no seu maior LOD, com mais de 770 juntas (referir-se ao item 2.1.5), sendo 260 faciais (EVANS, 2016), acima do recomendado por qualquer *engine* de jogos. Para que seja colocado em perspectiva, o último jogo desenvolvido pela Crytek, *Crysis 3* um personagem possuía 60 mil triângulos (NELVA, 2014). No caso de *Ryse* os desenvolvedores também criaram um sistema próprio de animação de destruição de objetos, podendo criar cenas com grande quantidade de movimento sem um custo computacional tão elevado (EVANS, 2016).

Observando alguns outros jogos, *The Order 1886*, jogo desenvolvido pela produtora *Ready at Dawn* para o console Playstation 4, possuía o personagem principal com mais de 100 mil polígonos e 250 juntas apenas na cabeça (NELVA, 2014).

Na desenvolvedora Square Enix também pode-se perceber números semelhantes no desenvolvimento dos seus personagens. Segundo Romano (2014) o jogo Final Fantasy XV para Playstation 4 e Xbox One possui cerca de 5 milhões de polígonos renderizados por *frame* sendo cerca de 100 mil por personagem sendo 20 mil apenas no cabelo dos mesmos, aproximadamente cinco vezes o número de polígonos em

jogos feitos pela mesma produtora em jogos semelhantes nos consoles da geração passada, Playstation 3 e Xbox 360. Cada personagem possui cerca de 600 *bones*, um número entre dez e doze vezes superior aos seus jogos da geração passada. A Figura 8 mostra uma tela do jogo para que se tenha referência da qualidade visual alcançada.

Figura 8 – Tela do jogo Final Fantasy XV.

Fonte: Caswell (2016).

Em outros jogos da mesma produtora é possível ver que o número de triângulos almejado é quase o mesmo. Em *NieR: Automata*, lançado no começo de 2017. O número de triângulos da personagem principal é cerca de 100 mil (MATSUDAIRA, 2017). A Figura 9 mostra o modelo da personagem principal para que se tenha uma ideia da qualidade alcançada.

Figura 9 – Modelo final da protagonista do jogo *NieR: Automata.*

Fonte: Matsudaira (2017).

Como um último exemplo temos o jogo *Horizon Zero Dawn* feito pela produtora *Guerrilla Games* (mesma de *Killzone Shadow Fall*) para Playstation 4, também lançado no início de 2017. A protagonista possui mais de 100 mil triângulos apenas na malha do cabelo (LITHVALL, 2017) e um inimigo do jogo chamado de *Thunderjaw* foi feito com cerca de 550 mil triângulos (FREDERIKSEN, 2015). A Figura 10 mostra parte do modelo 3D da protagonista do jogo.

Figura 10 – Modelo 3D da protagonista Aloy do jogo Horizon Zero Dawn.

Fonte: Lithvall (2017).

Concluindo este capítulo, verifica-se que a otimização da produção de um jogo é uma alocação de recursos, que devem ser equilibrados de forma a chegar em um visual e performance desejados. Os recursos financeiros e humanos também devem ser considerados na criação de um jogo já que foi observado que existem empresas que desenvolvem jogos com mais de 150 pessoas em seu *staff* com tempos de produção acima dos dois anos e meio, provando ser limitada a capacidade de reprodução desses *softwares* em escalas menores.

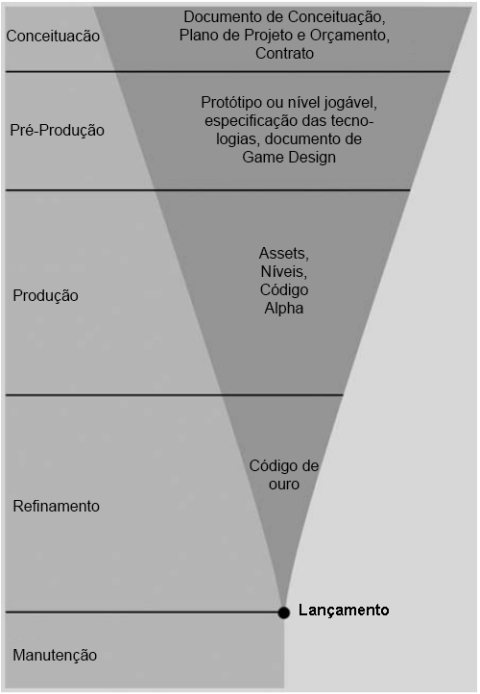
3 METODOLOGIA

Este capítulo tratará dos estágios do processo de produção de jogos que serão utilizados neste projeto, de forma a cumprir o objetivo de produzir um personagem 3D para um jogo digital. Será apresentado um método de produção comum na indústria juntamente com a explicação de como este trabalho utilizará estas etapas.

3.1 PROCESSO DE PRODUÇÃO DE JOGOS

O processo de produção estudado é descrito por Fullerton et al. (2008) no livro Game Design Workshop e consiste em 5 fases de produção de um jogo. A Figura 11 ilustra cada uma das etapas colocando algumas das principais atividades que são desenvolvidas em cada fase.

Figura 11 – Etapas do desenvolvimento de um jogo.



Fonte: Fullerton et al. (2008, p. 376).

Cada etapa busca alcançar metas específicas visando o jogo final. Este projeto não se utilizará de todas elas uma vez que o objetivo principal é a conclusão somente de um personagem. As etapas são descritas abaixo em ordem de execução.

1) **Conceituação** (*Concept Phase*): nesta etapa é criada a ideia e os conceitos iniciais do jogo, além de todo planejamento financeiro, viabilidade e plano do projeto. Em geral este processo deve ser feito em um mês (FULLERTON et al., 2008, p. 376).

2) **Pré-Produção** (*Pre-Production Phase*): é criado um protótipo ou um nível jogável que servirá como base para a produção final. Nesta etapa é possível determinar quais serão as principais dificuldades na implementação das tecnologias e características do jogo e descartar qualquer elemento que não irá funcionará ou não se enquadrará nos recursos propostos incialmente. Em várias produções é nesta etapa que é escrito o documento de Game Design que funciona como um guia para a equipe de desenvolvimento. Esta etapa geralmente dura cerca de 5 meses (FULLERTON et al., 2008, p. 378).

3) **Produção** (*Production*): Com base no protótipo e documento de Game Design criados, é implementado o *gameplay* completo e são criadas as versões finais dos objetos, tecnologias, níveis, personagens, som e todos os outros elementos necessários para que o projeto se torne o jogo idealizado inicialmente. Esta etapa é geralmente a mais longa do processo de produção podendo durar 15 meses ou mais (FULLERTON et al., 2008, p. 379).

4) **Refinamento** (*Quality Assurance* ou *Polish*): Também chamada de *QA Phase*, corresponde à etapa onde o jogo passará por testes intensivos para determinar e corrigir problemas na programação de forma que se chegue a um produto final. Esta etapa apenas lapida as características já implementadas anteriormente na fase de Produção, não se deve adicionar novos elementos ao jogo. Dura cerca de 2 meses (FULLERTON et al., 2008, p. 379).

5) **Manutenção** (*Maintenance*): Jogos desenvolvidos para as plataformas no presente tempo de execução deste trabalho podem ser atualizados via *download* de conteúdo, dessa forma, muitos jogos se mantêm nesta etapa após serem lançados, onde é gerado conteúdo adicional e corrigido problemas que possam ter passado pela etapa de Refinamento (FULLERTON et al., 2008, p. 382).

Este projeto se preocupará apenas com as três etapas iniciais, a Conceituação, Pré-Produção e Produção uma vez que o objetivo é produzir um personagem completo, mas não o jogo finalizado. Nos itens posteriores serão explicados os métodos internos de cada etapa para chegar-se ao personagem.

3.1.1 Conceituação

Nesta etapa, o foco é dado para a criação e imersão na ideia do jogo. No caso deste projeto, a história e mecânica do jogo já existirão e o elemento que será criado é um personagem dentro deste universo. Segundo Fullerton et al. (2008, p. 148) as seguintes etapas são realizadas:

1) **Criação da ideia:** geralmente gerada através de uma pesquisa ou um método criativo. Um exemplo de um método criativo é o *Brainstorm*, onde um conjunto de pessoas colocam livremente suas ideias por um período de tempo (geralmente até uma hora), criando um ambiente propício para a criatividade. O objetivo nesta etapa é gerar uma grande quantidade de ideias e não necessariamente que todas sejam de qualidade (FULLERTON et al., 2008, p. 150).
2) **Edição e refinamento:** são avaliadas, combinadas e descartadas ideias até a geração de uma ideia final. Devem ser consideradas para a avaliação as viabilidades técnicas das ideias, os recursos disponíveis para a produção no jogo e se há uma oportunidade de mercado para a ideia (FULLERTON et al., 2008, p.156).
3) **Transformação em jogo:** Com a ideia final, determina-se o gênero do jogo, se será um jogo de esporte, luta, aventura, RPG, etc. Além disso, são estabelecidas todas as mecânicas, regras, objetivos e ações dos jogadores (FULLERTON et al., 2008, p.

162)**.** Nesta etapa sugere-se também a criação de um *Storyboard* para a visualização das mecânicas desenvolvidas de forma a ser possível ter uma ideia inicial de como um jogador interagirá com o jogo (FULLERTON et al., 2008, p.168).

3.1.2 Pré-Produção

Após a etapa de Conceituação, um protótipo funcional deve ser gerado, recriando de maneira aproximada a arte, o som, os modelos, objetos e as mecânicas idealizadas na fase anterior. Segundo Fullerton et al. (2008, p. 175) um protótipo pode ser físico, visual, digital ou uma combinação de diferentes técnicas.

Neste projeto o personagem será criado de forma a ser implementado em um protótipo digital. Segundo Fullerton et al. (2008, p. 213) o protótipo digital deve se preocupar com os seguintes elementos:

- **Mecânicas:** São características discretas de implementação de gameplay, por exemplo, o jogo Spore possuía uma mecânica onde o jogador pode criar sua própria criatura, esta foi devidamente criada e estabelecida na etapa de prototipação (FULLERTON et al., 2008, p. 214).
- **Estética:** É o *Design* visual do jogo, incluindo animações, cores e modelos dos personagens, em um protótipo são utilizadas Artes Conceituais, *Storyboards*, *Animatic* e uma versão inicial da interface para criar a impressão estética inicial do jogo (FULLERTON et al., 2008, p. 216).
- **Cinestética:** representam a “sensação” do jogo. Frequentemente está relacionada à forma como os usuários percebem o jogo ao interagir com o controle, se utilizam mouse e teclado, ou um controle de console e também como são as respostas ao apertar um botão de qualquer uma dessas interfaces físicas (FULLERTON et al., 2008, p. 217).
- **Tecnologia:** Modelos de tecnologias que serão implementados no jogo final, como Inteligência Artificial, física ou qualquer outro problema tecnológico a ser criado (FULLERTON et al., 2008, p. 219).

Para a criação do personagem deste trabalho, o foco será na criação da estética do personagem, se preocupando com as áreas relacionadas ao *Design*, as outras áreas não serão tratadas.

Nesta etapa também são escolhidas as ferramentas que serão utilizadas para a construção do protótipo. Segundo Fullerton et al. (2008, p. 238) é possível utilizar uma Linguagem de Programação, uma *Engine* de jogos ou ainda um Editor de Níveis. Todas estas ferramentas dão a possibilidade de implementação de um protótipo completo.

3.1.3 Produção

A etapa de Produção é onde todos os objetos e personagens são elaborados até chegar à sua forma funcional e os programadores criam diversas versões incrementais do código do jogo até alcançar o chamado "código Alpha", onde todas as funcionalidades estarão implementadas e funcionando como foram concebidas no documento de *Game Design* produzido na etapa anterior (FULLERTON et al., 2008, p. 379).

O código Alpha não é a versão final na construção da programação do jogo, ele deve ser polido na etapa posterior, a de Refinamento, de forma que todos os *bugs* sejam eliminados e os controles do jogo sejam ajustados (FULLERTON et al., 2008, p. 379).

Da mesma forma, os personagens, arte e níveis do jogo não são completamente polidos nesta etapa, mas devem chegar o mais próximo possível da versão final descrita pelo documento de *Game Design* (FULLERTON et al., 2008, p. 379).

Este projeto irá chegar somente até esta etapa, procurando criar o modelo, texturas e animações o mais próximo possível da versão final, não utilizando, contudo, um documento de *Game Design*, uma vez que não irá descrever um jogo completo, mas apenas um personagem. Além disso, uma extensa documentação é utilizada mais comumente quando há uma equipe trabalhando no jogo.

As etapas de Refinamento e Manutenção não são utilizadas neste projeto e não serão descritas com mais detalhes neste documento. Para um aprofundamento sobre todas estas etapas é possível acessar o livro *Game Design Workshop de Tracy Fullerton* utilizado como base para este capítulo.

4. DESENVOLVIMENTO

Neste capítulo será apresentado como foi o desenvolvimento do personagem proposto neste trabalho, mostrando cada etapa do processo de criação, além de apontar soluções e dificuldades encontradas.

Este projeto utilizou parte da metodologia do desenvolvimento de jogos para a criação de um personagem 3D para um jogo do gênero de luta para consoles de jogos digitais. Foi feito inicialmente o processo de Conceituação do personagem, seguido de seu planejamento, para no final, modelá-lo, animá-lo e implementá-lo em uma *engine* de jogos. Todos os passos foram feitos levando em conta as limitações das plataformas selecionadas para produzir o jogo.

É importante explicitar que este projeto foi feito com a participação de mais uma aluna, Nayara de Souza Braga, que auxiliou na criação da ideia da história do jogo, além de ajudar a planejar a captura de movimentos que foi realizada para criar as animações do personagem.

Nos próximos itens serão descritos os passos do desenvolvimento desse personagem desde a criação de sua ideia até a implementação na *engine* de jogos Unreal Engine 4.

4.1 CONCEITUAÇÃO DO PERSONAGEM

4.1.1 Ideia e Roteiro

A primeira etapa descrita na metodologia é a Conceituação onde primeiramente deve-se gerar as ideias, neste caso tanto do jogo, como a do personagem. Para este projeto, o primeiro passo foi definir que o jogo seria do gênero luta.

Tendo em vista o tempo limitado para a execução deste projeto, estipulou-se o jogo seguiria uma linha mais cômica e não realista, se baseando em jogos como Street Fighter, Tekken e Super Smash Bros. Isto facilita a etapa de animação deste personagem, já que não há a necessidade de elaborar movimentos utilizados em lutas reais. A criação de animações dessa forma acarretaria em uma pesquisa adicional sobre estilos de luta, além de lutadores experientes para a fase de captura de movimento que será descrita posteriormente.

Após estas definições, foi construído um roteiro que serviu como base para a concepção do personagem. Para a elaboração do mesmo foi utilizada a técnica de *brainstorm* em conjunto com a aluna Nayara de Souza Braga. Como descrito no item 3.1.1 deste documento, o *brainstorm*

é um método onde as ideias são colocadas livremente para gerar um ambiente propício à criatividade.

Várias ideias de jogo foram geradas, desde um jogo de luta entre diferentes profissões, até jogos de luta entre moradores de diferentes regiões brasileiras. Todas as ideias foram colocadas em um papel e foi selecionada a ideia que ambas as partes consideraram mais interessante.

Sendo assim, no roteiro final do jogo, os personagens são representantes de bandas e estilos musicais diferentes e eles lutam em uma arena onde o vencedor poderá alcançar o DJ do local e trocar a música para algo de seu agrado. Desta forma, cada personagem deve ser caracterizado de acordo com seu interesse musical.

O roteiro não foi elaborado além deste ponto uma vez que este não é o foco do projeto, além de ser uma tarefa complexa o suficiente para ser descrita em outros trabalhos específicos sobre o tema. A parte criada é chamada de *storyline* e é apenas um pequeno parágrafo com a descrição geral da história sendo uma de muitas etapas do desenvolvimento de um roteiro completo. É possível ver o texto gerado neste projeto no ANEXO A deste documento.

4.1.2 Perfil do personagem e Referências

Ainda seguindo a etapa de Conceituação da metodologia empregada neste trabalho, após a definição da história, o próximo passo é a criação da ideia do personagem. Para isso, buscou-se imagens de referência, além de traçar um perfil psicológico para o mesmo. Foi definido que este teria a preferência pelo estilo musical *Rock and Roll*. Não houve, entretanto, um motivo particular para a escolha deste estilo.

O objetivo da busca por imagens de referência é alimentar o repertório visual do autor, de forma que, ao gerar os desenhos dos personagens, eles sejam condizentes com a proposta inicial tanto do jogo como da preferência do estilo musical.

Na Figura 12 é possível ver algumas das imagens que foram utilizadas como base para a construção dos conceitos do personagem. Foi verificado que muitas referências traziam:

- Jaquetas de couro ou jeans
- Cintos e sapatos de couro
- Tatuagens diversas, principalmente com símbolos do *rock*.
- Símbolos como as mãos fechadas com apenas o dedo indicador e o mínimo em riste, caveiras, asas e guitarras.

- Símbolos de bandas como raio (banda AC DC) ou a boca com a língua estendida para fora (*Rolling Stones*), entre outros.
- Ornamentação com muitas pulseiras e anéis com detalhes como estrelas e pequenas pirâmides brilhantes.
- Chapéus como boinas e cartolas.

Figura 12 – Painel com referências utilizadas para criar o personagem.

Fonte: Diversas imagens encontradas no Google Images

Dessa forma, a partir das imagens utilizadas como referência, tentou-se extrair algumas características pertencentes ao estilo musical *Rock and Roll*, para que essas pudessem ser utilizadas como perfil

psicológico de criação do personagem, fazendo com que ele representasse em si o próprio estilo. Algumas características selecionadas foram:

- Alerta
- Extravagante
- Extrovertido
- Narcisista
- Insônia
- Agitado

Estas palavras serviram como base para construção do personagem juntamente com as referências visuais adquiridas. Após este processo resta então criar várias ideias de personagem e selecionar uma.

4.1.3 Desenho de Conceitos do Personagem

A etapa de conceitos do personagem seguiu as mesmas características do *brainstorm* utilizado para a criação do roteiro. Primeiramente geram-se diversas ideias do personagem e em seguida seleciona-se uma ou mais ideias consideradas interessantes. Diferente do roteiro, entretanto, tomou-se mais cuidado com a definição do personagem uma vez que a produção dele é o foco deste trabalho.

Primeiramente foram produzidos diversos rascunhos dispersos em folhas e cadernos e em seguida algumas alternativas interessantes foram digitalizadas e desenhadas com mais cuidado.

A Figura 13 mostra algumas alternativas de personagens gerados, pode-se perceber a utilização das referências na maioria deles, sendo nos acessórios, roupas, símbolos ou expressões faciais. Também buscou-se seguir as palavras características selecionadas, principalmente nas expressões utilizadas. Alguns desenhos foram feitos com o objetivo de tentar diferentes poses de uma mesma solução e um ou outro desenho foi feito com o intuito de explorar outras alternativas como o personagem com o violão no ombro na parte inferior, ou o personagem com a boina no canto superior esquerdo da imagem.

Figura 13 – *Concepts* do personagem.

Fonte: Elaborada pelo autor

O refinamento faz parte da segunda etapa da fase de Conceituação descrita na metodologia e por isso foram produzidas diferentes alternativas do mesmo personagem.

Prosseguindo nesta etapa da metodologia, escolheu-se a melhor solução, que pode ser vista na Figura 14. Neste desenho procurou-se refinar os detalhes do personagem já fazendo um conceito mais bem definido, polindo-se diversos detalhes, principalmente o rosto.

É possível ver ainda na Figura 14 a influência das referências no personagem. As ombreiras foram baseadas nos acessórios da banda Kiss e o símbolo do raio foi criado a partir do logo da banda ACDC. A calça, sapatos e cinto fazem alusão ao modo como o cantor Élvis Presley se

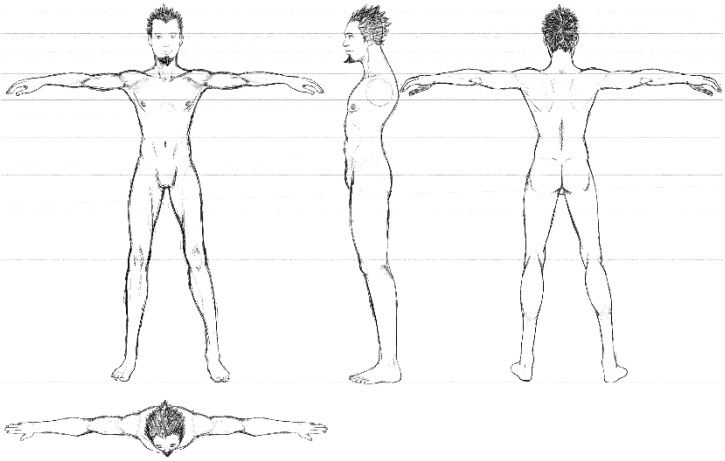
vestia. A jaqueta *jeans*, a posição dos dedos, as pulseiras, os cabelos e a própria expressão apareceram em diversas referências tanto de bandas quando de fãs deste estilo musical. O símbolo da estrela sob o olho esquerdo também é uma referência à banda Kiss.

Figura 14 – *Concept* final do personagem

Fonte: Elaborada pelo autor

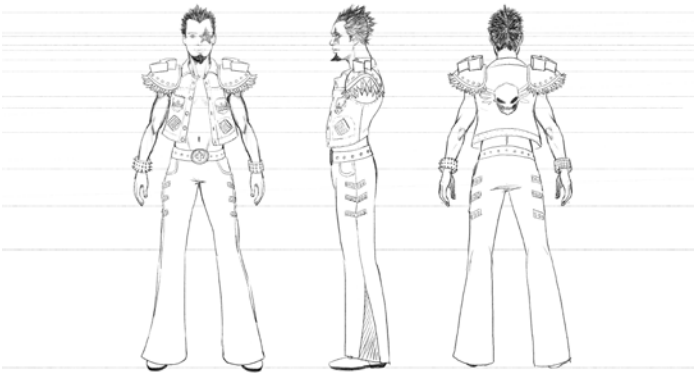
Por fim, foi criado um *Model Sheet* do personagem, mostrando suas proporções em diferentes ângulos, já preparado para a etapa seguinte, que é a modelagem 3D do mesmo. Foram gerados dois *Model Sheets* diferentes, um com o personagem sem as roupas, como se pode ver na Figura 15 e outro com as roupas, mostrado na Figura 16. O primeiro é utilizado na etapa inicial da modelagem 3D, quando é produzido o corpo e cabeça e o segundo é utilizado para a criação das roupas na etapa final.

Figura 15 – *Model Sheet* do personagem sem roupas.



Fonte: Elaborada pelo autor

Figura 16 – *Model Sheet* do personagem com roupas.



Fonte: Elaborada pelo autor

O personagem foi criado para ser realista em termos de musculatura, altura e outros componentes corporais, mas na questão das proporções, suas pernas são mais longas do que a média para um homem

e suas mãos levemente maiores. O objetivo é que estas partes fiquem mais em evidência durante o jogo já que ele utilizará estas partes para atacar.

Com isso, conclui-se a etapa de refinamento da fase de Conceituação da metodologia. Tendo em vista que o objetivo do projeto não é a criação do jogo completo, mas sim um personagem, algumas etapas da metodologia como a criação das mecânicas e regras não foram feitas.

4.2 PRÉ-PRODUÇÃO

A Pré-Produção é a segunda etapa da metodologia de produção de jogos utilizada neste projeto. Não se adentrou muito profundamente nesta fase uma vez que o objetivo é a produção de apenas um personagem e não um jogo completo. Portanto, atividades como testes de *gameplay*, interface e tecnologias não são realizadas.

Nesta etapa devem-se decidir as tecnologias que serão utilizadas para a criação do personagem. Para a modelagem 3D e construção do esqueleto de animação decidiu-se a utilização do *software* 3DS Max. Para a pintura das texturas o programa Adobe Photoshop foi definido e para a criação e refinamento das animações o aplicativo Motion Builder foi eleito.

Além disso, decidiu-se pelo uso de tecnologias de captura de movimento para a criação das animações iniciais e de escaneamento 3D para auxiliar na modelagem da face do personagem.

Não foi gerado nenhum protótipo nesta fase pois houve ênfase em avançar para o personagem completo, uma vez que não haveria tempo disponível para cumprir todas as etapas, além de ser necessário criar um ambiente para o protótipo funcionar.

Os *storyboards* de animação pertencentes a Pré-Produção foram feitos durante a Produção e são descritos mais adiante neste documento.

Dentro desta etapa, somente a parte de Estética será desenvolvida, com a criação do modelo 3D do personagem, *Storyboards* e animação do mesmo. As partes de Mecânicas, Cinestética e Tecnologia não são desenvolvidas neste projeto uma vez que estão relacionadas a *gameplay*, tecnologias na área de programação e controles, que estão fora do escopo deste trabalho.

4.3 PRODUÇÃO

Neta etapa, utiliza-se o planejamento e todas as ferramentas escolhidas nas fases anteriores para criar uma versão mais próxima da finalizada possível para o personagem. Esta é a parte mais longa deste projeto e engloba a modelagem 3D, *Rig*, Animação, Texturização e implementação na *engine* de jogos.

4.3.1 Modelagem 3D

Após a criação dos *Model Sheets* é possível iniciar a produção do modelo 3D do personagem. Antes de iniciá-lo, entretanto, é necessário estabelecer alguns limites para a criação do personagem uma vez que tanto as plataformas onde este personagem será implementado como o tempo de execução deste projeto são limitados.

No âmbito da modelagem 3D, como visto no item 2.2 deste documento, personagens para jogos desenvolvidos por grandes empresas para as plataformas atuais possuem um número variado de polígonos, com os primeiros jogos para Playstation 4 e Xbox One possuindo entre 40 e 60 mil triângulos e os jogos posteriores aumentando este número para 100 mil ou mais.

Neste trabalho almejou-se um número de cerca de cinquenta mil triângulos, dentro das especificações dos jogos iniciais destas plataformas. Ultrapassar esse número não é um problema, mas ele é um guia também para confinar o tempo de modelagem, uma vez que quanto maior a complexidade do personagem, maior o tempo gasto não só nesta etapa como nas posteriores.

Para modelar o personagem e fazer todo este projeto utilizou-se de um *notebook* com um processador Core i7 4710MQ, possuindo 16GB de memória RAM e uma placa de vídeo GeForce 860M.

Com relação à ferramenta de modelagem 3D, utilizou-se o programa 3DS Max da empresa Autodesk, uma vez que este *software* é utilizado nas aulas de graduação onde este trabalho foi realizado.

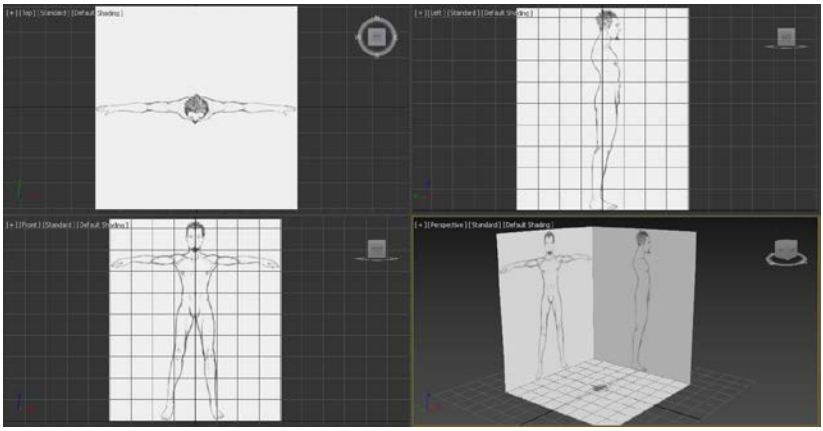
Considerou-se modelar o personagem a partir de um modelo genérico já pronto no programa MudBox, também da empresa Autodesk, entretanto, o custo de adaptação ao novo *software* e as dificuldades encontradas ao se tentar readaptar o personagem fizeram a ideia ser descartada.

A modelagem pode ser subdividida em 3 etapas, sendo a primeira o corpo, a cabeça e em seguida as roupas. Cada uma destas utilizou-se de uma técnica diferente para o processo.

4.3.1.1 Corpo

Para o corpo do personagem, utilizou-se o *model sheet* sem roupas do personagem. A imagem de cada ângulo do personagem foi colocada em cada lateral de um cubo aberto no programa 3DS Max como pode ser visto na Figura 17.

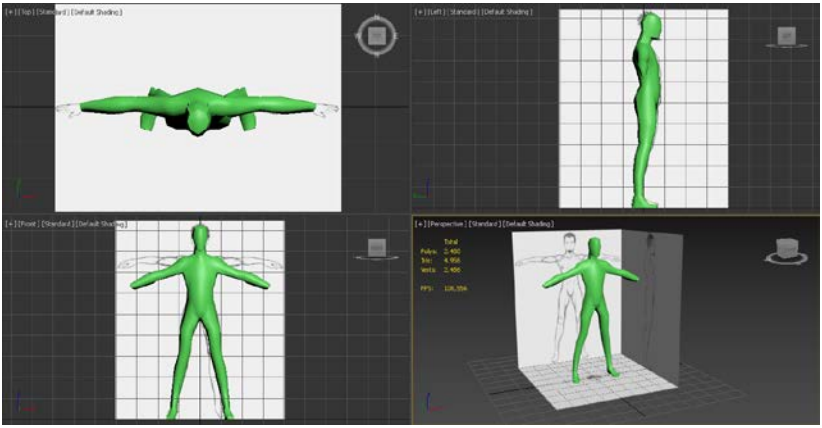
Figura 17 – *Layout* inicial no programa 3DS Max



Fonte: Elaborada pelo autor

Deste ponto em diante, o personagem foi sendo criado a partir de um cubo, seguindo cada uma das vistas dispostas no *model sheet*. Nesta etapa focou-se em primariamente criar as partes do corpo em blocos sem muitos detalhes, como pode ser visto na Figura 18, para que se tenha uma forma geral do personagem antes de avançar para algo mais minucioso. Todo o processo foi feito utilizando-se da ferramenta de simetria do 3DS Max, permitindo que apenas metade do personagem fosse modelado e a outra parte fosse simplesmente replicada pelo programa.

Figura 18 – Personagem modelado em blocos simples.



Fonte: Elaborada pelo autor

É possível ver ainda na Figura 18 que os braços do personagem foram criados na pose em “T” e posteriormente abaixados com relação ao *model sheet*. Isso foi feito para que a modelagem do torso fosse elaborada de maneira mais confortável para o autor, mas isso não é recomendado e gerou alguns problemas que serão explicados posteriormente.

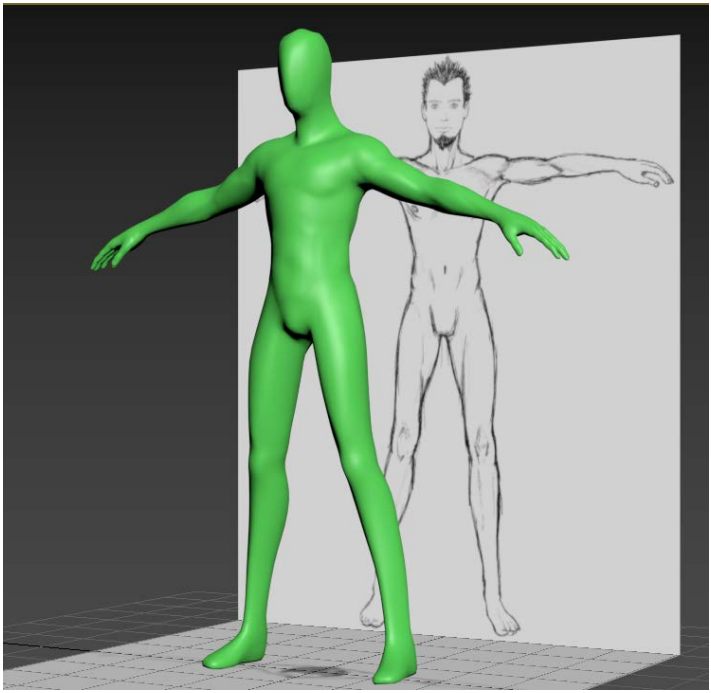
Em seguida detalhou-se a musculatura do torso, seguido dos ombros e braços e, como última etapa, a mão e os dedos foram criados. O fluxo dos polígonos foi feito seguindo a musculatura real de um corpo humano, para facilitar as deformações do personagem na etapa de animação.

Ressalta-se que cada músculo detalhado aumenta o número de polígonos da malha, e por esse fator, nem todos os músculos foram modelados. Também não houve a modelagem de nenhuma estrutura da perna ou pé uma vez que estes, diferente dos músculos do torso e braços, ficarão escondidos atrás de vestimentas fechadas.

As Figuras 19, 20 e 21 mostram as modelagens finais da região anterior, posterior e mão do personagem respectivamente. Utilizou-se também o modificador *Turbosmooth* do programa 3DS Max, que suaviza as curvas da malha do personagem com o aumento do número de triângulos como contrapartida. Aplicou-se apenas uma iteração desse modificador, que, de forma geral, quadruplica o número de triângulos a cada iteração.

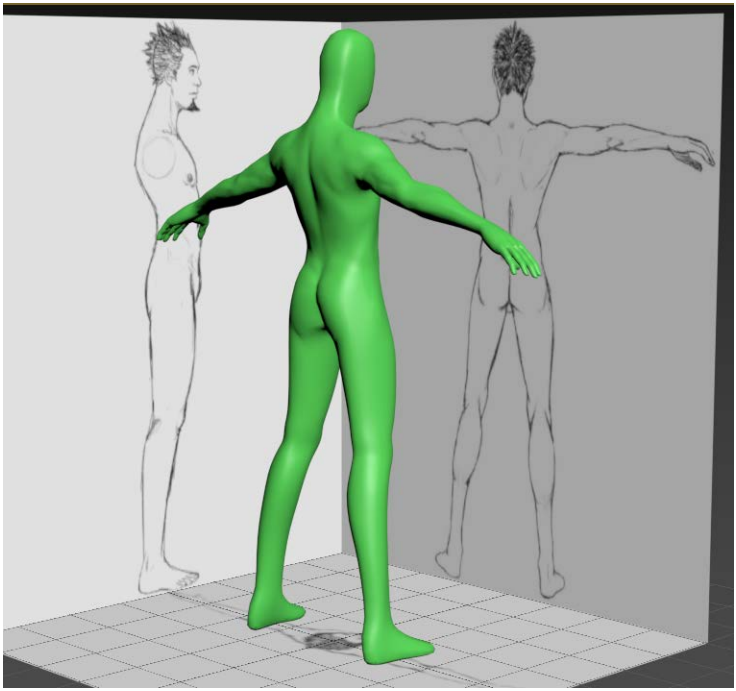
.

Figura 19 – Modelagem final da região anterior do corpo do personagem.



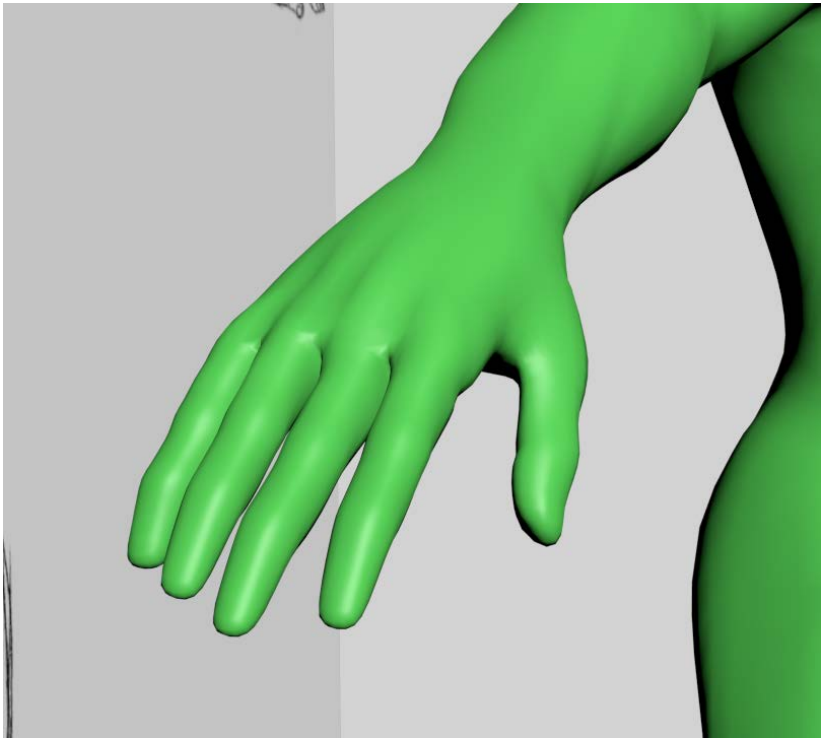
Fonte: Elaborada pelo autor

Figura 20 – Modelagem final da região posterior do corpo do personagem



Fonte: Elaborada pelo autor

Figura 21 – Detalhe da modelagem da mão do personagem.



Fonte: Elaborada pelo autor

Com isso, concluiu-se a modelagem do corpo do personagem, seguindo para a criação da cabeça onde foi utilizada uma técnica diferente de modelagem.

4.3.1.2 Cabeça

Para agilizar a construção da cabeça e também experimentar com uma outra técnica de modelagem, utilizou-se de escaneamento facial para a criação desta parte do corpo.

Este processo é feito retirando-se várias fotos em diversos ângulos da cabeça de uma pessoa e enviando as fotos para o *software* ReCap, este programa possui um algoritmo que utiliza as fotos para tentar criar um modelo 3D do objeto.

Na Figura 22 pode-se ver imagens das fotos sendo retiradas, a pessoa escolhida para ter a cabeça escaneada foi o próprio autor deste projeto. As fotos foram retiradas pelo professor Flávio Andaló, no laboratório TecMídia da Universidade Federal de Santa Catarina (UFSC), onde há uma infraestrutura para processos de captura de movimento.

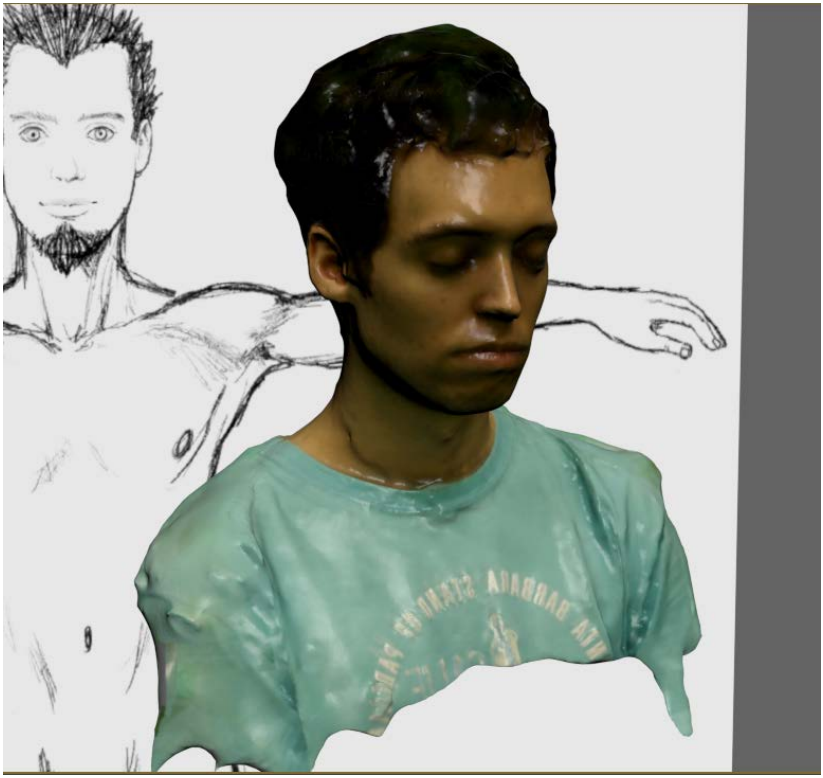
Também é possível ver refletores circulares no chão que têm o objetivo de distribuir melhor a iluminação no rosto da pessoa sendo escaneada, suavizando as sombras e ajudando no processo de criação da malha 3D pelo programa.

Figura 22 – Foto do processo de escaneamento de cabeça

Fonte: Fotos da aluna Nayara de Souza Braga

O resultado do escaneamento é um modelo tridimensional aproximado da cabeça da pessoa que pode ser visto na Figura 23. O modelo, entretanto, possui ruído e sua malha incorpora mais de quinhentos e cinquenta mil triângulos e causa um impacto perceptível de performance ao ser colocado dentro do programa 3DS Max. É possível ver os triângulos da malha na Figura 24 que mostra o resultado do escaneamento em perfil.

Figura 23 – Resultado do escaneamento 3D.

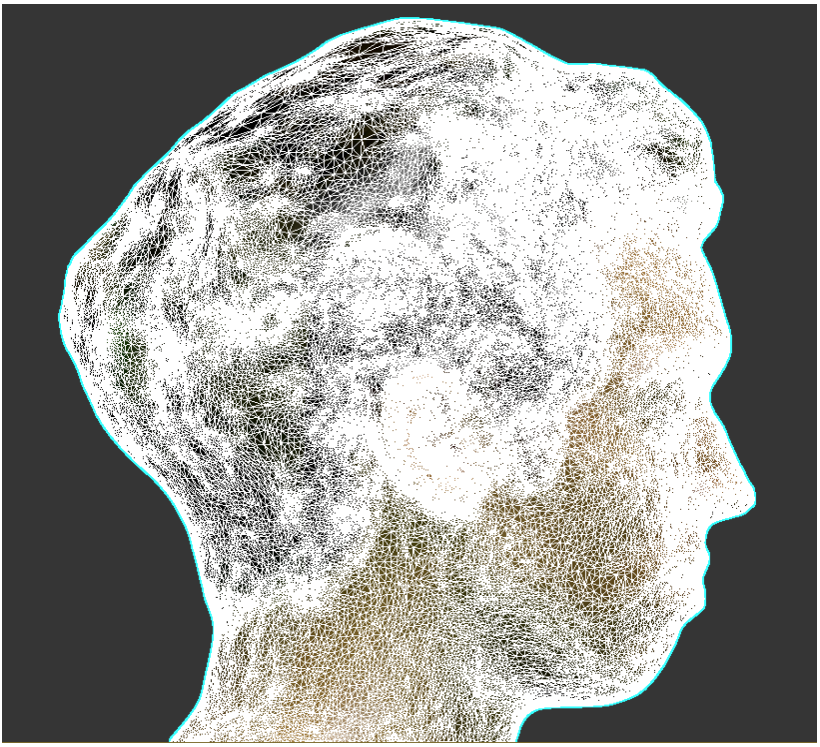


Fonte: Elaborada pelo autor

A estrutura de triângulos da malha também não serve para um personagem que vai ser animado devido ao fluxo dos mesmos não ser ideal. Dessa forma, é necessário construir uma nova cabeça com uma quantidade menor de triângulos e com um melhor fluxo de polígonos.

O processo de reconstruir a malha de um objeto pronto no âmbito da modelagem 3D é chamado de retopologia e foi aplicado nesta etapa do processo.

Figura 24 – Triângulos da malha proveniente do escaneamento

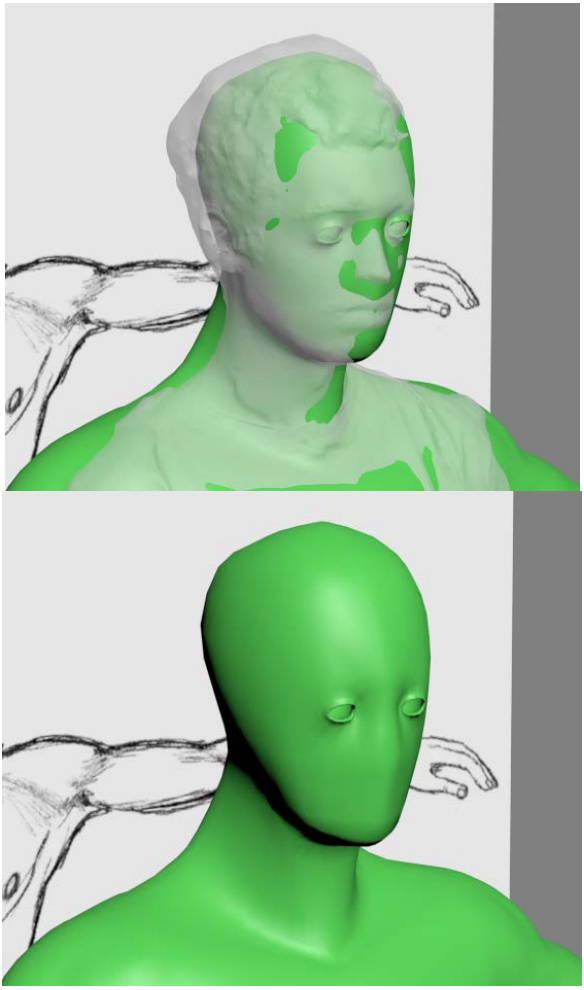


Fonte: Elaborada pelo autor

Para a retopologia, escolheu-se um lado da cabeça onde o escaneamento parecia mais similar com o rosto real da pessoa, e, em seguida, mudou-se os vértices do molde inicial da cabeça (malha das Figuras 19 e 20) para que eles ficassem posicionados sobre a malha da cabeça escaneada. O cabelo, entretanto, foi ignorado, sendo construído em uma etapa posterior.

A Figura 25 mostra a retopologia sendo feita, acima é possível ver a malha do escaneamento e da modelagem sobrepostas e abaixo é possível ver somente a malha da modelagem em processo de construção.

Figura 25 – Processo de retopologia da cabeça

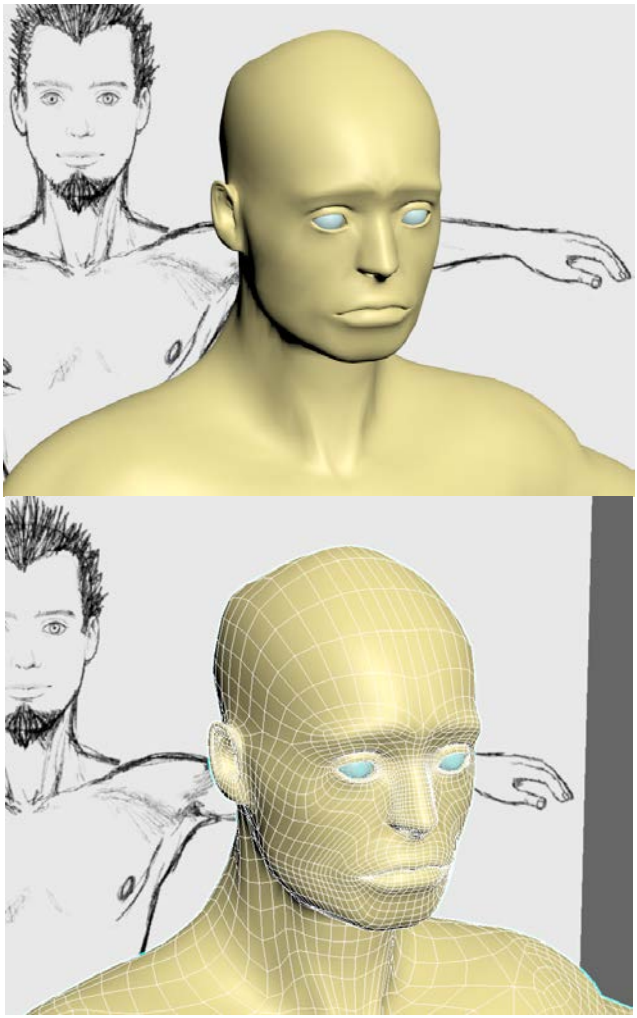


Fonte: Elaborada pelo autor

Usando este processo, a modelagem da cabeça foi feita com mais agilidade do que o corpo, o resto das características faciais foram construídas e uma esfera separada foi criada para fazer o globo ocular. É possível ver o resultado final na Figura 26. Abaixo, ainda na Figura 26,

mostram-se os polígonos que constituem a cabeça para que se possa comparar o resultado final com a malha proveniente do escaneamento (Figura 24). Na malha final há muito menos triângulos e imperfeições, além do fluxo dos polígonos respeitar a musculatura da face.

Figura 26 – Resultado final da modelagem da cabeça e sua estrutura de polígonos.

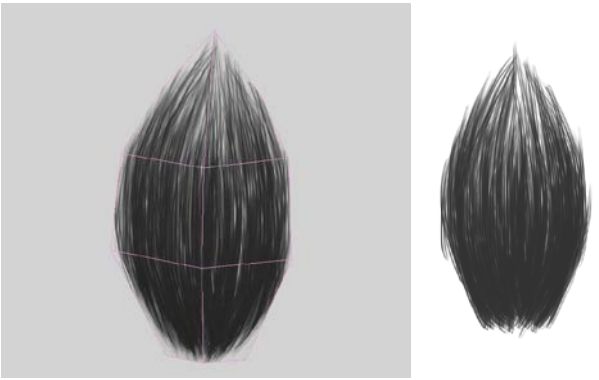


Fonte: Elaborada pelo autor

4.3.1.3 Cabelo

O cabelo foi modelado de maneira separada do corpo, ele é um plano torcido, feito de doze triângulos, onde uma textura parecida com fios de cabelo é aplicada. Na Figura 27 pode-se visualizar o processo de construção dele, onde à esquerda é possível ver o resultado final no programa 3DS Max e à direita a imagem utilizada como textura que foi construída no programa Adobe Photoshop e pintada manualmente. O fundo da imagem utilizada como textura é transparente para aumentar a fidelidade com relação aos fios de cabelo reais.

Figura 27 – Construção do cabelo utilizado no personagem.

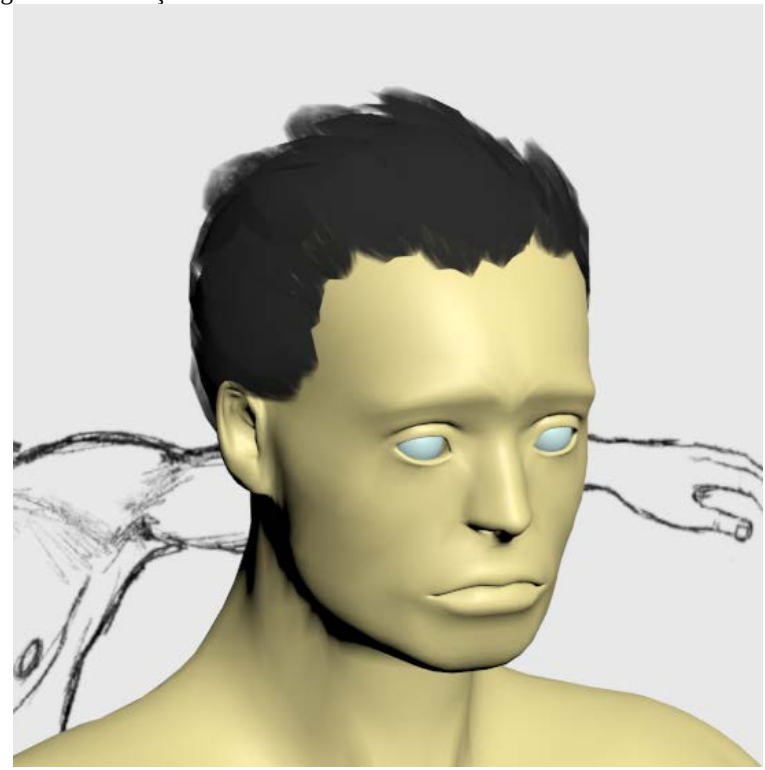


Fonte: Elaborada pelo autor

Para compor todo o cabelo, vários desses planos são colocados sobre a superfície da cabeça do personagem modelado, utilizando a função *Object Paint* presente no *software* 3DS Max, que pode ser usada para distribuir várias cópias de um objeto sobre uma malha.

Depois desse processo, alguns dos planos são rearranjados ou excluídos para melhorar a estética da construção de toda a malha do cabelo e o resultado final pode ser visto na Figura 28, com a configuração final da cabeça e cabelo.

Figura 28 – Cabeça com cabelo final



Fonte: Elaborada pelo autor

No total foram utilizados cento e quarenta e sete planos para a construção do cabelo, tomou-se o cuidado de excluir qualquer plano que não acrescentasse visualmente na estética final, de forma que não fossem utilizados triângulos desnecessários.

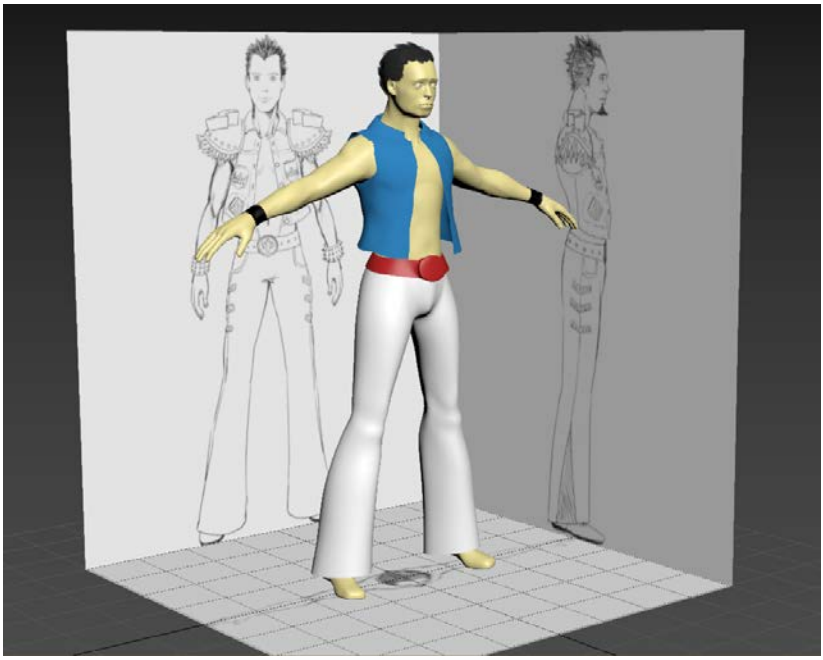
4.3.1.4 Roupas

As roupas foram os últimos itens a serem criados. No programa 3DS Max trocou-se o *model sheet* do personagem para as imagens dele com roupas e modelou-se primeiramente a jaqueta, calça, cinto e pulseiras.

Para modelar essas quatro peças, selecionou-se uma parte da malha do personagem correspondente a onde essas roupas iriam ser

colocadas e fez-se uma cópia dessa região em um objeto separado. A partir daí, ajustou-se a malha até que o formato final ficasse próximo das roupas descritas no *model sheet.* O resultado deste processo pode ser visto na Figura 29.

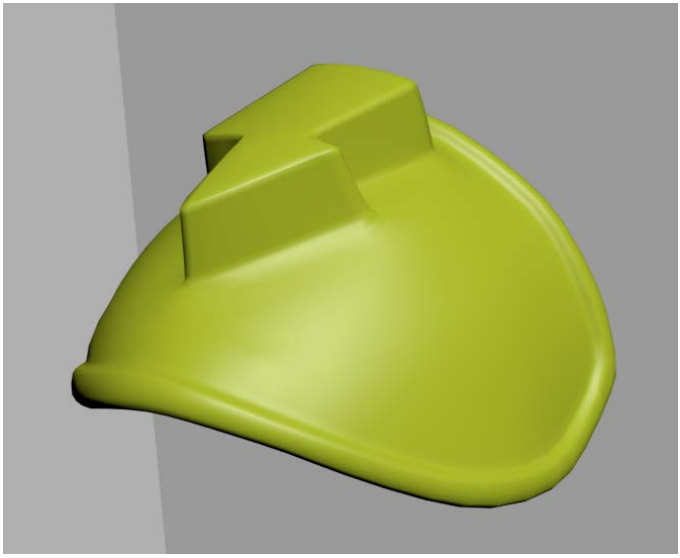
Figura 29 – Personagem com quatro peças da roupa modeladas.



Fonte: Elaborada pelo autor

Em seguida foi modelada a ombreira, que foi construída a partir do *model sheet* da mesma forma que o corpo do personagem. Criou-se um plano simples que foi sendo modelado até chegar-se na versão final da ombreira (Figura 30), lembrando que a função de simetria do software 3DS Max replica a mesma, não sendo necessário modelar a segunda.

Figura 30 – Versão final da modelagem da ombreira.



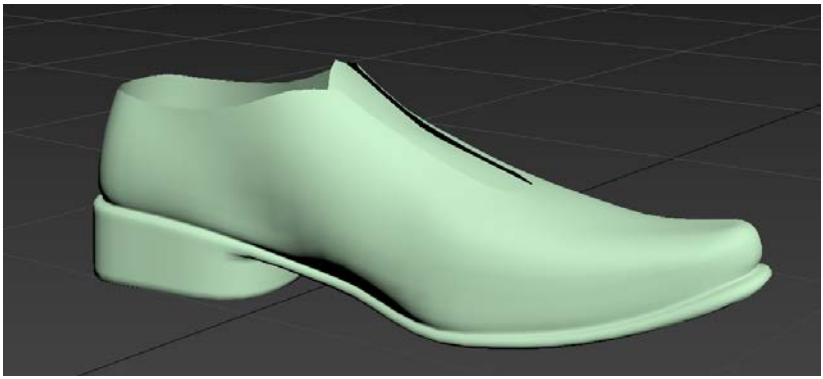
Fonte: Elaborada pelo autor

Por fim, foi criado o calçado. Foram utilizadas fotografias de um sapato em diversos ângulos encontradas em um tutorial de modelagem disponível na internet. Estas imagens serviram como base para a construção do modelo, da mesma forma que o *model sheet* foi a base para o personagem, e podem ser vistas na Figura 31. O sapato final modelado está na Figura 32.

Figura 31 – Fotos de diferentes ângulos de um sapato de couro.

Fonte: Kumar (2012)

Figura 32 – Modelagem final do sapato utilizado no personagem.

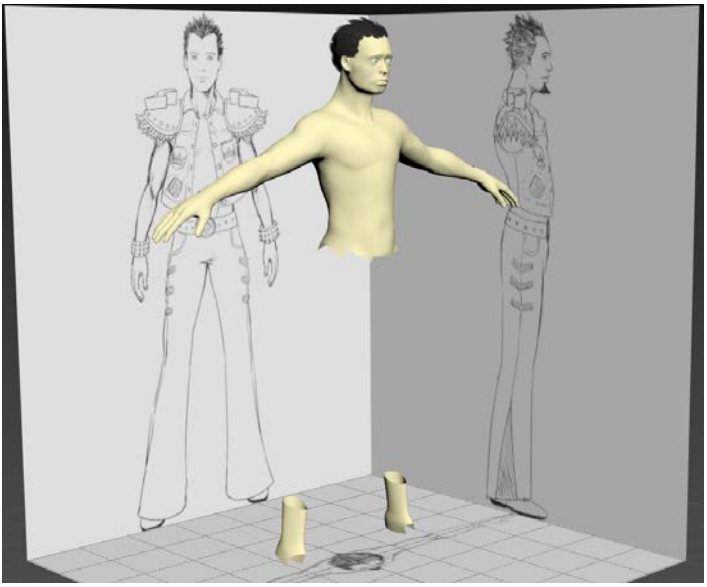


Fonte: Elaborada pelo autor

O último passo para a obtenção do modelo final foi excluir quaisquer polígonos que não fossem visíveis por estarem ocultos pela calça ou sapato, reduzindo a contagem geral de triângulos para

economizar desempenho. A Figura 33 mostra o personagem final sem as roupas e a Figura 34 exibe o modelo final do mesmo, mostrando também, abaixo, sua estrutura de polígonos.

Figura 33 – Modelo do personagem final sem roupas.



Fonte: Elaborada pelo autor

Figura 34 – Modelo do personagem final com roupas e sua estrutura de polígonos.

Fonte: Elaborada pelo autor

O personagem final, incluindo suas roupas, alcançou a contagem de cerca de quarenta e seis mil triângulos, ficando um pouco abaixo dos cinquenta mil esperados. Não foi modelada a penugem abaixo das ombreiras e nem detalhes do cinto, pulseiras e calça afim de evitar o aumento do número de polígonos. Parte destes detalhes foi construído na etapa de texturização e parte foi descartado devido ao tempo de execução previsto ultrapassar o limite disponível para este projeto.

4.3.2 Rigging

A etapa de *Rigging* tem o propósito de criar uma lógica de movimentação dos vértices do personagem para que a malha dele possa se manter coesa com sua estrutura quando ele for animado. O resultado final dessa fase é uma *Skinned mesh* descrita na Revisão Bibliográfica, no item 2.1.4 deste documento.

O primeiro passo é a construção de um esqueleto interno onde cada vértice do personagem se ligará a uma ou mais ossos (*bones*) deste esqueleto. Cada osso ligado a vértices, entretanto, aumenta o custo computacional ao animar o personagem.

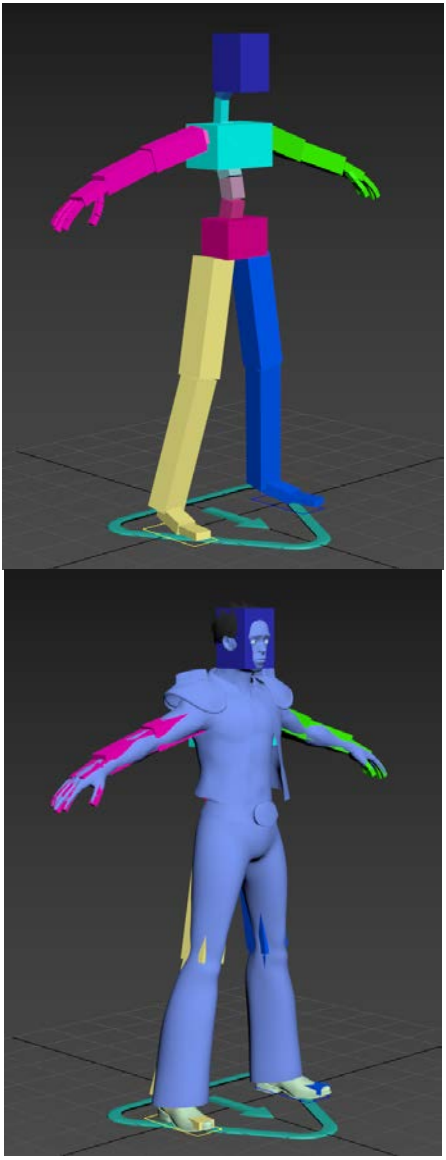
O item 2.1.4 deste trabalho também traz a informação que o manual da *engine* Unity recomenda entre quinze e sessenta ossos (*bones)* no esqueleto de um personagem, sendo recomendável um deles não influenciar mais do que quatro vértices.

Entretanto, como também descrito no capítulo de Revisão Bibliográfica, jogos desenvolvidos para as plataformas atuais, como *Final Fantasy XV* e *Ryse: Son of Rome*, ultrapassam esses números, com personagens com mais de seiscentos *bones*.

Este trabalho se focou em ficar dentro dos valores descritos pelo manual da *engine* Unity pois a criação de algo próximo de seiscentos *bones* é inviável dado o tempo e recursos disponíveis.

O esqueleto foi criado dentro do programa 3DS Max, que possui um sistema de *bones* chamado *Character Animation Toolkit* (CAT). Na Figura 35, acima, há a estrutura de ossos criada e, abaixo, como o personagem a envolve.

Figura 35 – Estrutura de bones feito para o *rig*



Fonte: Elaborada pelo autor

O cabelo não possui *bones* internos e, portanto, não terá este tipo de animação. Foram criados cinquenta e oito para o corpo. Nota-se que o local mais propício para economizá-los, caso seja necessário, são as mãos, que, sozinhas, constituem-se de trinta e dois *bones* devido às diversas juntas nos dedos.

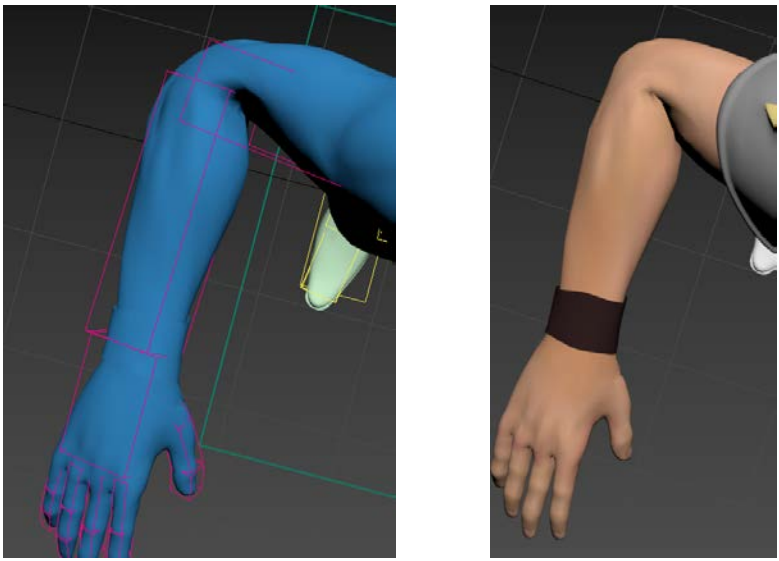
Após este processo conecta-se os vértices da malha do personagem aos *bones* com um modificador chamado *Skin* dentro do *software* 3DS Max. Dentro deste modificador foi escolhida a opção para cada vértice não ser influenciado por mais de quatro *bones*, de forma a ficar dentro das especificações do manual da *engine* Unity, além de facilitar o processo.

O *rigging* se inicia nesta etapa e consiste em informar manualmente ao programa o quanto de influência cada vértice recebe de cada *bone* ao movimentá-lo. Em geral são criadas animações curtas com movimentação de juntas e rotações do corpo humano, que consistem locais e movimentos onde os vértices se movem de maneira mais problemática, muitas vezes atravessando outra parte da malha ou fazendo a mesma perder volume.

A Figura 36 expõe, à esquerda, um exemplo da movimentação da junta do cotovelo do *rig* do personagem com perda de volume e, à direita, a movimentação com o *rig* corrigido, sem perda de volume e mais fiel a uma movimentação natural de um ser humano.

É interessante notar, também a partir da Figura 36, que ao corrigir-se o *rig*, a deformação da malha ao redor da junta é parecida com a deformação real dessa mesma junta em um corpo humano, uma vez que a modelagem foi feita seguindo a musculatura real.

Figura 36 – *Rig* com perda de volume à esquerda, *rig* corrigido à direita.

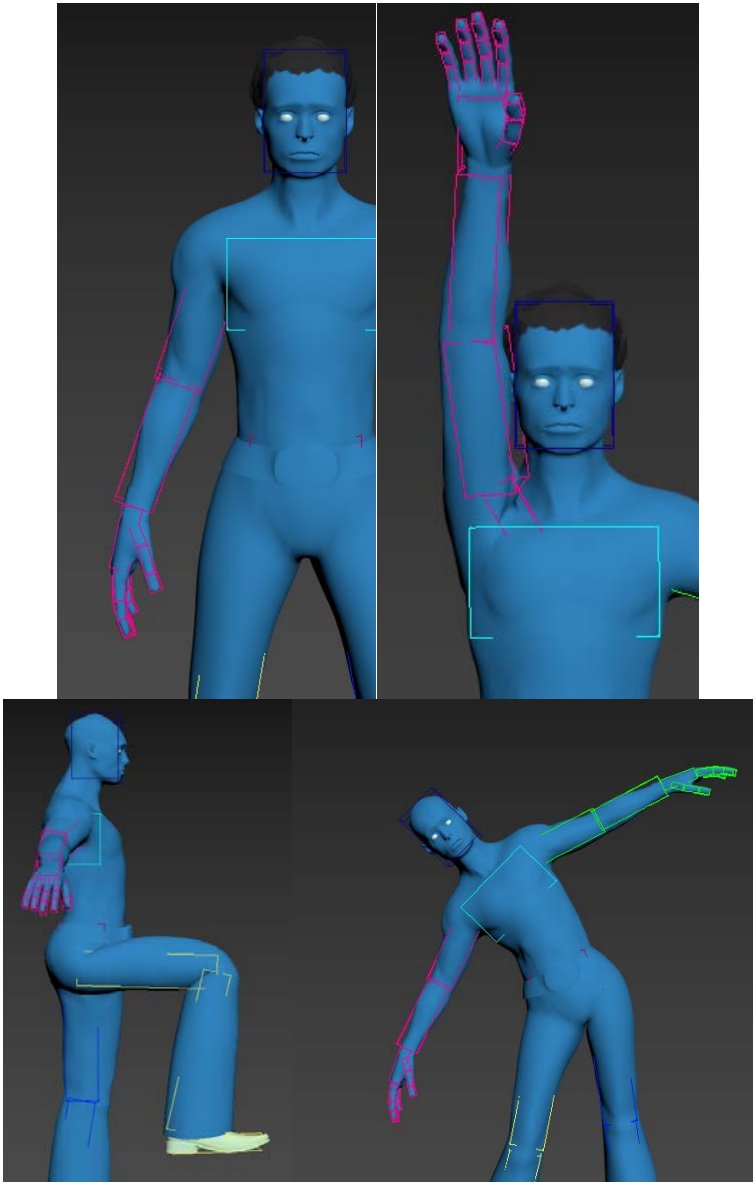


Fonte: Elaborada pelo autor

Com relação às roupas e ornamentos, a jaqueta foi conectada totalmente ao *bone* da caixa toráxica (maior *bone* em azul claro na Figura 35) e as ombreiras conectadas completamente ao *bones* de suas respectivas clavículas, fazendo com que essas partes sempre fossem rígidas em qualquer tipo de movimentação. A calça passou pelo processo de *rigging* normalmente, sendo tratada como as pernas do personagem, com trabalhos sendo feitos principalmente na dobra do joelho e no encaixe dos quadris com o torso.

A Figura 37 apresenta algumas poses que foram utilizadas para corrigir os problemas de *rig*. Na parte superior da figura é possível ver movimentações de ombro e na parte inferior movimentação de joelho e do torso. Frequentemente ocultou-se as roupas do torso para facilitar a visualização das deformações. As deformações nas figuras não necessariamente representam a versão final do *rig* de cada pose.

Figura 37 – Algumas poses utilizadas para o rig.



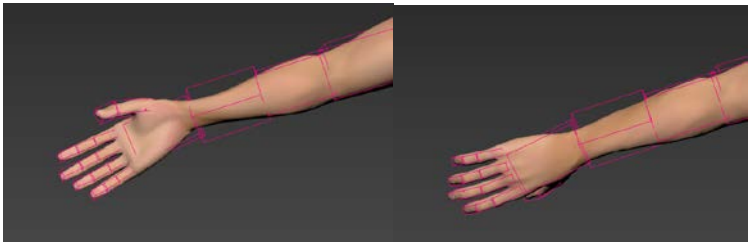
Fonte: Elaborada pelo autor

Reitera-se que foram utilizadas mais poses para fazer o *rig* completo, cada dedo da mão possui 3 juntas, por exemplo, que devem ser corrigidas uma por uma (em apenas um lado do modelo, já que é possível replicar os pesos dos vértices através da ferramenta de simetria). Demonstrar cada junta sendo resolvida não faz parte do escopo deste trabalho.

Durante o processo de *rig* o professor Flávio Andaló informou sobre a existência de um *website* chamado “Mixamo.com”, pertencente à empresa Adobe, que faz o *rig* de personagens humanos automaticamente. O personagem desenvolvido neste projeto foi testado no *site* e o *rig* automático gerou resultados satisfatórios. Entretanto, ele ainda precisava de ajustes e como o *rig* manual encontrava-se em etapa avançada, decidiu-se prosseguir com ele. O *site* Mixamo, entretanto, pode servir de base para *rigs* humanos futuros, agilizando o processo.

No final do processo, percebeu-se que a rotação da mão em conjunto com o braço estava distorcendo o cotovelo. Para resolver o problema decidiu-se criar um *bone* adicional em cada antebraço para que ele fosse responsável pela rotação sem interferir com o cotovelo. Ele pode ser visto na Figura 38, que mostra a rotação da mão tanto para a região anterior como posterior e o *bone* adicional começando no pulso e terminando no meio do antebraço. Com isso, o número de *bones* final subiu para sessenta, no valor superior limite do recomendado pelo manual da *engine* Unity.

Figura 38 – Rotação da mão com a criação de um *bone* adicional.



Fonte: Elaborada pelo autor

A etapa de *rigging* foi encerrada após a finalização da rotação do antebraço e o personagem estava pronto para ser animado.

4.3.3 Texturização

Após o *rigging* seguiu-se em paralelo a etapa de texturização e animação. Nessa seção será descrita como foram criadas e aplicadas as texturas que compõem o personagem. Esta etapa não é necessariamente posterior ao *rig* e pode ser feita a partir do momento em que a malha do personagem está concluída.

Para texturizar é necessário criar um mapa, planificando toda a malha tridimensional do personagem para uma imagem bidimensional, que pode ser pintada e manipulada digitalmente. O princípio é o mesmo da criação de um mapa mundi tradicional, onde o globo terrestre é planificado para uma imagem bidimensional. Nem todos os objetos 3D são planificáveis sem que ocorram distorções na malha, e no caso do personagem 3D isso deve ser esperado.

Primeiramente, utilizando-se da ferramenta UWV Map do *software* 3DS Max, cria-se divisões na malha do personagem, chamadas de *seams* (costuras, em inglês). Estas divisões servem para separar partes da malha para serem planificadas de maneira independente.

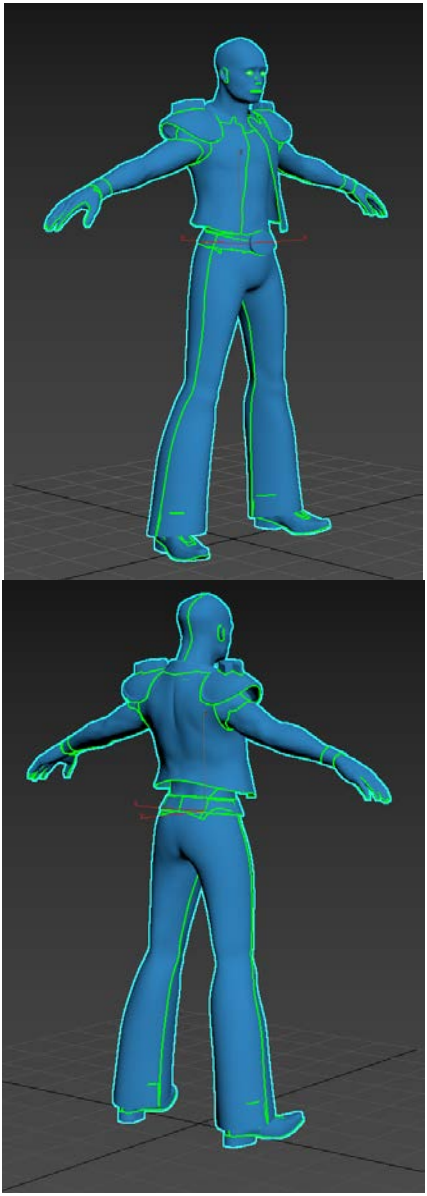
As linhas em verde vistas na Figura 39 representam as *seams* e no caso do corpo do personagem, dividiu-se a parte anterior e posterior tanto da calça como do torso do mesmo. Os braços foram separados do torso e das mãos além de uma única *seam* que se inicia na axila e vai até o pulso. As mãos foram divididas em parte superior e inferior e ambas as regiões são conectadas pela região lateral do dedo mínimo.

Na cabeça, as orelhas são separadas e o interior da boca, nariz e olhos também. Na cabeça ainda há uma *seam* que começa no topo e vai até o músculo trapézio no começo das costas, que pode ser visto na porção inferior da Figura 39.

Com relação às roupas, não há divisão interna na jaqueta. Nas ombreiras se separou o símbolo do raio do resto do objeto e o sapato é dividido entre sola e o resto do couro, com uma divisão na parte posterior, próximo ao calcanhar.

É conveniente evitar muitas *seams* uma vez que os vértices entre as divisões são processados duas vezes em uma *engine* de jogos, como descrito no item 2.1.2 deste documento. Por isso, evitou-se criar mais divisões nos braços, nas mãos ou na própria jaqueta. Algumas divisões, entretanto, são necessárias para ser possível planificar o objeto, como aquela encontrada no topo da cabeça até as costas.

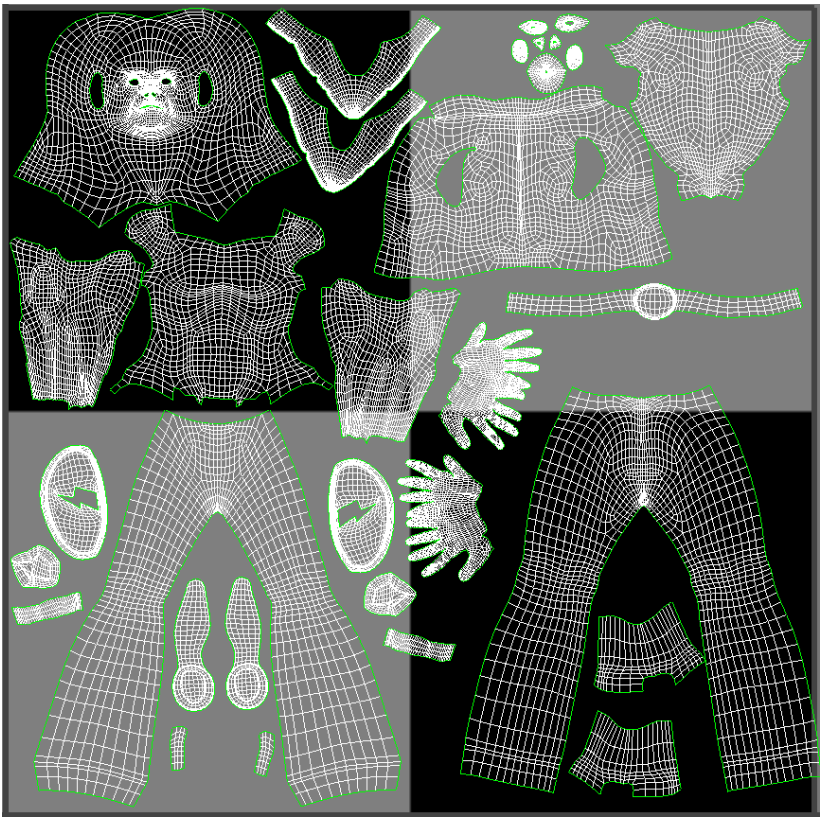
Figura 39 – *Seams* (em verde) do mapa de textura do personagem.



Fonte: Elaborada pelo autor

Após a criação das *seams*, é utilizada a ferramenta UVW Map do programa 3DS Max para “abrir” e esticar a malha em um mapa bidimensional contendo todas as partes divididas. A Figura 40 mostra o resultado final desse processamento, nota-se algumas partes reconhecíveis como a cabeça no canto superior esquerdo ou a calça na porção inferior. O globo ocular não está incluído neste mapa pois foi feito de maneira separada.

Figura 40 – Mapa de textura do personagem.



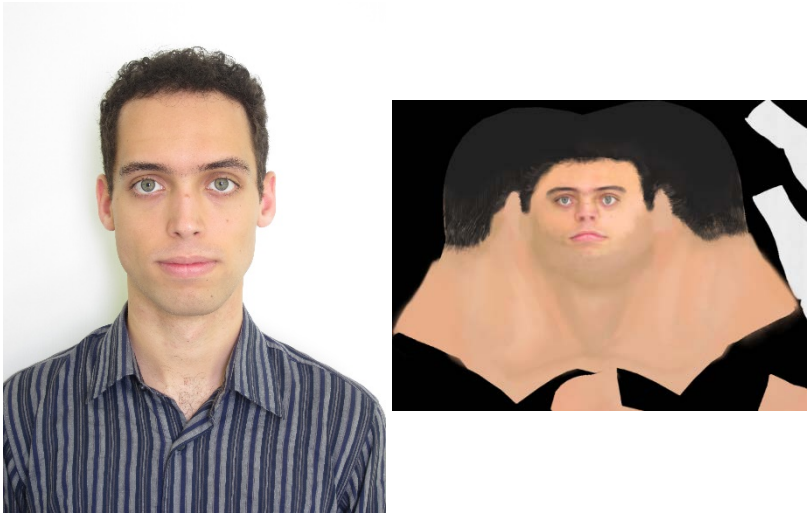
Fonte: Elaborada pelo autor

Esse mapa é associado ao personagem e é possível salvar essa imagem em um arquivo comum, com extensão png ou jpg e em seguida cria-se a textura em um programa de edição de imagens. No caso, foi

utilizado o *software* Photoshop da empresa Adobe. Este mapa foi exportado em uma resolução de 4096 por 4096 pixels. No item 2.1.3 deste documento explica-se que este tamanho não é recomendado para ser aplicado em mapas de textura de personagens de um jogo eletrônicos, mas imagens podem ser reduzidas sem nenhum custo adicional a qualquer momento após a criação do mapa e, portanto, não houve uma preocupação maior com o tamanho da mesma.

Como a cabeça foi escaneada a partir do autor deste documento, selecionou-se uma foto do mesmo (Figura 41 à esquerda) e ela foi colocada sobre a região do mapa correspondente a cabeça (Figura 41 à direita), já com as devidas distorções previstas. Utilizou-se das ferramentas de edição de imagem do programa Photoshop para pintar o resto da cabeça e aplicar textura na mesma.

Figura 41 – Foto usada como base para a textura da cabeça (à esquerda) e textura final (à direita).

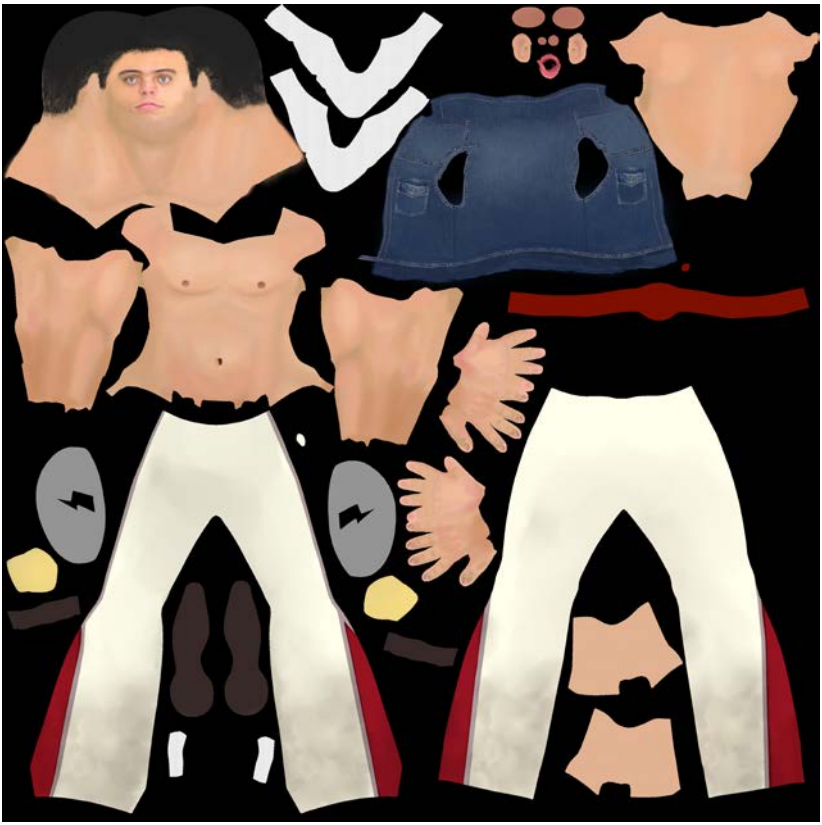


Fonte: Elaborada pelo autor

O resto das texturas foi pintada manualmente, por vezes utilizou-se de imagens auxiliares para a construção de fibras de texturas, como por exemplo para a criação da jaqueta jeans ou do couro do sapato, mas as cores foram feitas todas manualmente.

A Figura 42 mostra o mapa de textura final do personagem. Neste ponto, não se encontrou boas soluções para alguns detalhes como o cavanhaque do personagem ou para as cores do cinto e os detalhes da jaqueta e calça, como bolsos e ornamentos presentes no *model sheet*. Estes aspectos foram descartados devido à falta de tempo para a elaboração dos mesmos.

Figura 42 – Mapa de textura final do personagem.

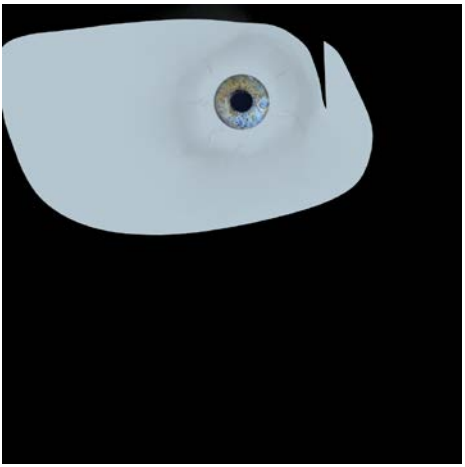


Fonte: Elaborada pelo autor

O mapa de textura dos olhos foi feito de forma separada e pode ser visto na Figura 43, utilizou-se uma imagem de tamanho 1024 por 1024 pixels, que também é grande para uma parte tão pequena do corpo, entretanto, como citado anteriormente, é possível reduzir a imagem a

qualquer momento caso ela cause algum impacto indesejado na performance do sistema.

Figura 43 – Mapa de textura do olho do personagem.



Fonte: Elaborada pelo autor

Com isso, chegou-se ao final da etapa de texturização do personagem. Ao passar o personagem para a *engine* ainda foram feitas algumas modificações que serão detalhadas posteriormente. O personagem com texturas sendo executado no programa 3DS Max pode ser visto na Figura 44.

Figura 44 – Personagem final texturizado.

Fonte: Elaborada pelo autor

4.3.4 Animação

A animação foi feita em paralelo com a etapa de texturas. Nesta fase são planejados e implementados movimentos do personagem que serão utilizados no jogo.

Para fazer o planejamento das animações, foi instalado em um computador o jogo Ultra Street Fighter 4, produzido pela empresa Capcom. Foi feita uma análise das animações do personagem Ryu deste jogo. Uma tela do jogo pode ser vista na Figura 45.

Verificou-se que há três botões que ativam movimentos de soco e três botões que ativam movimentos de chute diferentes, cada soco ou chute pode ser executado no chão, no ar ou agachado, criando de imediato dezoito combinações diferentes de animação. O personagem ainda pode usar 3 tipos de ataques especiais, alguns combinados com o botão de soco e alguns combinados com o botão de chute, alguns podem ser executados no ar outros não.

Figura 45 – Tela do jogo Ultra Street Fighter 4

Fonte: Captura de tela do jogo executando no Windows 10

Além disso o personagem tem diversas outras animações como a pose *idle* (personagem quando não está executando nenhuma ação), andar para frente, andar para trás, defender, pular em três diferentes direções, levar dano, entre outras, como provocações e ataques ultra especiais.

Notou-se que as animações são preparadas de forma a fecharem ciclos de movimento. A pose *Idle* por si só é uma animação que se repete constantemente, além disso, vários movimentos começam e terminam nessa pose, sendo muitas animações preparadas para fecharem ciclos nesta posição.

Contou-se cento e oito movimentos apenas para esse personagem. Este número foi obtido a partir de uma contagem por observação e não deve ser considerado como um valor correto do número de animações, sendo apenas uma estimativa.

Com o tempo disponível para o projeto, não era possível criar um personagem completo, com mais de cem animações e, por isso, focou-se na criação de dez animações.

Foi decidido que as animações seriam feitas laboratório Tecmídia da Universidade Federal de Santa Catarina (UFSC), através da técnica de captura de movimentos, ajudando a criar movimentos de luta mais naturais, além de agilizar o processo de criação das animações.

As dez animações foram pensadas de forma que o personagem pudesse interagir com ele mesmo no jogo de luta. Criando, por exemplo, um soco e o personagem defendendo. Com isso em mente, decidiu-se desenvolver as seguintes animações:

1. *Idle* (animação do personagem quando não está executando nenhuma ação);
2. Caminhada para frente;
3. Caminhada para trás;
4. Soco fraco;
5. Soco forte;
6. Chute fraco;
7. Chute forte;
8. Defesa;
9. Recebendo dano leve;
10. Recebendo dano pesado;

4.3.4.1 Storyboard

Para cada uma dessas animações foi gerado um *storyboard* para não só melhor planejar a movimentação do personagem, como também para facilitar a explicação para o ator que fará a captura de movimentos.

Os *storyboards* podem ser vistos nas Figuras 46, 47, 48, 49, 50, 51, 52, 53 e 54. A posição de *Idle* (Figura 46) não possui mais desenhos uma vez que, da forma como foi imaginada, tem movimentação sutil. Os *storyboards* de caminhada para frente e para trás são os mesmos, mas com movimentação invertida (Figura 47). Tentou-se criar um ciclo diferente para a caminhada para trás, mas a melhor solução ainda foi a inversão da caminhada frontal.

Figura 46 – Posição *Idle*

Fonte: Elaborada pelo autor

Figura 47 – Caminhada para frente/trás.

Fonte: Elaborada pelo autor

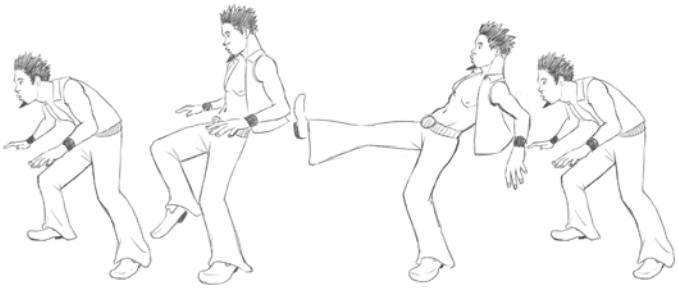
Figura 48 – Soco Fraco

Fonte: Elaborada pelo autor

Figura 49 – Soco forte.

Fonte: Elaborada pelo autor

Figura 50 – Chute fraco.



Fonte: Elaborada pelo autor

Figura 51 – Chute forte.

Fonte: Elaborada pelo autor

Figura 52 – Defesa.

Fonte: Elaborada pelo autor

Figura 53 – Recebendo dano leve.

Fonte: Elaborada pelo autor

Figura 54 – Recebendo dano pesado.

Fonte: Elaborada pelo autor

4.3.4.2 Captura de Movimentos

O passo seguinte é a gravação das animações no estúdio de captura de movimentos. Para isso é necessário um ator para interpretar o personagem enquanto uma pessoa dirige suas ações. A Figura 55 mostra o ator, o aluno Rhaniel Daux, com a roupa preta, enquanto o autor deste documento faz a direção ao seu lado. Um computador é posicionado próximo para que se possa recorrer visualmente aos *storyboards.*

Figura 55 – Fotos do processo de captura de movimento.

Fonte: Fotos da aluna Nayara de Souza Braga

A gravação é feita usando um arquivo separado para cada movimento, são feitas diversas tentativas de execução do mesmo em cada arquivo, dessa forma, pode-se selecionar a melhor versão na etapa posterior.

Recomenda-se a gravação da pose *Idle* primeiramente, uma vez que todos os outros movimentos começam a partir dela. Orientou-se ao ator para ficar nessa pose antes de iniciar qualquer um dos outros movimentos. Este processo facilita também criar a transição entre uma pose e outra na etapa de preparação e refinamento das animações.

Não houve foco no tempo de execução dos movimentos uma vez que este tipo de tarefa é mais importante na área de *gameplay* que não está no escopo deste trabalho.

4.3.4.3 Preparação e Refinamento das Animações

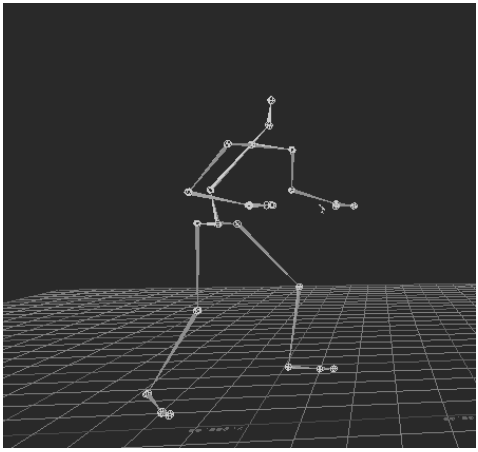
Com os arquivos da captura em mãos, faz-se necessário preparar os movimentos para o jogo. Como visto anteriormente, no jogo Street Fighter, as animações são criadas de forma a fecharem ciclos. A animação *Idle*, por exemplo, é um ciclo fechado que o personagem executa

enquanto não interage com nada e não recebe nenhum comando do jogador. A maior parte das outras animações do jogo *Ultra Street Fighter IV* se iniciam dessa posição e terminam também nela e os *storyboards* feitos para este projeto também seguem este mesmo padrão.

Para preparar as animações utilizou-se o *software* Motion Builder, da empresa Autodesk. Importam-se os dados da captura de movimento para ele, e em seguida importa-se o personagem com o *rig* pronto. Em um processo chamado *Retargeting* os movimentos capturados são colocados no personagem final para em seguida serem refinados.

Na Figura 56 é possível visualizar o esqueleto obtido através da captura de movimentos executando a pose *Idle* no programa Motion builder.

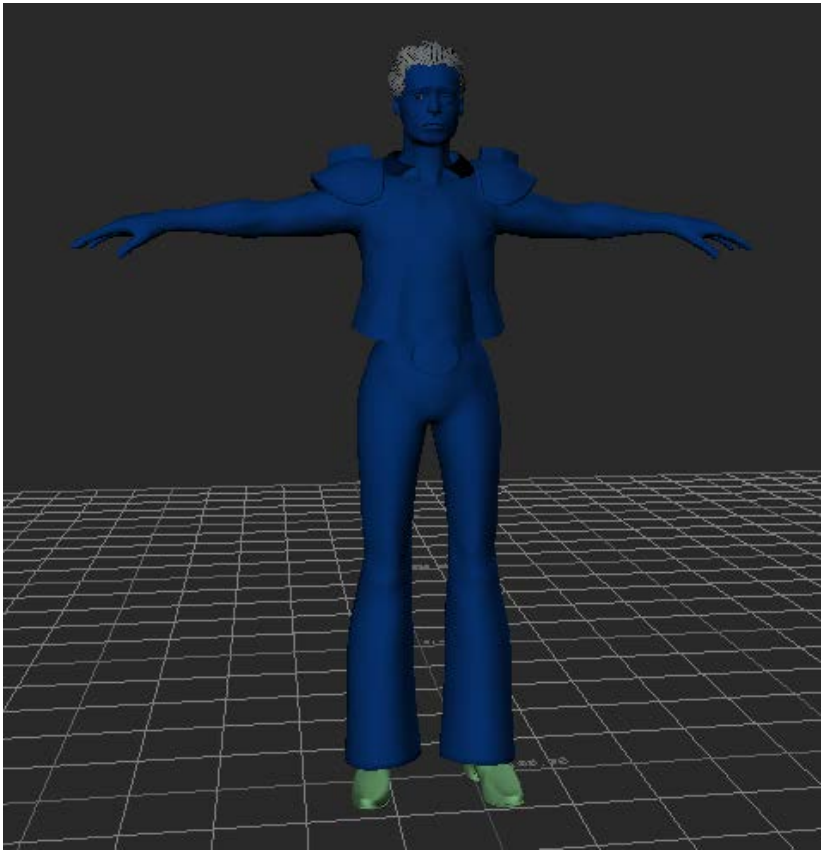
Figura 56 – Pose *Idle* proveniente da captura de movimentos.



Fonte: Elaborada pelo autor.

Para importar o personagem e fazer o processo de *retargeting* corretamente o programa Motion Builder pede que ele esteja na chamada *T-pose* com os braços paralelos ao plano do chão. O personagem não foi modelado dessa forma e por isso nesta etapa foi necessário corrigir a orientação dos braços ainda no programa 3DS Max, a pose final do personagem pode ser vista na Figura 57 já no importado para o *software* Motion Builder.

Figura 57 – Personagem na *T-pose* no *software* Motion Builder.



Fonte: Elaborada pelo autor

Neste programa, faz-se um processo chamado de caracterização, onde o esqueleto desenvolvido para o personagem é associado a um modelo de esqueleto existente dentro do programa Motion Builder. Depois deste processo é possível associar as animações provenientes da captura com o personagem modelado e iniciar a preparação das mesmas para o jogo.

A criação das animações iniciou-se com o personagem ainda sem texturas e com um problema no *rig* no cotovelo, que foi corrigido posteriormente. Entretanto, para a melhor visualização, serão mostradas

imagens do personagem com texturas para ilustrar o processo de preparação das animações.

Nas Figura 58, 59, 60, 61 e 62 há quadros com o personagem se mexendo de acordo com as animações capturadas, ainda sem tratamento.

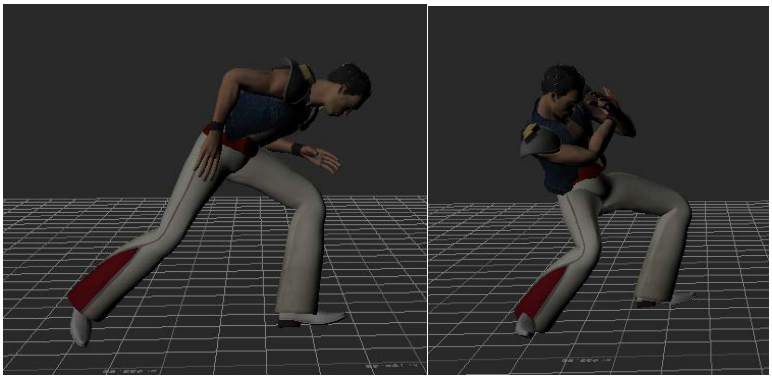
Figura 58 – Captura de movimento da pose *Idle* (à esquerda) e da caminhada para frente (à direita).

Fonte: Elaborada pelo autor

Figura 59 – Captura de movimento da caminhada para trás (à esquerda) e do soco fraco (à direita).

Fonte: Elaborada pelo autor

Figura 60 – Captura de movimento do soco forte (à esquerda) e da defesa (à direita).



Fonte: Elaborada pelo autor

Figura 61 – Captura de movimento do chute forte (à esquerda) e do chute fraco (à direita).

Fonte: Elaborada pelo autor

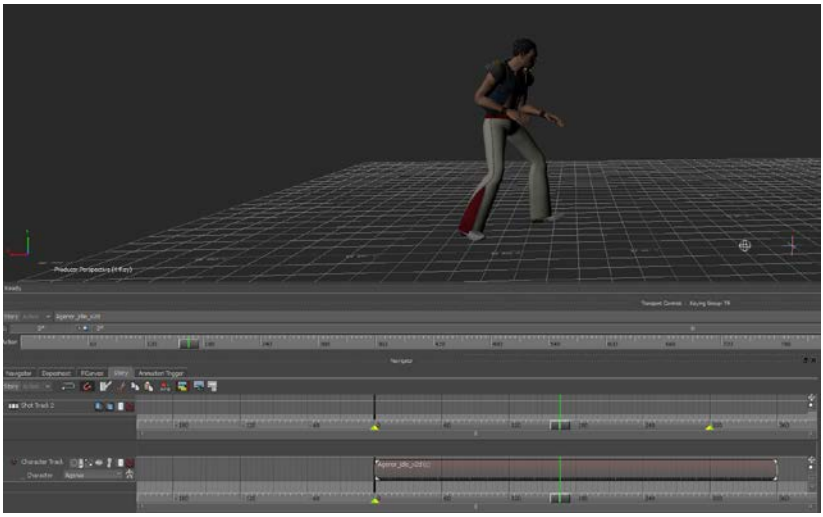
Figura 62 – Captura de movimento de recebendo dano leve (à esquerda) e de recebendo dano pesado (à direita).

Fonte: Elaborada pelo autor

Neste ponto, verificou-se que não seria possível preparar todos os movimentos em tempo de conclusão deste projeto. Devido a este fator, escolheu-se 5 movimentos para serem finalizados: *Idle*, caminhada para frente, caminhada para trás, soco fraco e recebendo dano leve.

O primeiro movimento finalizado foi o *Idle* uma vez que todos os outros dependem dele. O processo funciona da seguinte forma: dentro do programa Motion Builder, seleciona-se uma parte da gravação é considerada esteticamente mais interessante para a construção do movimento. Em seguida, o exemplo da construção do movimento *Idle* pode ser visto na Figura 63. A faixa avermelhada na parte inferior é a região com os quadros (*frames*) da captura de movimento selecionada. Corta-se essa faixa ao meio (linha verde) e inverte-se o posicionamento das duas faixas, dessa forma, o primeiro e o último frame da animação serão exatamente o mesmo, garantindo que um ciclo seja fechado. Na região central, utiliza-se a ferramenta de mesclagem do programa Motion Builder para misturar as duas animações.

Figura 63 – Processo de geração de um ciclo no programa Motion Builder.



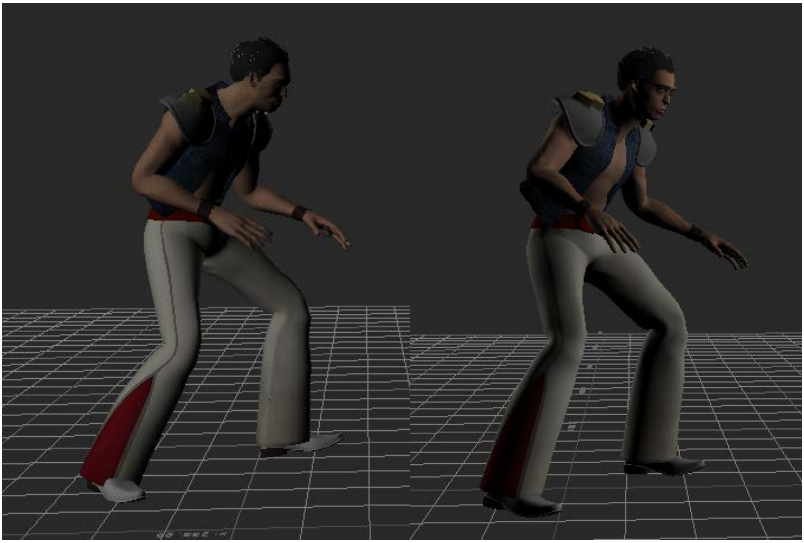
Fonte: Elaborada pelo autor

Depois deste passo, algumas inconsistências aparecerão na região onde a animação foi mesclada. As mais comuns são: pés

deslizando e/ou mudando de altura, reposicionamento repentino de braços e/ou joelhos e mudanças na inclinação da coluna. Todos estes problemas devem ser corrigidos na animação, até deixá-la limpa e com o ciclo fechado.

Além disso, problemas na pose em si do personagem podem ser corrigidas nesta etapa, mudando a inclinação da coluna e/ou cabeça e alterando posições de braços e pernas de forma a melhorar a silhueta e informação transmitida pelo personagem, um exemplo disto pode ser visto na Figura 64, mostrando as diferenças entre a pose capturada (à esquerda) e a pose limpa no final (à direita), com o personagem com o torso e a cabeça mais virados para a câmera e os braços mais afastados um do outro.

Figura 64 – Comparação entre pose da captura de movimento (à esquerda) e da animação final (à direita) da animação *Idle.*

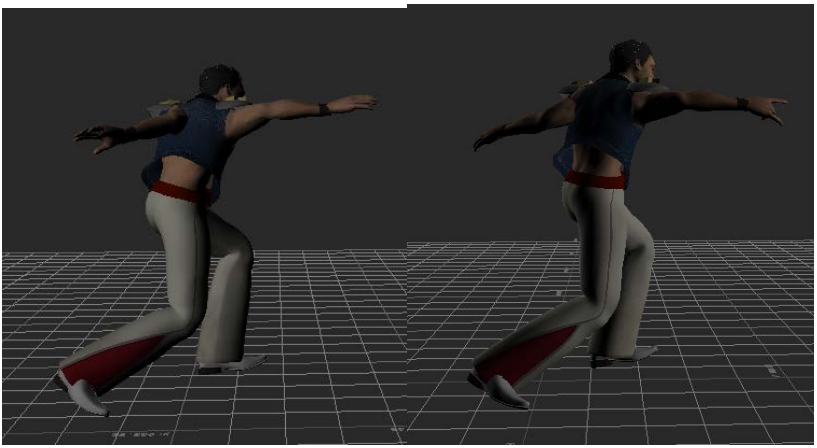


Fonte: Elaborada pelo autor

Depois de pronta a pose *Idle*, as outras são feitas com este mesmo processo. A única diferença que pode ser apontada é que a versão final da *Idle* é mesclada no começo e no final de cada das outras animações, para que a transição entre elas seja mais natural ao serem colocadas em uma *engine* de jogos. Na Figura 65 pode-se ver o exemplo da criação do

movimento de soco fraco, com a captura à esquerda e a animação final à direita. Verifica-se que ambos os braços mudaram de posição, colocou-se animação nos dedos e alterou-se o posicionamento da cabeça e tronco.

Figura 65 – Comparação entre pose da captura de movimento (à esquerda) e da animação final (à direita) da animação do soco fraco.



No final, com todas as animações prontas, elas foram salvas e exportadas em arquivos separados para serem colocadas dentro da *engine* de jogos Unreal Engine 4.

4.3.5 Desempenho do personagem dentro da Unreal Engine 4

Antes de utilizar o personagem na Unreal Engine 4, decidiu-se separar as texturas dos objetos do personagem: as ombreiras, a jaqueta, o cinto, a calça e o sapato receberam uma identificação nova dentro do programa 3DS Max. Dessa forma, é possível separar a qualidade de cada material dentro da Unreal Engine. Isto, entretanto, aumenta o número de *Draw Calls* (ver item 2.1.1), reduzindo o desempenho do sistema. Como o personagem foi feito de maneira conservadora, julgou-se que a queda de desempenho não seria relevante e o personagem seria percebido como esteticamente melhor.

Manipulou-se alguns parâmetros como a rugosidade e o reflexo especular das texturas do cinto e ombreiras de forma que ficassem com uma aparência metálica, aumentou-se os reflexos nos olhos e fez-se com

que a pele ficasse mais avermelhada ao receber iluminação. O resultado final do personagem dentro da *engine* pode ser visto na Figura 66.

Figura 66 – Personagem dentro da Unreal Engine 4.

Fonte: Elaborada pelo autor

Todas as cinco animações foram importadas para o programa. Elas podem ser vistas dentro do programa nas Figuras 67, 68 e 69. Colocou-se o personagem em uma cena pré-existente dentro da *engine* para testar seu desempenho, ao lado de um outro modelo, a Figura 70 exibe uma imagem desta configuração.

Figura 67 – Animação Idle (à esquerda) e Caminhada para frente (à direita).

Fonte: Elaborada pelo autor

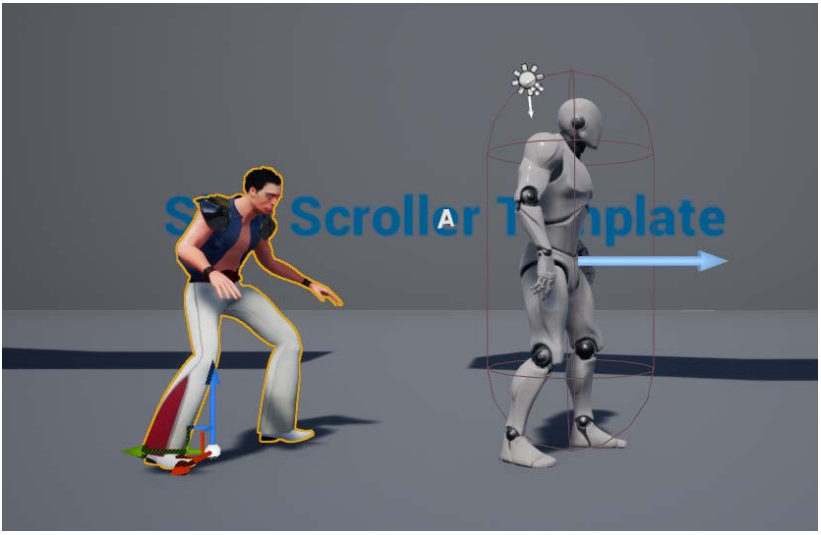
Figura 68 – Animação soco fraco (à esquerda) e caminhada para trás (à direita).

Fonte: Elaborada pelo autor

Figura 69 – Animação recebendo dano fraco.

Fonte: Elaborada pelo autor

Figura 70 – Personagem na pose *Idle* dentro de cena da Unreal Engine.



Fonte: Elaborada pelo autor

Dentro desta cena, o jogo executou sempre a 60 frames por segundo. Isto não deve ser usado como parâmetro para um jogo final uma vez que os modelos do ambiente e as iluminações não são iguais aos de um produto completo.

Por último, verificou-se, de maneira aproximada, o número de *Draw Calls* que o personagem executa. Utilizou-se uma ferramenta da própria Unreal Engine, que mostra os números referentes ao desempenho do jogo.

Com o personagem visível no jogo, todos os elementos da tela somam sessenta *Draw Calls* como pode ser visto na Figura 71 com o nome de *Mesh Draw Calls*. Sem o personagem na cena este número cai para trinta e cinco. Portanto, de maneira aproximada, o personagem aumenta o número de *Draw Calls* em vinte e cinco, este número irá variar dependendo da iluminação, sombras, posição de câmera, entre outros aspectos.

Figura 71 – *Draw Calls* com o personagem em cena (acima) e fora de cena (abaixo).

Fonte: Elaborada pelo autor

Como pode ser visto na tabela da figura 3, no item 2.1.2 deste documento, que mostra um típico caso de número de *Draw Calls* de cada elemento em um jogo, um personagem realizar quarenta *Draw Calls* não é incomum, e, portanto, o personagem desenvolvido neste projeto encontra-se dentro das especificações de jogos desenvolvidos atualmente.

O trabalho foi encerrado nesta etapa, mas ainda há bastante espaço para a otimização do personagem tanto nas texturas, quanto na redução das *Draw Call*s. Contudo, o processo de ajuste fino não faz parte dos objetivos desse projeto pois requer mais tempo de produção do que o disponibilizado.

O resultado final é um personagem ainda não totalmente refinado dentro da etapa de Produção, para ser considerado completo deve ser totalmente animado, polido e implementado em um jogo.

5. CONCLUSÃO

Este trabalho foi bem-sucedido em apontar diferentes limitações que artistas e *Designers* devem considerar ao desenvolver um personagem para um jogo eletrônico 3D. Mostrou-se como diversas variáveis, utilizadas constantemente por esses profissionais, podem ser controladas na modelarem de objetos 3D como, por exemplo, quantidade de polígonos e vértices, resolução e quantidade de mapas de textura e número de *bones* para um esqueleto de animação.

Além disso, ficou demonstrado que um artista deve também se preocupar com a influência da manipulação destas variáveis nos fatores mais técnicos, como o número de *Draw Calls*, que estão conectadas, em sua essência, ao modo como um computador faz a comunicação entre seus diferentes componentes.

Também houve sucesso em encontrar um planejamento para a produção deste personagem, mostrando e empregando parte de uma metodologia que é aplicada em grandes empresas do ramo de jogos eletrônicos.

A construção do personagem e sua implementação em uma *engine* de jogos, mesmo sem concluir a produção, foi bem-sucedida, mostrando a viabilidade da criação deste tipo de conteúdo com os recursos disponíveis.

Foi possível verificar que a produção de jogos 3D não é uma tarefa trivial, evidenciado primeiramente pela variada gama de programas utilizados; foram seis programas diferentes para a produção apenas de um personagem: Adobe Photoshop, Autodesk 3DS Max, Autodesk ReCap, Autodesk Motion Builder, Autodesk Mudbox e Unreal Engine. Cada programa executa diferentes funções e requer conhecimentos específicos para ser utilizado.

As habilidades e conhecimentos necessários para avançar e refinar cada uma das etapas da criação também mostram a complexidade deste projeto. Para produzi-lo foram necessárias habilidades nas áreas de criação de personagens, modelagem 3D, pintura digital, desenho e animação. Um jogo completo requer ainda mais conhecimentos como *Level Design*, programação, som, entre outros.

Além disso, tecnologias de ponta como o escaneamento 3D e a captura de movimento foram utilizadas, requerendo conhecimentos específicos também para manipulá-las. Portanto, com todos estes fatores apresentados, equipes de trabalho se mostram necessárias para otimizar a criação de um personagem ou um jogo completo.

Em termos acadêmicos, este projeto contribuiu para a formação do aluno, não apenas com o desenvolvimento das habilidades descritas, mas também com o conhecimento do fluxo de trabalho em cada um dos programas utilizados, além de mostrar as áreas que o aluno ainda pode progredir. A prática também explicitou quais etapas podem incluir grupos de trabalho para otimizar a produção e mostrou problemas que não seriam conhecidos sem a vivência do processo.

Para trabalhos futuros na área, sugere-se explorar com mais detalhes cada uma das fases do processo, documentando, por exemplo, diferentes soluções para a modelagem 3D, otimização do número de polígonos na construção de personagens, técnicas para a construção de mapas de textura para jogos, organização de texturas com o objetivo de redução do número de *Draw Calls*, otimização do número de *bones* em uma malha, e comparação entre diferentes técnicas de animação 3D para jogos.

Além disso, é possível desenvolver projetos em áreas que fugiram do escopo deste trabalho, como a implementação de uma máquina de estados completa de um personagem, papel da iluminação no desempenho de jogos, *Design* de ambientes, *level Design*, entre outros.

Verifica-se que a área de jogos para consoles é extensa e pode ser ainda bastante explorada. Este trabalho cumpriu seu papel de desenvolver as habilidades aprendidas durante o curso de *Design* e traz conhecimento acerca da produção de jogos que poderá ser utilizada pela Universidade, abrindo portas para outros trabalhos do gênero.

REFERÊNCIAS

BEETS, K. **GPU compute – what is it and why do we care?** 2013. Disponível em: <https://www.imgtec.com/blog/gpu-compute-what-is-it-and-why-do-we-care/> Acesso em: 01 novembro 2017.

CASWELL, T. **5 tips for playing Final Fantasy XV.** 2016. Disponível em: <http://www.gamezone.com/originals/5-tips-for-playing-final-fantasy-xv-k2x4> Acesso em: 20 maio 2017.

CHENG, L. **About Level of Detail (LOD) in Game.** 2015. Disponível em: <http://blog.gameartworkbook.com/game-art-theory/about-level-of-detail-lod-in-game/> Acesso em: 20 maio 2017.

CHIKHANI, R. **The History of Gaming: Na Evolving Community**. 2015. Disponível em: <https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/> Acesso em 14 de maio 2017

CIFALDI, F. **We asked three PS4 developers how expensive their games are.** 2013. Disponível em: <http://www.gamasutra.com/view/news/187037/We_asked_three_PS4_developers_how_expensive_their_games_are.php> Acesso em: 20 maio 2017.

COPELAND, W. **Understanding The Importance of Frame Rate.** 2014. Disponível em: <http://www.ign.com/articles/2014/11/05/understanding-frame-rate-and-its-importance> Acesso em: 20 maio 2017.

CRYENGINE. **Cryengine Manual**. 2016 <http://docs.cryengine.com/display/SDKDOC2/Home> Acesso em: 20 maio 2017.

EVANS, C. **Ryse: Son of Rome.** 2013. Disponível em: <http://www.crytek.com/download/Ryse_ChrisEvans_Sigg.pdf > Acesso em: 20 maio 2017.

FREDERIKSEN, E. **Horizon: Zero Dawn eyes-on preview – post-post apocalyptic.** 2015. Disponível em: <https://www.technobuffalo.com/2015/06/18/horizon-zero-dawn-eyes-on-preview-post-post-apocalyptic/> Acesso em: 20 maio 2017.

FULLERTON, T.; SWAIN, C.; HOFFMAN, S. **Game Design Workshop**: A Playcentric Approach to Creating Innovating Games. 2.ed. Estados Unidos: Elvesier, 2008. 470p.

GESOTA, R. **How To Make Your Games Run Superfast By Using Draw Call Reduction.** 2016. Disponível em: <http://www.theappguruz.com/blog/learn-draw-call-reduction-and-make-your-games-run-superfast> Acesso em: 20 maio 2017.

GHAZI, K. **The Gamer's Graphics & Display Settings Guide.** 201.2 Disponível em: <http://www.tweakguides.com/Graphics_1.html> Acesso em: 01 novembro 2017.

GREENWALD, W. **HDTV Refresh Rates Explained: 60 Hz, 120Hz, and Beyond.** 2014. Disponível em: <http://www.pcmag.com/article2/0,2817,2379206,00.asp> Acesso em: 20 maio 2017.

HOGAN, M. **Unity Optimizing For Mobile Using SubTile Meshes.** 2014. Disponível em: <http://www.gamasutra.com/blogs/MarkHogan/20140721/221458/Unity_Optimizing_For_Mobile_Using_SubTile_Meshes.php> Acesso em: 20 maio 2017.

HOOD, L. **Street Fighter V to Be Televised Live on ESPN2 from Evo 2016**. 2016. Disponível em: <http://shoryuken.com/2016/07/01/street-fighter-v-to-be-televised-live-on-espn2-from-evo-2016/> Acesso em: 14 maio 2017

IVANOV, I. **Practical Texture Atlases.** 2006. Disponível em: <https://gamedevelopment.tutsplus.com/articles/using-texture-atlas-in-order-to-optimize-your-game--cms-26783> Acesso em: 20 maio 2017.

JAMES, B. **UV Mapping Explained.** 2012. Disponível em: <https://worldorigin.wordpress.com/2012/05/30/uv-mapping-explained-36/> Acesso em: 20 maio 2017.

JIE, J.; YANG, K.; HAIHUI, S. Research on the 3D Game Scene Optimization of Mobile Phone Based on the Unity 3D Engine**.**

Internacional Conference on Computational and Information Sciences. p.875-877, 2011.

KERR, B. **Get Rich or Die Streaming: Making Money on Twitch.tv.** 2015. Disponível em: <https://thehustle.co/get-rich-or-die-streaming-making-money-on-twitch-tv> Acesso em 14 de maio 2017

KOTAKU. **How Much Does It Cost TO Make A Big Video Game?** 2014. Disponível em: <http://kotaku.com/how-much-does-it-cost-to-make-a-big-video-game-1501413649> Acesso em: 20 maio 2017.

KUMAR, C. **Model, UVMap, Texture, Light and Render Leather Shoes in Mayara – Part 1.** 2012. Disponível em: <https://cgi.tutsplus.com/tutorials/model-uvmap-texture-light-and-render-leather-shoes-in-maya-part-1--cg-17642> Acesso em: 06 de setembro de 2017.

LITHVALL, J. **Horizon Zero Dawn Hair.** 2017. Disponível em: <https://www.artstation.com/artwork/EDbk4> Acesso em: 20 maio 2017.

MARCHAND, A.; HENNIG-TRURAU, T. Value Creation in the Video Game Industry: Industry Economics, Consumer Benefits, and Research Opportunities**. Journal of Interactive Marketing,** v. 27, n. 3, p. 141-157, jul. 2013.

MATSUDAIRA, H. **The Making of 2B from NieR: Automata.** 2017. Disponível em: <http://na.square-enix.com/us/blog/making-2b-nierautomata> Acesso em: 20 maio 2017.

MCDONALD, E. **The global games Market will reach $108.9 billion in 2017 with mobile taking 42%.** 2017. Disponível em: <https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/> Acesso em: 14 maio 2017.

MICHAUD, S. **What Exactly is a Draw Call (and What Can It Do)?.** 2015. Disponível em: <https://www.pcper.com/reviews/Editorial/What-Exactly-Draw-Call-and-What-Can-It-Do> Acesso em: 20 maio 2017.

MPAA. **Theatrical Market Statistics.** 2016. Disponível em: <https://www.mpaa.org/wp-content/uploads/2017/03/MPAA-Theatrical-Market-Statistics-2016_Final.pdf> Acesso em: 01 novembro 2017.

MUELLER, S. **Report: League of Legends made $1.6 billion in revenue last year.** 2016. Disponível em: <https://dotesports.com/league-of-legends/league-of-legends-2015-revenue-2839> Acesso em: 14 maio 2017

MUTTAQIEN, W. **Tutorial: Modeling, UV Unwrapping and Texturing a Mushroom.** 2017. Disponível em: <https://www.blendernation.com/2017/04/22/tutorial-modeling-uv-unwrapping-texturing-mushroom/#prettyPhoto> Acesso em: 20 maio 2017.

NELVA, G. **The Order: 1886 Characters have over 100.000 Polygons, More than Ryse's Marius, but Less Blend Shapes.** 2014. Disponível em: <http://www.dualshockers.com/the-order-1886-characters-have-over-100k-polygons-more-than-ryses-marius-but-less-blend-shapes/> Acesso em: 20 maio 2017.

NORRIS, J. **What does PhysX do?** 2016. Disponível em: <http://www.pcgamer.com/what-does-physx-do/> Acesso em: 01 novembro 2017.

PETKOV, N. **The importance of CPU in gaming.** 2017. Disponível em: <https://sapphirenation.net/importance-cpu-gaming/> Acesso em: 01 novembro 2017.

ROUSE, M. **Judder.** 2010. Disponível em: <http://whatis.techtarget.com/definition/judder> Acesso em: 20 maio 2017.

SCHULTZ, W. **AAA Game.** 2017. Disponível em: <https://www.thoughtco.com/what-is-aaa-game-1393920> Acesso em: 20 maio 2017.

SCIUTTERI, M. **Using a texture Atlas to Optimize Your Game**. 2016. Disponível em: <https://gamedevelopment.tutsplus.com/articles/using-texture-atlas-in-order-to-optimize-your-game--cms-26783> Acesso em: 20 maio 2017.

SHAIKH, F. **Why are GPUs necessary for training Deep Learning models?** 2017. Disponível em: <https://www.analyticsvidhya.com/blog/2017/05/gpus-necessary-for-deep-learning/> Acesso em: 01 novembro 2017.

SIMPSON, R. **Rigging and Skeletal Animation.** 2014. Disponível em: https://bigblackdrawings.wordpress.com/2014/01/09/rigging-and-skeletal-animation/ Acesso em: 20 maio 2017.

TAKAHASHI, D. **PwC: Game industry to grow nearly 5% annually through 2020.** 2016. Disponível em: <https://venturebeat.com/2016/06/08/the-u-s-and-global-game-industries-will-grow-a-healthy-amount-by-2020-pwc-forecasts/> Acesso em: 14 maio 2017.

TRÜMPLER, S. **Render Hell 2.0.** 2015. Disponível em: <https://simonschreibt.de/gat/renderhell/> Acesso em: 20 maio 2017.

UNITY. **Unity User Manual.** 2017. Disponível em: <https://docs.unity3d.com/Manual/index.html> Acesso em: 20 novembro 2017.

UNITY3D. **Optimizing Graphics rendering in Unity Games.** 2017. Disponível em: <https://unity3d.com/pt/learn/tutorials/temas/performance-optimization/optimizing-graphics-rendering-unity-games > Acesso em: 20 maio 2017.

UNREAL ENGINE. **Performance and Profiling.** 2017. Disponível em: <https://docs.unrealengine.com/latest/INT/Engine/Performance/> Acesso em: 20 maio 2017.

VALIENT, M. **Killzone Shadow Fall Demo Postmortem.** 2013. Disponível em: <https://www.guerrilla-games.com/read/killzone-shadow-fall-demo-postmortem> Acesso em: 20 maio 2017.

VENUTA, A. **Implementing Skeletal Animation.** 2014. Disponível em: <http://veeenu.github.io/2014/05/09/implementing-skeletal-animation.html> Acesso em: 20 maio 2017.

OWEN, D. **Killzone: Shadow Fall screenshots are packed with next-gen shine.** 2013. Disponível em: <https://www.vg247.com/2013/10/24/killzone-shadow-fall-screenshots-are-packed-with-next-gen-shine/> Acesso em: 20 maio 2017.

WAUGH, R. **Modern Warfare 3 hits $1billion in 16 days – beating Avatar's record by one day.** 2011. Disponível em: <http://www.dailymail.co.uk/sciencetech/article-2073201/Modern-Warfare-3-hits-1-billion-16-days--beating-Avatars-record-day.html> Acesso em: 14 maio 2017.

ANEXO A - *Storyline*

O jogo trata de uma luta entre fãs de bandas e estilos musicais diferentes, como rock, pop e jazz. Um DJ toca músicas ao fundo de uma pista de dança e os competidores lutam entre si para chegar ao DJ que dará o privilégio ao vencedor de colocar seu estilo musical favorito para tocar na pista. O confronto ocorre seguindo as regras de um jogo de luta tradicional onde cada personagem possui uma barra representando sua vida que ao ser esvaziada representa a derrota do mesmo. Os lutadores estarão caracterizados com roupas e acessórios representando seu estilo musical e bandas favoritas. Que a batalha comece!