UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO UFC VIRTUAL

CURSO DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

LUCAS PARDI CORRÊA

EXPLORANDO O WORKFLOW BLENDER-UNITY PARA O DESENVOLVIMENTO DE ANIMAÇÕES 3D CARTUNESCAS PARA JOGOS

FORTALEZA-CE

2022

LUCAS PARDI CORRÊA

EXPLORANDO O WORKFLOW BLENDER-UNITY PARA O DESENVOLVIMENTO DE ANIMAÇÕES 3D CARTUNESCAS PARA JOGOS

Relatório técnico-científico apresentado ao Programa de Graduação em Sistemas e Mídias Digitais da Universidade Federal do Ceará, como requisito parcial à obtenção do título de profissional em Sistemas e Mídias Digitais. Área de concentração: Modelagem e animação 3D.

Orientador: Prof. Me. Neil Armstrong Rezende.

FORTALEZA-CE
2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C843e Corrêa, Lucas Pardi.
Explorando o workflow blender-unity para o desenvolvimento de animações 3D cartunescas para jogos / Lucas Pardi Corrêa. – 2022.
68 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2022.
Orientação: Prof. Me. Neil Armstrong Rezende.

1. Animação 3D. 2. Cartunesco. 3. Games. 4. Blender-unity. I. Título.

CDD 302.23

LUCAS PARDI CORRÊA

EXPLORANDO O WORKFLOW BLENDER-UNITY PARA O DESENVOLVIMENTO DE ANIMAÇÕES 3D CARTUNESCAS PARA JOGOS

Relatório técnico-científico apresentado ao Programa de Graduação em Sistemas e Mídias Digitais da Universidade Federal do Ceará, como requisito parcial à obtenção do título de profissional em Sistemas e Mídias Digitais. Área de concentração: Modelagem e animação 3D.

Aprovada em: 25/07/2022.

BANCA EXAMINADORA

_____

Prof. Dr. Adriano Anunciação Oliveira
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Natal Anacleto Chicca Junior
Universidade Federal do Ceará (UFC)

_____

Prof. Me. Neil Armstrong Rezende (Orientador)
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Muito obrigado a todos que me ajudaram durante este percurso, ao meu orientador Neil Armstrong, aos professores Adriano Oliveira e Natal Chicca pelo tempo, pelas valiosas colaborações e sugestões.

Um agradecimento muito especial aos meus amigos Adonai e Victor que me apoiaram durante toda a produção do meu trabalho, à minha namorada Amanda que ficou ao meu lado e deixou essa situação mais leve e à minha família por sempre acreditar em mim! Eu amo vocês!

Resumo

O meio de animação é muito rico e a sua adaptação de forma expressiva e cartunesca à mídia de jogos 3D é uma vertente muito interessante. A intenção deste trabalho é justamente se utilizar de recursos provenientes da animação tradicional, como os *smears*, e adaptá-los em um ambiente de jogo 3D, com a utilização de deformações na malha do personagem e a duplicação de membros. Este tipo de prática é comumente usada em filmes de animação, mas também é relevante buscar entender quais passos devem ser tomados para alcançar esse tipo de resultado em jogos. Assim, será exploranda a pipeline de desenvolvimento de animações para jogos 3D, com o objetivo de, ao final, demonstrar o processo de desenvolvimento de animações 3D cartunescas, que se utilizem dos princípios clássicos de animação, para jogos. Este é um trabalho prático, que irá guiar o leitor por todas as etapas deste desenvolvimento que seguirá um workflow Blender-Unity, que engloba desde a concepção do personagem, a modelagem, texturização, *rigging*, animação e implementação das animações para dentro da game *engine*.

Palavras-chave: Animação 3D, Cartunesco, Games, Blender-Unity.

Abstract

The animation medium is very rich and its adaptation in a more expressive and cartoonish way into 3D game media is a very interesting aspect. The intention of this work is precisely to take resources from traditional animation, like smears, and apply them into a 3D game environment, with the usage of deformations in the character's mesh and the duplication of limbs. This type of practice is commonly used in animated films, but it is also relevant to seek understanding of what steps should be taken to achieve this kind of result in games. Thus, the animation development pipeline for 3D games will be explored, with the objective of, in the end, demonstrating the process of developing cartoon 3D animations, that use the classic principles of animation, for games. This is a practical work, which will guide the reader through all the stages of the development, that will follow a Blender-Unity workflow, which encompasses character design, modeling, texturing, rigging, animation and implementation of animations into the game engine.

Palavras-chave: 3D animation, Cartoon, Games, Blender-Unity.

Lista de Figuras

Sumário

1. Introdução

Em 1981, os animadores Ollie Johnston e Frank Thomas, lançam o livro *The Illusion of Life:* Disney *Animation*, baseado nas obra dos principais animadores dos estúdios Disney até então, e que propõe princípios a serem seguidos na busca da criação da **ilusão de vida**, que abrange tanto a fisicalidade do objeto, como o *timing*, o *appeal*, etc. Cada um destes princípios será discutido em maior detalhe mais adiante. Segundo Walt Disney, "Nosso trabalho deve ter fundamento nos fatos para ter sinceridade. A comédia mais hilária sempre tem base em coisas reais." (THOMAS e JOHNSTON, 1981, p. 62)

Mas de que forma deve-se abordar a animação em um objeto 3D? É claro que podem ser criadas poses e feita uma interpolação[1] simples entre elas, mas essa abordagem não é interessante visualmente. Um exemplo de jogo que trás consigo uma personalidade e atrativo muito grande em sua animação é Crash Bandicoot (1996) (figura 1), desenvolvido pela Naughty Dog, para o Playstation. Segundo o co-criador do jogo, Andy Gavin:

> Nós sabíamos que queríamos fazer esse tipo de animação que nunca tinha sido vista em videogames; do estilo Looney Tunes, uma animação distorcida. É um estilo de animação bem elástico e borrachento que é feito na animação tradicional de células. Isso significava que os personagens tinham que realmente ser animados. (DACANAY, 2020)

[1] A interpolação, no contexto da animação 3D, é o processo de criar poses intermediárias entre as poses principais.

Figura 1 - Animação de pulo do personagem Crash Bandicoot

Fonte: Jogo Crash Bandicoot (1996).

O meio de animação é muito rico e a sua adaptação de forma expressiva e cartunesca à mídia de jogos 3D é uma vertente muito interessante. Por isso, é relevante buscar entender quais passos devem ser tomados para alcançar esse tipo de resultado; de quais formas distorcer o modelo 3D para alcançar um efeito similar ao das animações tradicionais. É importante que as pessoas não só saibam que esta também é uma possibilidade, mas que ela pode agregar um quê lúdico à animação e, por conseguinte, ao jogo.

Tendo isso em mente, o projeto visa, de modo geral, demonstrar o processo de desenvolvimento de animações 3D cartunescas para jogos que se utilizem dos princípios clássicos de animação. Para demonstrar isso será desenvolvida uma *tech demo*[2] que consistirá de um personagem em um ambiente 3D onde o jogador poderá executar uma série de ações. De forma mais específica, o projeto busca:

[2]Demonstração técnica.

criar um personagem adequado para o desenvolvimento de animações; trazer uma animação mais convincente, tentando não apenas imitar a realidade, mas expressar a ação o melhor possível; e explorar a pipeline de desenvolvimento de animações para jogos, mais especificamente na *pipeline* Blender-Unity.

2. Referencial Teórico

A seguir serão apresentados os assuntos que serão mais relevantes durante o desenvolvimento deste projeto. Estes irão abranger desde o básico de animação até a composição de modelos 3D, ferramentas usadas para movimentar e deformá-los, além de utilidades para facilitar a vida dos animadores, na forma de proceduralidade.

2.1. Os princípios de animação

Os princípios de animação foram primeiro formulados nos estúdios da Disney nos anos 30. Naquela época havia um time formado pelos “nove anciões” (old nine men), os animadores mais experientes, que se responsabilizavam por dar alma às tão famosas animações da Disney. Dois deles, Frank Thomas e Ollie Johnston, criaram um manual para manter a qualidade dos filmes da Disney nas futuras gerações, seu livro *The Illusion of Life*. Lá estão os direcionamentos para o desenvolvimento de animações mais convincentes.

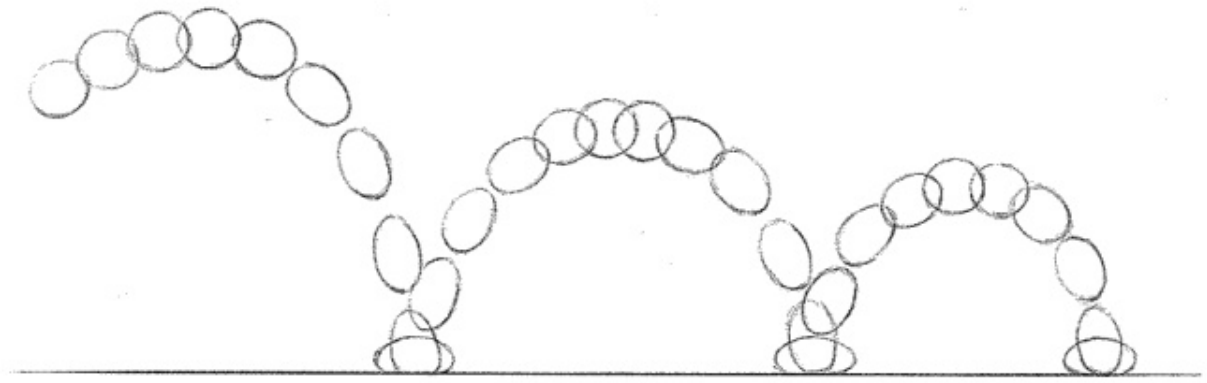
"Há um ingrediente especial em nosso tipo de animação que produz desenhos que aparentam pensar e tomar decisões e agir por conta própria; é o que cria a ilusão de vida." (THOMAS e JOHNSTON, 1981, p. 9).

O livro, que foi lançado em 1981, tornou-se referência sólida para os animadores, que continuam a utilizá-lo até os dias de hoje. Estes princípios são doze e são os descritos a seguir:

2.1.1. Squash and Stretch (Esmagar e Esticar)

Este é o princípio mais básico e provavelmente o mais intuitivo. Ele consiste na deformação do objeto com o objetivo de enfatizar sua velocidade, impulso, potência, peso e massa. Ele faz com que objetos muito rápidos sejam esticados e que sejam esmagados ou comprimidos com o impacto. É importante ressaltar que se um objeto for esticado verticalmente, tem que ser também afinado horizontalmente, de forma a conservar o mesmo volume. A figura 2 mostra o exemplo clássico da bola quicando.

Figura 2 - Bola quicando



Fonte: The Animator 's Survival Kit (2001, p. 39).

2.1.2. *Anticipation* (Antecipação)

O princípio de antecipação agrega em muito à animação, trazendo consigo não só o realismo que uma ação precisaria, como uma potência muito maior, que de outra forma ficaria faltando. Outro benefício de sua utilização é preparar o público para a ação que está por vir, para isso é importante facilitar a compreensão ao máximo possível, para que este possa compreender o que está acontecendo sem ter que assistir mais de uma vez.

"Eles devem estar preparados para o próximo movimento e esperá-lo antes que ele de fato ocorra." (THOMAS e JOHNSTON, 1981, p. 51)

2.1.3. *Staging* (Encenação)

É um dos princípios mais amplos, pois ele se aplica à atuação, ao *timing*, ao ângulo e posição da câmera, à distribuição dos personagens na cena, etc.

Segundo Frank Thomas e Ollie Johnston, a Encenação:

> É a apresentação de qualquer ideia de forma que seja completamente e inconfundivelmente clara. Uma ação é encenada de modo que seja compreendida, a personalidade de modo que seja reconhecível, a expressão de modo que possa ser vista, o clima de modo que possa afetar a audiência.(THOMAS e JOHNSTON, 1981, p. 53)

Assim, ao se estar aplicando este princípio, o objetivo do animador deveria ser o de obter o controle total sobre para onde o público está olhando e as sensações que estes deveriam sentir a cada momento.

2.1.4. *Straight Ahead Action and Pose to Pose* (Ação Direta e Pose a Pose)

Este princípio descreve os dois métodos usados para a animação:
Ação Direta, que é uma sequência contínua de desenhos, um atrás do outro; já Pose a Pose é quando são feitas as poses principais, também conhecidas como *keyframes*, e os intervalos são preenchidos com *in-betweens*.

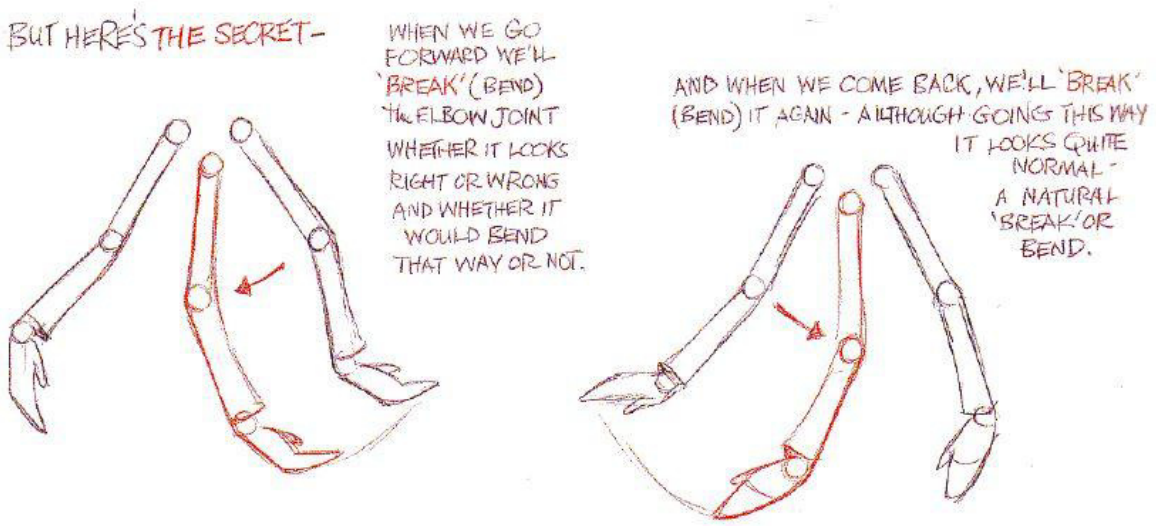
Cada uma é mais apropriada para um tipo de animação, por exemplo, a Ação Direta não é recomendável para desenhar personagens, mas se destaca na animação de formas fluídas como fogo, fumaça e água. Já a Pose a Pose não tem que se preocupar em manter as proporções do modelo, já que estas foram estabelecidas nos *keyframes*.

2.1.5. *Follow Through and Overlapping Action* (Sequência e Ação Sobreposta)

Este princípio descreve dois fenômenos complementares entre si.
A Sequência se refere à quando partes do corpo dão continuidade ao movimento, mesmo depois do corpo principal ter terminado o seu. Isso se dá, por exemplo, quando um personagem para de correr abruptamente, onde ele precisa se recompor antes de chegar ao estado parado, ou ao fazer uma expressão indo para uma versão mais exagerada da pose antes de chegar na mesma.

Ação Sobreposta é o intervalo de tempo entre o movimento do corpo principal e suas outras partes. Podemos observar isso na figura 3, no movimento de um braço ao caminhar, o movimento que inicia pelo braço, depois passa para o antebraço e por fim à mão.

Figura 3 - Movimento do braço



Fonte: The Animator's Survival Kit (2001, p. 151).

2.1.6. *Slow In and Slow Out* (Suavização do Início e do Fim)

Este princípio vem demonstrar que a velocidade deve ser construída gradativamente. Ao iniciar o movimento, deve ir aos poucos acelerando e antes de parar o movimento, ir desacelerando. Para isso, invés de espaçar os *frames* igualmente durante a animação, estes devem estar concentrados onde ocorre a suavização do início e do fim.

Objetos na natureza agem assim, é por isso que uma velocidade linear passa a sensação de ser algo artificial e robótico.

2.1.7. *Arcs* (Arcos)

Este princípio diz respeito à forma do movimento, que em sua grande maioria se dá na forma de arcos na natureza. Por isso é interessante traçar um arco entre os *keyframes*[3] para ter o discernimento na hora de desenhar os *in-betweens*[4] e assim obter um resultado mais natural.

2.1.8. *Secondary Action* (Ação Secundária)

Muitas vezes, a ideia passada em uma animação pode ser reforçada e ampliada através da utilização de uma animação auxiliar. No exemplo de uma pessoa falando, se ela cruzar os braços e olhar para outro lado ela está mostrando

[3] Quadros principais são a guia para animação pose a pose.
[4] Quadros intermediários servem para preencher a lacuna entre dois quadros principais.

desinteresse; Nesse caso a pessoa falando é a ação principal e o resto é ação secundária. A ação secundária sempre vai agregar à ação principal, sem se sobrepor.

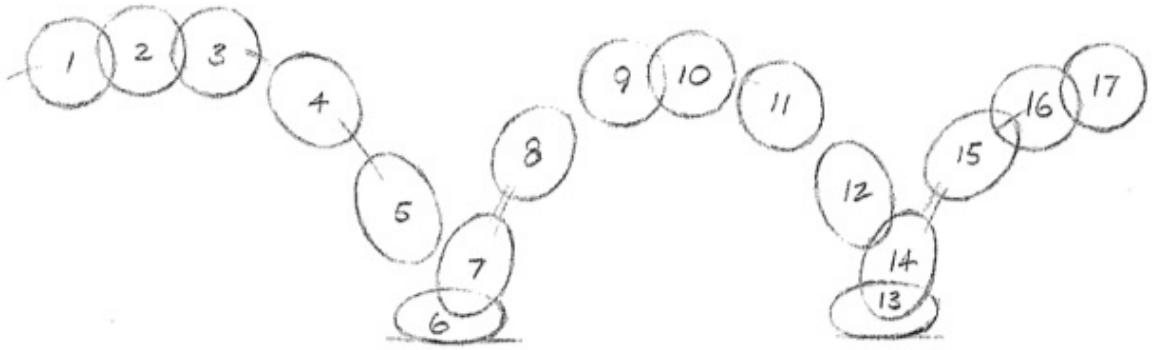
2.1.9. *Timing* (Temporização)

É o princípio que determina a quantidade de tempo que cada ação deve levar. Para isso é importante pensar em quantos *frames* cada ação vai utilizar, porque a quantidade de *frames* por segundo é predeterminada, geralmente sendo animados em uns (24 *frames* por segundo) ou em dois (12 *frames* por segundo). De forma simplificada, quantos mais *frames* uma ação tiver, mais lenta esta vai ser. O inverso também é verdade, quantos menos *frames*, mais rápido vai passar.

"Nem atuação nem atitude poderiam ser representados sem prestar muita atenção no *timing*.", segundo os autores. (THOMAS e JOHNSTON, 1981, p. 64)

A figura 4 representa quantos frames a animação da bola terá e quão espaçados estão uns dos outros.

Figura 4 - Bola quicando enumerada



Fonte: The Animator's Survival Kit (2001, p. 94).

2.1.10. *Exaggeration* (Exagero)

Na Disney, no começo havia uma confusão entre os animadores quando Walt pedia mais realismo, mas então criticava por não estar exagerado o suficiente. Um dos artistas analisou corretamente quando falou que o que ele quis dizer não era "realismo", mas algo que fosse mais convincente, que tivesse mais contato com as pessoas. (THOMAS e JOHNSTON, 1981) Um desenho mais convincente no sentido de fazer a ação com intenção, com mais vontade.

2.1.11. *Solid Drawing* (Desenho Sólido)

Através deste princípio, procura-se passar a impressão de que as formas estão em um espaço tridimensional, com peso, profundidade e equilíbrio.

Segundo o artista, animador e diretor de cinema, Grim Natwick, "Quanto melhor você souber desenhar, mais fácil será para você. Você terá que desenhar o personagem em todas as posições e de todos os ângulos, pois se você não souber e tiver que encenar de outro ângulo, é muito restritivo e leva mais tempo."

Um problema que surge é o *twin*, ou *twinning*, que é quando os braços ou as pernas estão não só paralelas, como fazendo a mesma ação. Nesses casos é importante quebrar a simetria da pose, para trazer mais ênfase ao equilíbrio e dar mais dinamismo ao personagem.

2.1.12. *Appeal* (Atrativo)

O nome pode ser enganoso, pois atrativo, neste contexto, não significa algo necessariamente bonito, mas algo que seja interessante ao olhar, que tenha um charme, simplicidade e que comunique bem o que o personagem quer passar.

> Um desenho fraco carece de atrativo. Um desenho que é complicado demais ou difícil de ler carece de atrativo. Design pobre, formas desajeitadas, movimentos estranhos, todos têm baixo atrativo. Expectadores gostam de assistir algo que os atraia, seja uma expressão, um personagem, um movimento, ou toda uma situação da história. Enquanto o ator tem carisma, o desenho animado tem atrativo. (THOMAS e JOHNSTON, 1981, p. 68)

2.2. *Smear Frames*

Um engano comum sobre *smears* é que estes são parecidos com *squash and stretch*, mas a verdade é que estes são contrários, pois enquanto que no *squash and stretch* você tem que se preocupar em manter a massa do objeto, no *smear* esta pode ser quebrada para representar de forma mais exagerada a energia da ação. É interessante considerar essa possibilidade, pois como disse Tina Nawrocki, a animadora do jogo Cuphead em entrevista para Polygon (STOEBER, 2021), se você tiver um frame muito distante do próximo, a animação parece entrecortada, enquanto que se você tiver um *smear*, a animação flui melhor, porque imita o que o olho vê.

O animador Toniko Pantoja, em seu vídeo-ensaio, define as funções do *smear frame* da seguinte forma:

> Pode enfatizar e fortalecer a aparência de um movimento, pode ser usado para estilizar movimentos rápidos e, provavelmente a razão mais importante de todas, ele pode descrever completamente os movimentos que são invisíveis a olho nu! Veja, o cérebro humano não consegue processar cada movimento ou quadro de algo em movimento, especialmente se algo está se movendo a uma velocidade incrivelmente alta. Eles saberão o contexto da ação, mas não necessariamente o sentirão, porque o cérebro humano precisa de informações que descrevam a força do movimento em um curto espaço de tempo. (PANTOJA, 2018)

Ele toca num conceito muito importante, característica essencial do *smear*, o tão alusivo **sentir**, e elabora sobre a ideia: “Podemos ver muitos quadros por segundo na vida real, mas ver não é o mesmo que perceber e sentir um movimento muito rápido. Às vezes, podemos precisar fundir ideias para sentir esse tipo de movimento.” (PANTOJA, 2018)

Atualmente existem diversas formas de representar os *smears*, já que vários artistas experimentaram diferentes soluções para tal problema. “Os curtas do começo dos anos trinta eram cheios de ações rápidas, derrapagens e batidas. Cada animador procurou por uma forma melhor ou mais engraçada de desenhar o efeito preciso para fortalecer sua ação.” (THOMAS e JOHNSTON, 1981, p. 116)

A figura 5 mostra uma sequência de *smears* sobre o rosto do Pateta, um dos personagens mais queridos da Disney.

Figura 5 - Faces do Pateta sofrendo níveis de smear diferentes
níveis de *smear*

Fonte: The Illusion of Life (1981, p. 117).

"Este efeito de vibração é alcançado por animar múltiplas imagens em uma única imagem. Não é preciso seguir um padrão específico, e as necessidades particulares de cada cena irão determinar como se lida com ela." (THOMAS e JOHNSTON, 1981, p. 117)

Além da técnica de duplicar o objeto, os *smears* podem ser representados com linhas de ação, com rastros de cores, com a distorção do próprio objeto, dentre outros. Não há jeito inerentemente errado de representar um *smear*, porém, um cuidado que deve ser tomado é o de que duas imagens não devem estar no mesmo local em desenhos sucessivos, senão o olho irá sentir uma pausa na ação. (THOMAS e JOHNSTON, 1981, p. 117)

2.3. Animação 3D

É importante ressaltar que tanto o uso dos princípios de animação quanto o uso de *smear frames* não se restringe ao 2D. Pelo contrário, a utilização dos princípios de animação abrange as animações 3D, sendo ainda a base para uma animação fluida e interessante. Há diversos exemplos, tanto no cinema quanto em jogos, como o já citado Jak and Daxter (2001), que no exemplo do seu pulo (figura 6), se utiliza dos princípios de *stretch and squash*, Sequência e Ação Sobreposta, arcos, para citar alguns.

Figura 6 - Fases do pulo de Jack

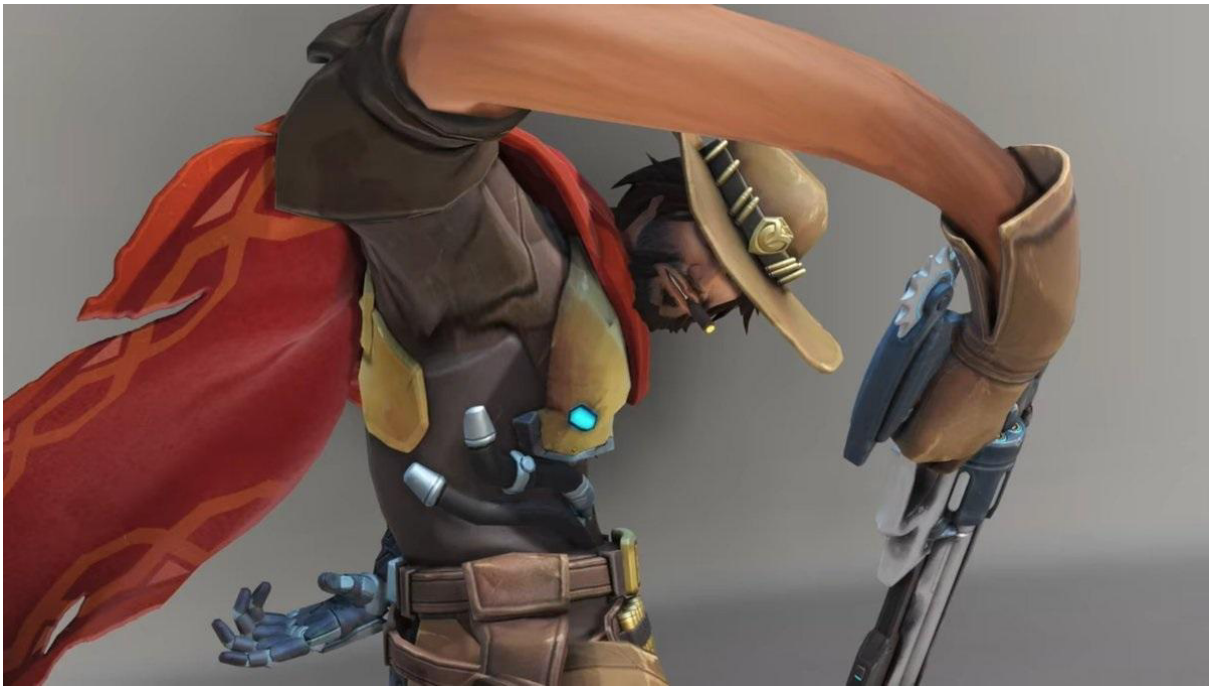
Fonte: Jogo Jak and Daxter (2001).

Embora seja menos intuitivo, muito mais complicado e trabalhoso, alguns animadores 3D se dão ao trabalho de adicionar *smear frames* à sua escolha estilística.

Da mesma forma que as animações em 2D, os *smears* em 3D não têm que manter a forma, regularmente são quebradas por alguns instantes para enfatizar o movimento. Dependendo do estilo, a utilização desta técnica pode ser mais sutil, como no exemplo do jogo da Blizzard, Overwatch, no qual os personagens sofrem deformações de forma a acentuar o dinamismo da animação. Porém, segundo o animador principal, Jesse Davis, em entrevista para a Polygon (STOEBER, 2021), seu estilo não é cartunizado, mas também não é realista: é baseado na fisicalidade do mundo real com mais 20% extra de estilo. Dessa forma, como não é dado um foco para os *smear frames*, estes podem passar despercebidos por muitos dos jogadores, mas o importante é que eles os sintam.

Isto se torna possível graças aos chamados "*noodle joints*", que são juntas adicionais que agregam outra camada de informação, que não sofre as mesmas restrições que o resto do *rig*, podendo escalonar de maneira desigual e, assim, distorcer os personagens (figura 7).

Figura 7 - *Smear frame* do personagem McCree



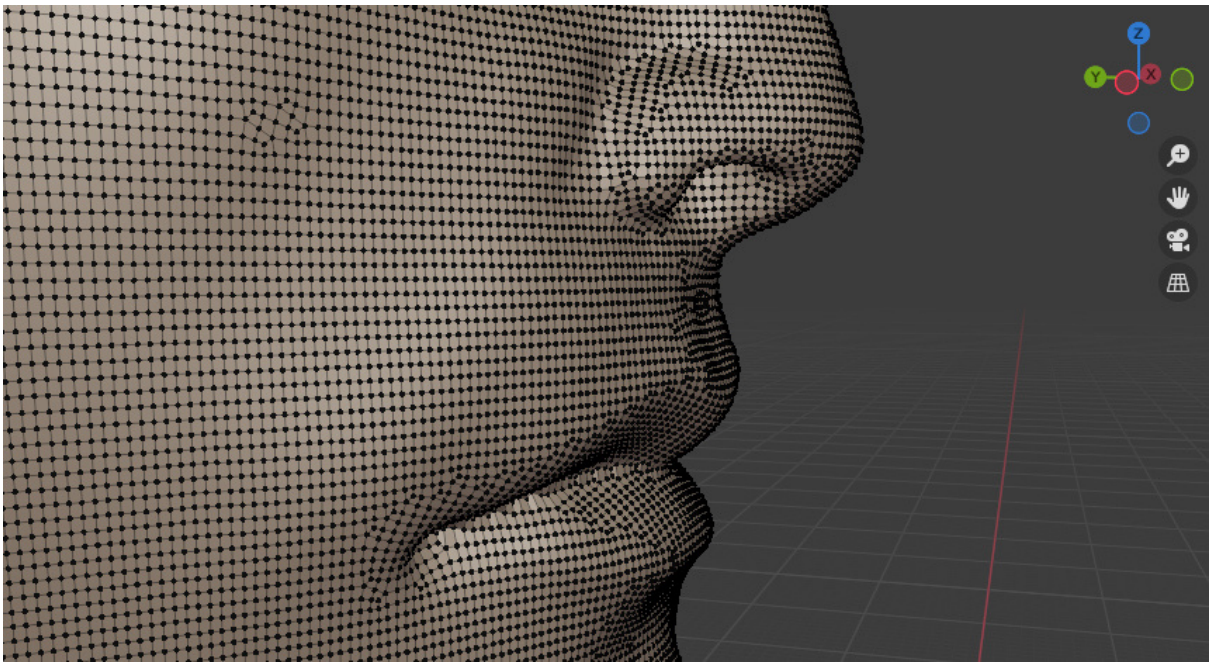
Fonte: Jogo Overwatch (2016).

2.3.1. *Mesh* (Malha)

Dentro do ambiente 3D o que existe não são objetos de fato, com fisicalidade, mas volumes vazios formados por pontos no espaço. Ligando os pontos, se obtém arestas e então faces. Estas faces são poligonais e podem ser compostas por três (*tris*), quatro (*quads*) ou mais arestas (*n-gons*). O conjunto dessas faces é o que configura a malha de um objeto, que pode ter densidade de centenas de milhares de faces (VAUGHAN, 2012).

A figura 8 mostra um personagem com grande densidade de malha.

Figura 8 - Malha densa em polígonos



Fonte: Elaborada pelo autor.

O problema surge na hora de renderizar, quando um objeto é muito pesado, que tenha um número muito grande de vértices, o seu processamento se torna muito custoso e, por consequência, demorado. Pode não ser um problema para renderizar uma única imagem, mas quando se trata de um filme de animação, que é composto por milhares de frames, ou em um jogo, onde a renderização ocorre em tempo real, um cuidado especial é imprescindível.

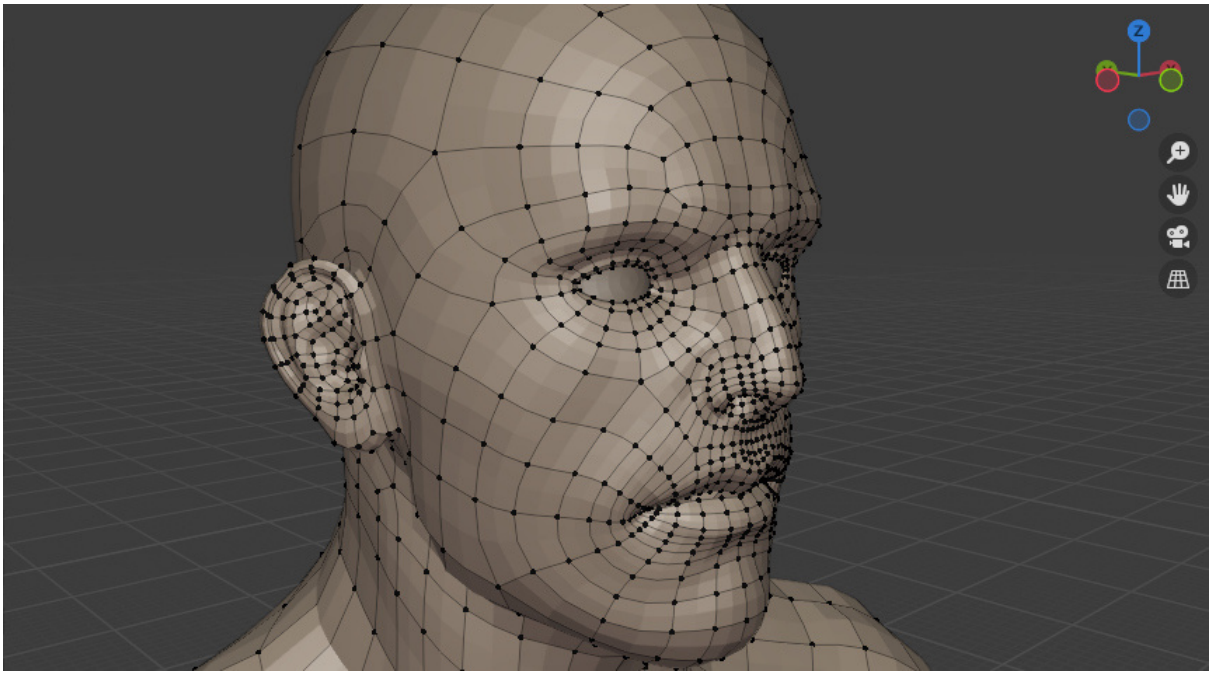
Para solucionar isso, é necessário retrabalhar a topologia do objeto com o propósito de deixá-la mais leve, num processo congruentemente intitulado de retopologia. Embora a topologia também seja referida como o fluxo dos polígonos, o

layout dos vértices e arestas de uma malha também têm um papel importante em dizer se sua topologia é considerada boa ou ruim. Embora haja um acordo sobre muitos dos aspectos do que se define como boa topologia, não há um verdadeiro padrão na indústria ou conjunto de diretrizes a serem seguidas. (VAUGHAN, 2018)

Um cuidado que deve ser tomado ao fazer a topologia de objetos cuja malha será deformada, como no exemplo de personagens, é a criação de *edge loops* que dêem suporte à deformação. Quando usado corretamente, os *edge loops* podem simular os músculos em um personagem, produzindo deformações mais convincentes. (VAUGHAN, 2018)

A figura 9 mostra um personagem após o processo de retopologia.

Figura 9 - Retopologia do rosto



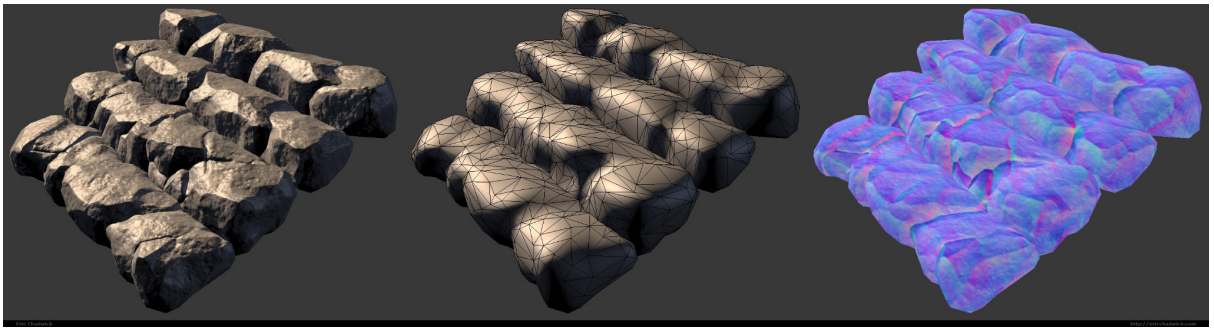
Fonte: Elaborada pelo autor.

Uma vez feita a topologia, o nível de detalhe da mesma pode ser modulado através do *subdivision surface* (subdivisão da superfície), que é um modificador que você pode adicionar à malha. (VAUGHAN, 2012)

Quando a topologia está pronta, é possível partir para o UV *unwrap*, ou desembrulhar da UV, que é uma representação 2D da malha 3D, e é a partir dela que são feitas as texturas. Um processo que pode ser feito é o *bake*, seja das

normais, da oclusão do ambiente, do metálico, etc., no qual é impresso sobre a UV de uma malha mais leve os detalhes da malha mais pesada (figura 10).

Figura 10 - Um modelo com mapa de normais, a malha sem o mapa e apenas o *normal map*, respectivamente.

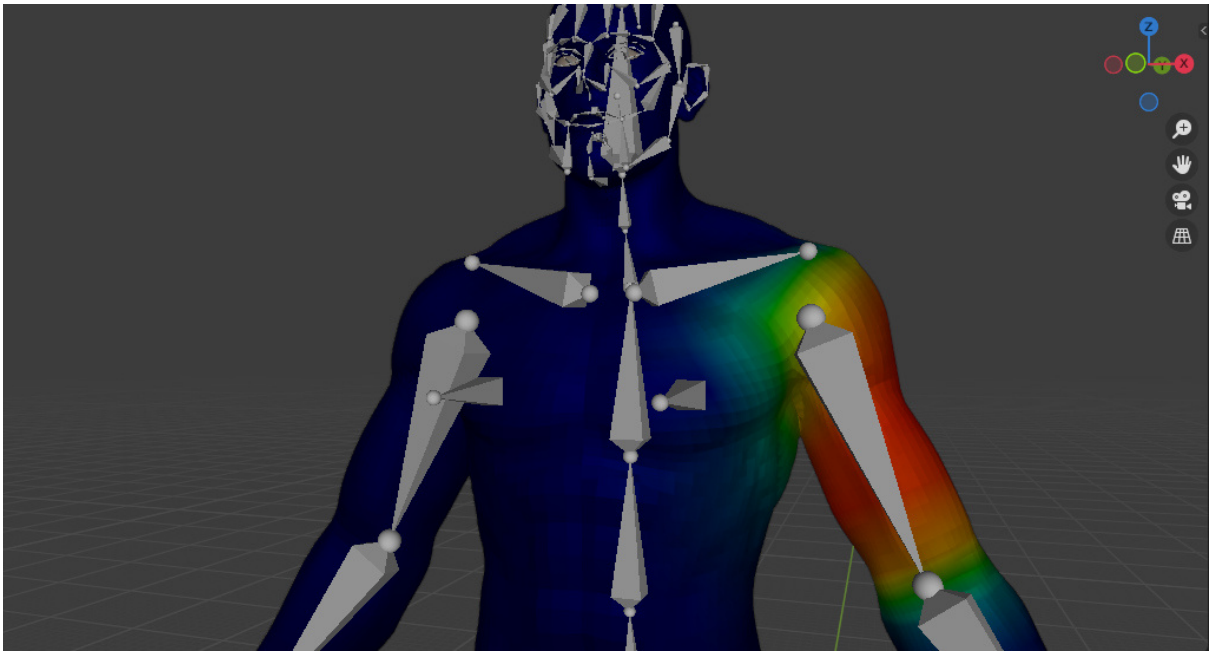


Fonte: Eric Chadwick.

2.3.2. *Rig* (Esqueleto)

O *rig*, também conhecido como *armature* ou esqueleto, é o que nos permite animar objetos 3D sem deformar de fato a malha. Ele é formado por um ou mais *bones* (ossos) e cada um deles irá exercer certa influência sobre a malha. O nível de influência de cada *bone* pode ser modificado através do *weight paint* (figura 11), que altera o *vertex group*, ou grupo de vértices, que cada um irá afetar. (VAUGHAN, 2012)

Figura 11 - Visualização do peso que o *bone* tem sobre a malha

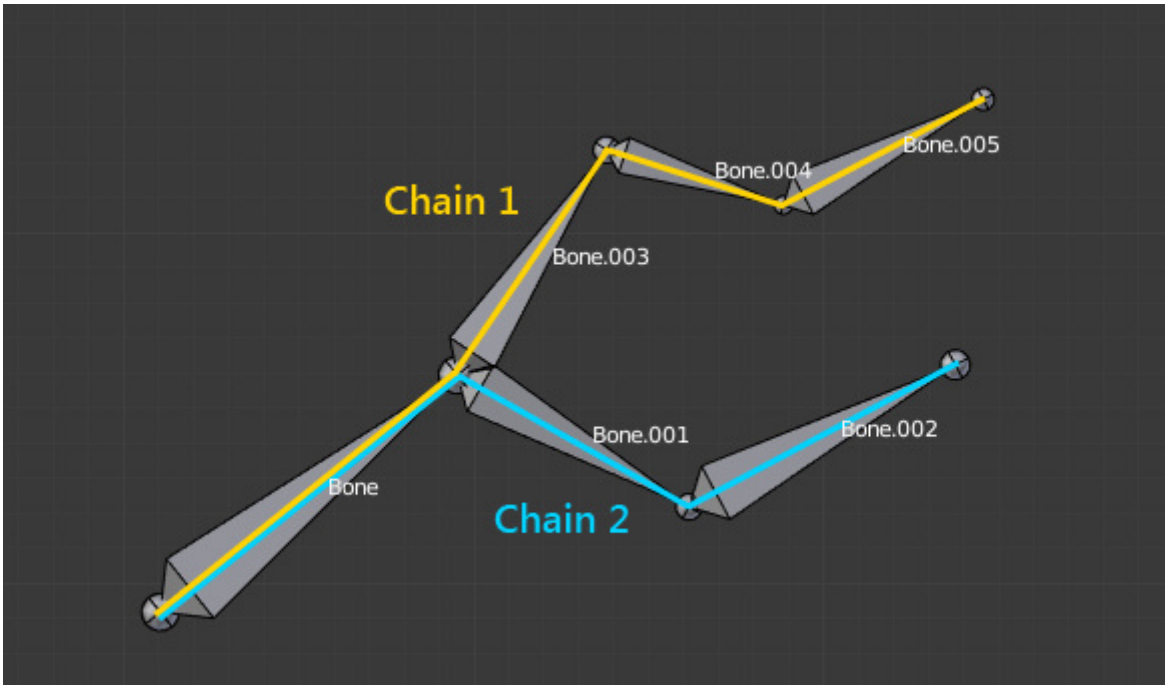


Fonte: Elaborada pelo autor.

Existe no Blender um *add-on* que facilita muito na hora de fazer *rigs* de qualquer personagem, chamado *rigify*, que traz várias opções de *rig* base, de humano, humanóide, quadrúpedes e até aves. Estes são ótimos como base, podendo ser ajustados para melhor se adequar às necessidades. A vantagem do *rig*, frente a um esqueleto real, é que seus *bones* podem ser rotacionados, movidos e escalonados. (Blender manual, 2021)

Embora estes não precisem estar conectados por uma relação de parenteamento, que é o processo pelo qual os filhos herdam as transformações feitas no pai, as configurações mais naturais e úteis implicam que alguns ossos estejam relacionados, formando as chamadas “cadeias de ossos” (figura 12), que criam estruturas semelhantes a membros. (Blender manual, 2021)

Figura 12 - Um esqueleto com duas cadeias de ossos



Fonte: Blender manual, 2021.

2.3.3. *Kinematics* (Quinemáticas)

As cadeias de ossos são um tópico essencial para a movimentação e subsequente pose do objeto, que é a base para a animação 3D, seja esta através da cinemática direta (*Forward Kinematics*) ou inversa (*Inverse Kinematics*). Em outras palavras, a animação 3D consiste de uma série de poses, distribuídas na linha de tempo, e suas respectivas interpolações.

2.3.3.1. *FK*

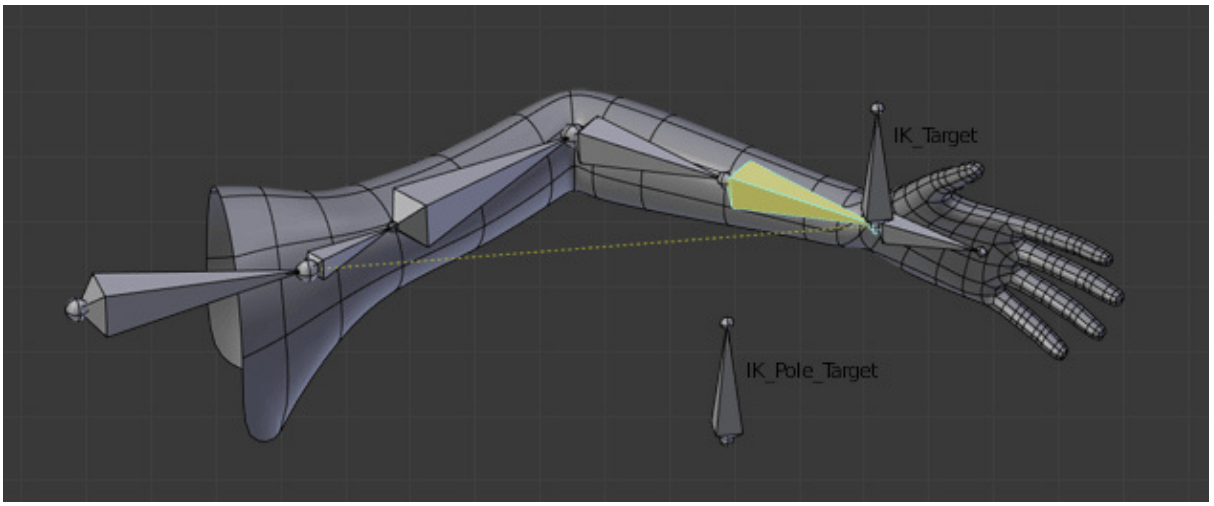
A cinemática direta, ou *Forward Kinematics*, é o processo no qual se utiliza a sequência da cadeia de ossos para posar o objeto. Nela, a posição de um osso é baseada na sequência desta cadeia de ossos, pois, como já foi explicado sobre o *rig*, os filhos herdam as transformações do pai. Então, para calcular a posição de um dedo, é preciso fazer o caminho, desde o quadril, espinha, ombro, braço, antebraço, mão, até chegar no dedo. Isso significa que cada um desses *bones* deve ser modificado individualmente para chegar à pose desejada (Blender manual, 2021). Este é um processo muito laborioso.

2.3.3.2. *IK*

O *Inverse Kinematics*, ou cinemática inversa, vem com o intuito de simplificar esse processo de animação, tornando possível fazer animações mais avançadas com menos esforço.(Blender manual, 2021)

Esta permite que você posicione o último osso em uma cadeia de ossos e os outros ossos são posicionados automaticamente. É como mover o dedo de alguém fazendo com que seu braço o seguisse. Assim, é possível fazer mudanças minúsculas e precisas nas posturas no final da cadeia sem a necessidade de ajustar primeiro todos os *bones* pais. (Blender manual, 2021) Um detalhe importante para o funcionamento correto do *IK*, é a criação de um *pole target*, ou objetivo, que marca a direção para a qual a junta irá se dobrar (figura 13).

Figura 13 - Exemplo de *Ik* aplicado a um braço

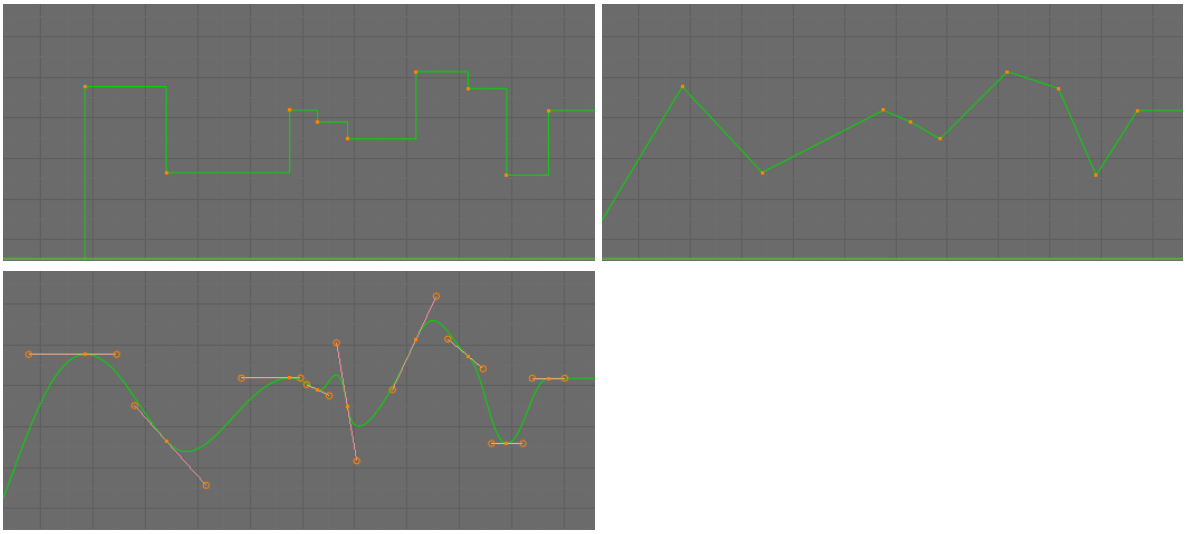


Fonte: Blender manual, 2021.

2.3.4. Interpolação

As interpolações são o cálculo feito pelos programas de animação para fazer a transição entre uma pose e a seguinte. Esta pode ser **constante**, na qual ela pula de uma pose à outra sem transição alguma, podendo ser muito útil na hora da blocagem, para acertar o *timing*; **linear**, uma interpolação simples e com velocidade constante entre as duas poses, que pode ser extrapolada para obter uma curva linear; e **bézier**, a mais útil e poderosa das interpolações, produzindo curvas suavizadas, ou seja, animações suaves (figura 14). (Blender manual, 2021)

Figura 14 - Exemplos de interpolação constante, linear e bézier



Fonte: Blender manual, 2021.

2.3.5. *Constraints* (Restrições)

As restrições são uma forma de controlar as propriedades de um objeto, como localização, rotação e escala, usando valores estáticos simples, como os de limite, ou outro objeto, chamado de *target* ou alvo. (Blender manual, 2021)

Elas podem ser utilizadas para animar um objeto através de alvos, em um processo conhecido como animação indireta. Isto ocorre porque as propriedades do objeto estão restritas às propriedades do alvo, então, ao mover o alvo, o objeto também será movido. (Blender manual, 2021)

Outra forma de animar usando restrições é associar o movimento de um alvo a uma ação. Assim, ao movimentar o alvo você pode alternar entre o estado inicial e o movimento completo.

As restrições podem ter diversas funções, como fazer com que uma pessoa olhe para outra, permitir que as rodas de um carro girem juntas, e tornar mais fácil para uma mão segurar o cabo de uma espada e fazer com que esta balance com a mão. (Blender manual, 2021)

2. 3.6. *Shape Keys*

As *Shape Keys* ou formas chave. São uma forma de modificar a malha original para que esta possa se adaptar melhor às diferentes necessidades, podendo alternar entre formas chaves e até combiná-las.

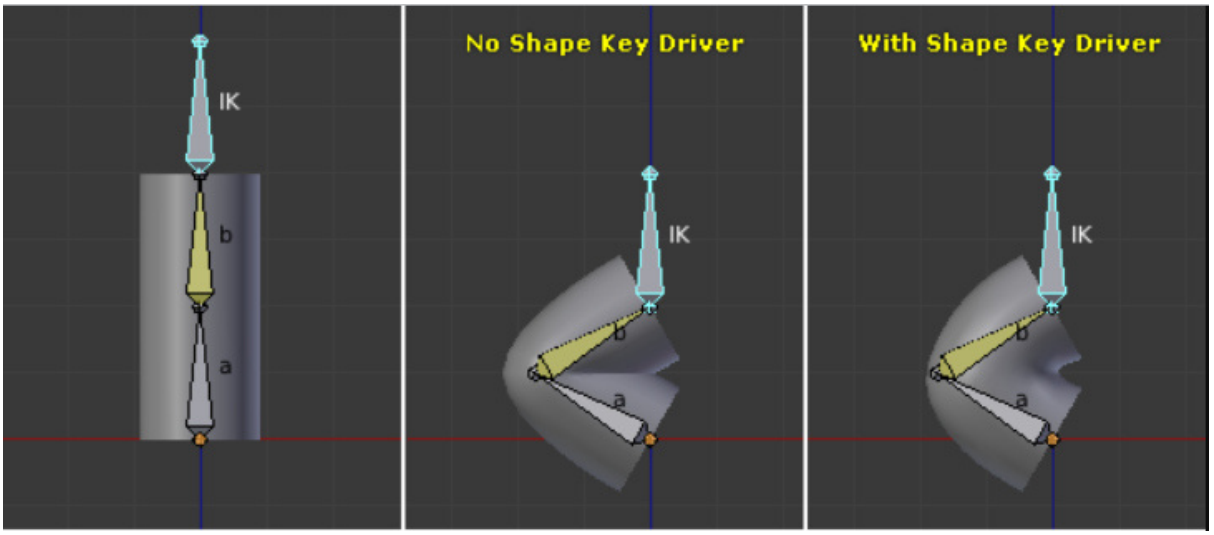
> Os casos de usos mais populares para as formas chave são a animação facial do personagem e o ajuste e refinamento de um esqueleto. Eles são particularmente úteis para modelar partes moles orgânicas e músculos onde há uma necessidade de mais controle sobre a forma resultante, do que o que pode ser alcançado com a combinação de rotação e escala. (Blender manual, 2021)

2.3.7. *Drivers*

Drivers são *scripts* cujo objetivo principal é controlar propriedades com outras propriedades. Por exemplo, quando a rotação de um objeto é controlada com a localização de outro objeto. (Blender manual, 2021)

O exemplo a seguir utiliza um driver em conjunto com uma *shape key* para que, à medida que o *bone* do *IK* vai descendo, a influência do *shape key* irá aumentando (figura 15).

Figura 15 - Deformação de um cone através de um drive em conjunto com uma shape key



Fonte: Blender manual, 2021.

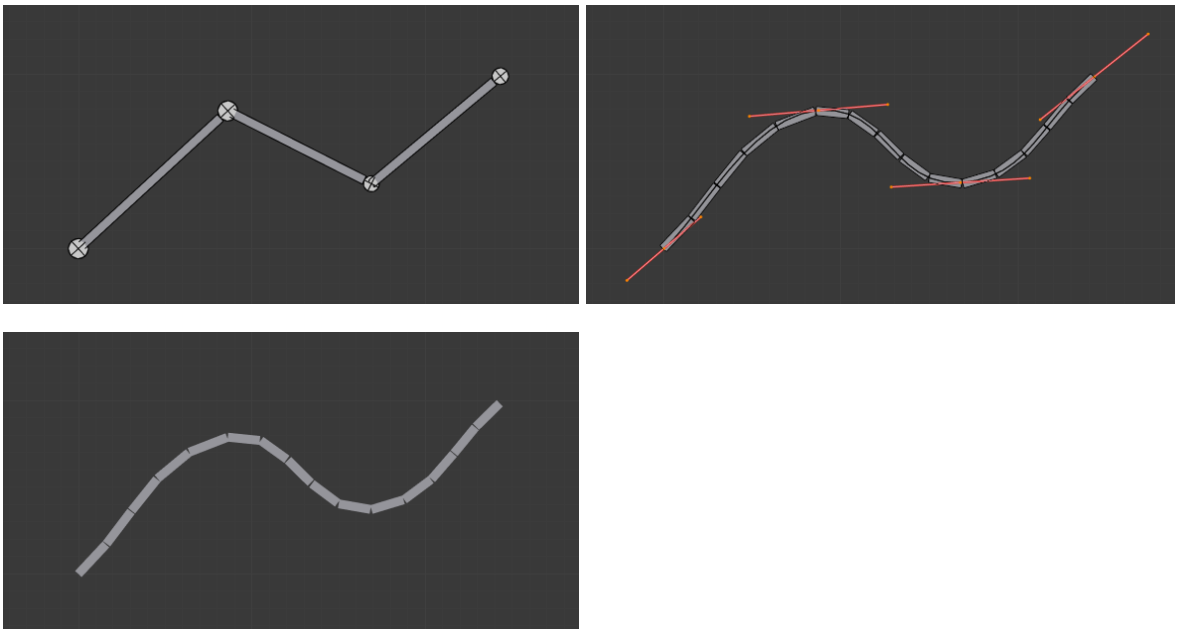
2.3.8. *Bendy Bones*

Bendy Bones, ou ossos dobradiços, são uma maneira fácil de substituir longas cadeias de muitos pequenos ossos rígidos. Um uso comum para os *Bendy Bones* são a modelagem de colunas vertebrais e ossos faciais. (Blender manual, 2021) Esta técnica será muito utilizada, pois ela é essencial para alcançar o objetivo deste projeto, elaborar animações 3D cartunescas, uma vez que ela irá permitir a

deformação do personagem de maneira que não seria possível de outra forma, alcançando assim resultados mais convincentes.

O Blender trata o osso como uma seção de uma curva de Bézier que passa pelas articulações dos ossos. A sua forma pode ser controlada usando uma série de propriedades, ou indiretamente através dos ossos vizinhos (ou seja, primeiro filho e pai). As propriedades constroem alças em cada extremidade do osso para controlar a curvatura (figura 16). A sua forma inicial pode ser definida no Modo de Edição como uma pose de repouso desse osso. Isso é útil para características faciais curvas, como sobrancelhas ou bocas curvas. (Blender manual, 2021)

Figura 16 - Bending Bone visto em *Edit Mode*, *Pose Mode* e *Object Mode*



Fonte: Blender manual, 2021.

2.4. Produção de jogos em Unity

Esta seção aborda algumas ferramentas importantes para o *workflow* de desenvolvimento de animações para jogos, dentro da *engine* Unity. Estas servem diferentes papéis, porém o objetivo geral é o de facilitar o trabalho dos animadores. Isto se dá graças a proceduralidade, que possibilita não só o reaproveitamento das animações (evitando retrabalho), como a combinação e o ajuste destas de forma dinâmica, alcançando um resultado mais polido.

2.4.1. Animation retargeting

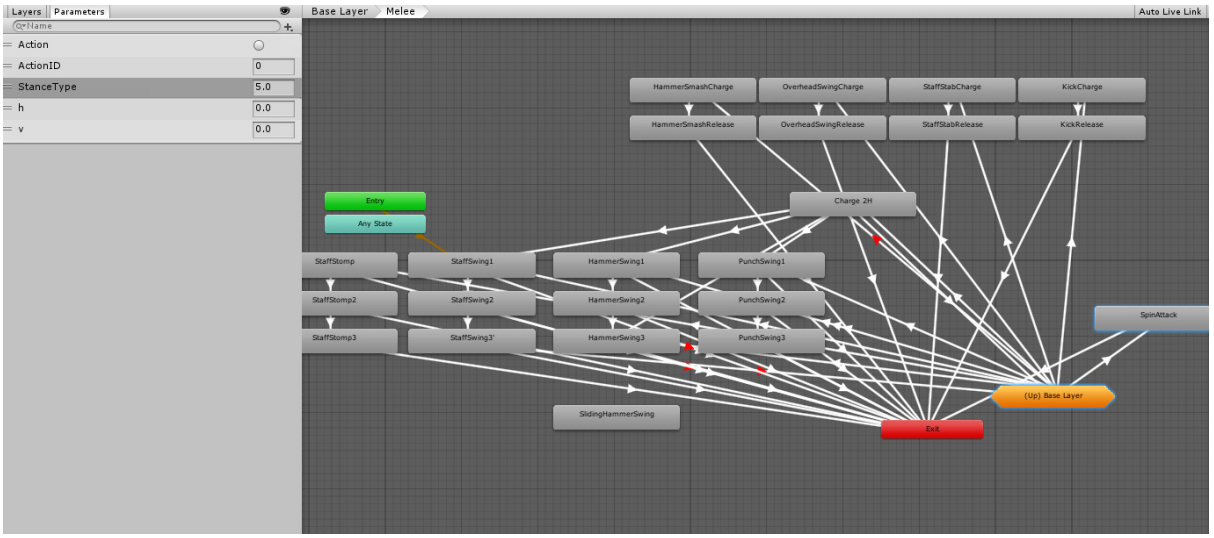
Na Unity é preciso especificar qual tipo de *rig* está sendo utilizado, seja este humanóide ou genérico. Graças a isso, desde que as diversas animações estiverem usando o *rig* humanoide, estas podem ser reutilizadas para quaisquer personagens que tiverem o mesmo tipo de armadura. Isto é ótimo, pois pode economizar muito tempo que seria gasto em trabalho redundante.

2.4.2. Animator (Animador)

O animador é um gerenciador de animações na Unity. Ele serve a função de definir quais animações serão realizadas a cada momento, sob quais circunstâncias e como será feita a transição entre elas. Esta ocorre na forma de estados interligados entre si.

Dependendo da complexidade do jogo, se este possuir um número muito grande de animações, o animador se torna um emaranhado de ligações entre os diferentes estados (figura 17). Para combater esse problema foram desenvolvidas algumas ferramentas.

Figura 17 - Aglomerado de estados em um animador



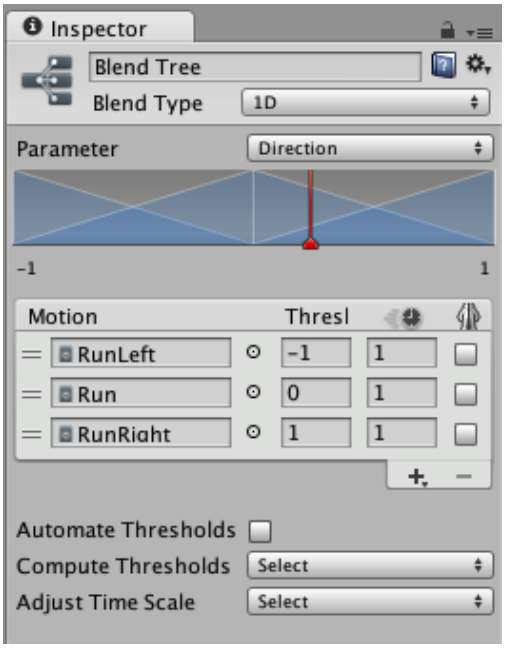
Fonte: usuário omg_ketchup no site Reddit.

2.4.2.1. Blend Trees

As *Blend Trees* ou árvores de mesclagem combinam estados semelhantes, como a caminhada, a corrida, e suas variações de direção, em um único estado. Estas podem ser de dois tipos: mesclagem 1D, que combina os movimentos filhos

de acordo com um único parâmetro (figua 18); e mesclagem 2D, que os combina de acordo com dois parâmetros. (Unity Manual, 2021)

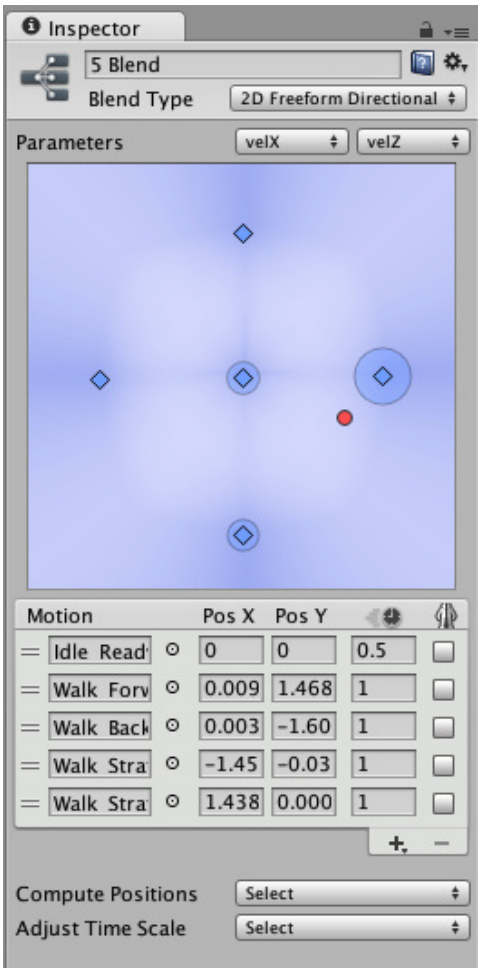
Figura 18 - Inspector de uma *blend tree* 1D



Fonte: Unity Manual, 2021.

A mesclagem 2D tem modos diferentes que diferem em como a influência de cada movimento é calculada (figura 19). Estes são: **2D *Simple Directional***, melhor usado quando seus movimentos representam direções diferentes, como para frente e para trás, porém, não deve haver mais de um movimento na mesma direção, como andar e correr para a frente; **2D *Freeform Directional***, que é usado quando seus movimentos representam direções diferentes, no entanto, permite ter vários movimentos na mesma direção, como andar e correr para frente; **2D *Freeform Cartesian***, que é melhor usado quando seus movimentos **não** representam direções diferentes, pois seus parâmetros X e Y podem representar conceitos diferentes, como velocidade linear e velocidade angular; e **Direto**, que permite que o usuário controle o peso de cada nó diretamente, sendo útil para formas faciais ou combinação aleatória de poses *idle*. (Unity Manual, 2021)

Figura 19 - Inspector de uma *blend tree* 2D *Freeform Directional*



Fonte: Unity Manual, 2021.

2.4.2.2. Animation Layers (Camadas de animação)

As *Animation Layer* ou camadas de animação servem para gerenciar máquinas de estado complexas. Em cada camada, você pode especificar a máscara e o tipo de mesclagem; a máscara especifica as partes do corpo nas quais a animação será aplicada, já o tipo de mesclagem como a animação é aplicada. É selecionado o tipo **Substituir** para usar a animação desta camada no lugar daquelas em camadas anteriores; enquanto que o tipo **Aditivo** irá mesclar a animação nesta camada à animação das camadas anteriores. Para que a combinação aditiva seja bem-sucedida, a animação na camada aditiva deve conter as mesmas propriedades das camadas anteriores. (Unity Manual, 2021)

Há situações nas quais é conveniente reutilizar uma máquina de estado em camadas diferentes. Um exemplo seria o personagem tenha estados diferentes,

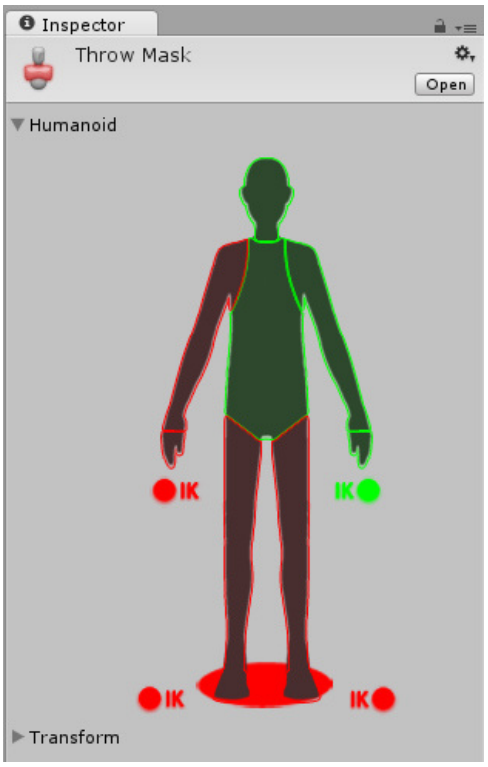
como feliz e triste, e cada um deles realiza as mesmas ações com versões diferentes da mesma animação. Isso significa que a camada sincronizada não tem sua própria definição de máquina de estado, sendo uma instância da origem da camada sincronizada. Todas as alterações feitas no *layout* ou estrutura da máquina de estado na visualização da camada sincronizada são feitas na origem da camada sincronizada. As únicas alterações exclusivas da camada sincronizada são as animações selecionadas usadas em cada estado. (Unity Manual, 2021)

Outro modo de utilizar as camadas é em conjunto com as *Avatar Masks*, que são muito úteis para atribuir animações diferentes às partes do corpo. Isto é muito útil para os animadores, pois evita o retrabalho de ter de fazer todas as combinações possíveis de movimento. Neste *workflow* você pode fazer uma animação caminhando e uma animação segurando um objeto e combiná-las de modo a criar uma nova animação na qual o personagem está andando e segurando o objeto.

2.4.2.3. Avatar mask

Avatar masks são máscaras que você criar no *rig* e depois pode utilizar nas camadas, de forma que uma animação se aplique somente à parte do *rig* desejada. Caso o modelo seja humanoide, este pode ser editado simplesmente selecionando quais partes você não quer que se movam. A alternativa é selecionar manualmente quais *bones* serão mascarados (figura 20).

Figura 20 - Inspector de uma *Avatar Mask*



Fonte: Unity Manual, 2021.

3. Metodologia

3.1. *Design Thinking*

Este projeto será dividido em duas fases. A primeira seguirá o processo do *Design Thinking*, de Ambrose e Harris (2010), em que será idealizado e criado o personagem 3D, desde a concepção do conceito até o *Rigging* e *Weight painting*. Nela serão realizadas as seguintes etapas:

3.1.1. Definição

O projeto tem início com o processo de *briefing*, ou seja, onde são coletadas as informações e objetivos a serem alcançados. Nele precisam ser respondidos os 5 *W's*: (*who*) quem é o cliente; (*what*) que solução o cliente está pensando; (*when*) quando o projeto precisa estar pronto; (*where*) onde será utilizado; (*why*) por que o cliente acha necessário. (AMBROSE; HARRIS, 2010, p. 15)

O resultado deste processo é o *brief* com o qual será possível elaborar uma proposta de design, que será utilizada como base para a criação do produto.

3.1.2. Pesquisa

A etapa de pesquisa irá divergir do processo de design thinking, no qual esta é fundamentalmente voltada para o mercado, sendo realizadas duas pesquisas qualitativas com base na proposta de design.

A primeira tratará de pesquisar mecânicas, ações, estados, poses, etc. que um personagem poderia realizar/ser submetido. O resultado desta pesquisa é uma lista, que será útil no processo de design do personagem.

A segunda é uma pesquisa estética, na qual serão buscadas influências para o design do personagem, sejam estas de estilos, traços, penteados, roupas de época, cores, etc.. O resultado desta será a criação de um *moodboard*[5], que auxiliará no processo de design do personagem.

3.1.3. Ideação

É a hora de expandir sob a proposta de design, aprofundando o conceito do personagem. Isto se dá através da alternância entre a escolha dos elementos

[5] O moodboard é uma ferramenta utilizada para ajudar a visualizar uma ideia ou um conceito a partir de uma coletânea de referências.

estéticos e das mecânicas que serão aplicadas ao personagem, de forma a que estes conversem entre si.

3.1.4. Prototipação

Esta é uma etapa fundamental para poder experimentar diferentes possibilidades e o que cada uma delas trás. Assim, serão feitos rascunhos das animações, em 2D, para ter uma noção de que tipos de deformação são necessárias. Em seguida, será testado no Blender como se dá a deformação da malha, seja através da utilização de *edge loops*, *bones* adicionais na armadura, *shape keys*, etc.

3.1.5. Seleção

A partir dos resultados da etapa anterior, são escolhidas quais das possibilidades melhor se adequam a quais casos. Assim, já fazendo um planejamento prévio de como serão implementados.

3.1.6. Implementação

Com base no planejamento realizado na etapa anterior, o produto é de fato concretizado. Este processo é o mais direto, se inicia pela modelagem 3D do personagem, seguida pela texturização, o *rigging*, o *Weight painting* e a criação de *shape keys*. Finalizando esta etapa, temos um modelo pronto para ser utilizado na produção das animações.

3.1.7. Aprendizagem

Nesta etapa acontece a contemplação de todo o processo, um momento para analisar o que funcionou ou deixou de funcionar, o que poderia ter sido feito diferente e de que forma este método poderia ser utilizado para projetos futuros.

3.2. *Feature-Driven Development* (FDD)

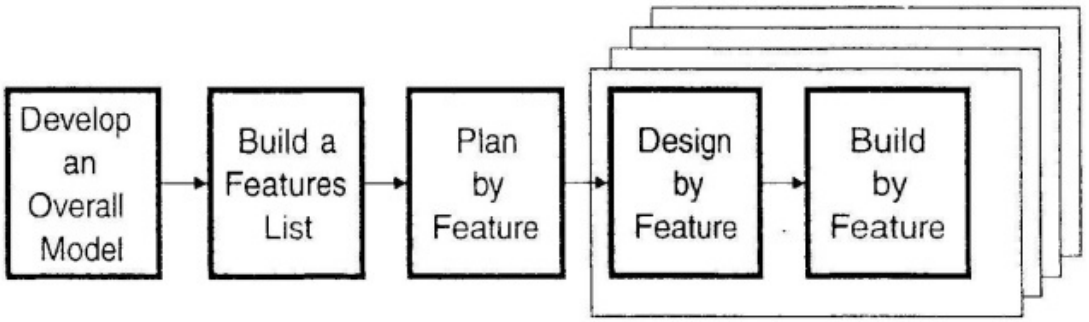
A segunda etapa será desenvolvida utilizando o modelo *Feature-Driven Development* (FDD), ou Desenvolvimento Dirigido por Funcionalidades, pois segundo Posvolski et al. (2014, p. 7), o “FDD se mostra adequada ao desenvolvimento de jogos devido a melhorias ágeis no desenvolvimento de software

que proporciona, tais como o desenvolvimento iterativo, tempo de desenvolvimento aprimorado, testes integrados, e resposta rápida às mudanças de requisitos.”

Segundo Palmer e Felsing (2002), o FDD consiste de cinco etapas (figura 21): (1) a criação de um modelo de domínio pelas equipes de desenvolvimento e domínio, sob a orientação de um modelador de objetos experiente (*Chief Architect*); (2) a criação de uma lista de *features* e criação de *features sets*; (3) sequenciar as *features sets* em planos de alto nível e atribuí-los aos programadores chefe; (4) os programadores chefe selecionam um grupo com os proprietários de classes correspondentes às *features* dessa iteração, que elaboram os diagramas de sequência detalhados para os recursos e escreve prólogos de classes e métodos, os quais são inspecionados; (5) os proprietários de classes adicionam o código real para suas classes, testam a unidade, integram, e realizar uma inspeção de código.

É nas etapas 4 e 5 que ocorrerá a iteração, quer dizer que estas serão repetidas para cada um dos *features sets*.

Figura 21 - Etapas do Feature Driven Development



Fonte: A Practical Guide to Feature Driven Development (2002, p. 57).

Além de requerer um time de desenvolvedores que ocupem uma série de papéis, estas etapas foram criadas com o *workflow* de desenvolvimento de software em mente. Assim, tendo em vista que o foco deste projeto é o desenvolvimento de animações 3D e sua implementação em jogo, é necessário alterar as etapas, para que estas se adequem às necessidades e limitações deste projeto.

Dessa forma, cada *feature* se refere a uma animação e os *features sets* são conjuntos de animações. Desenvolvendo dessa premissa, temos as seguintes etapas:

3.2.1. Construir uma lista de Features

Nesta primeira etapa será aproveitada da primeira fase do projeto, na qual foi criada uma lista com todas as animações que virão a ser trabalhadas. Com base nela são criados os conjuntos de animações, que agrupam animações similares, como por exemplo um grupo caminhada, composto por uma caminhada para a frente, para a direita e para a esquerda.

Ao final deste processo é importante definir também qual a prioridade de cada um dos conjuntos de animações definidos.

3.2.2. Planejar por Feature

Sequenciar os conjuntos de animações em planos de alto nível, estabelecendo metas de escopo, prazo e qualidade do projeto. É uma etapa importante para verificar o estado do projeto ao final de cada estágio, para ver como o projeto está progredindo em relação ao plano original.

3.2.3. Projetar por Feature

A partir dos rascunhos de animações, feitos na fase anterior do projeto, elaborar os animatics em 2D do conjunto de animações dessa iteração. Usando eles base, fazer a blocagem das animações no modelo 3D do personagem.

É importante, então, realizar uma inspeção da animação para avaliar a sua qualidade segundo os princípios de animação de Thomas e Johnston (1981). A importância desta avaliação ocorrer já na etapa de blocagem é de corrigir o que for necessário e melhorar o possível o quanto antes, para evitar um retrabalho desnecessário.

3.2.4. Construir por Feature

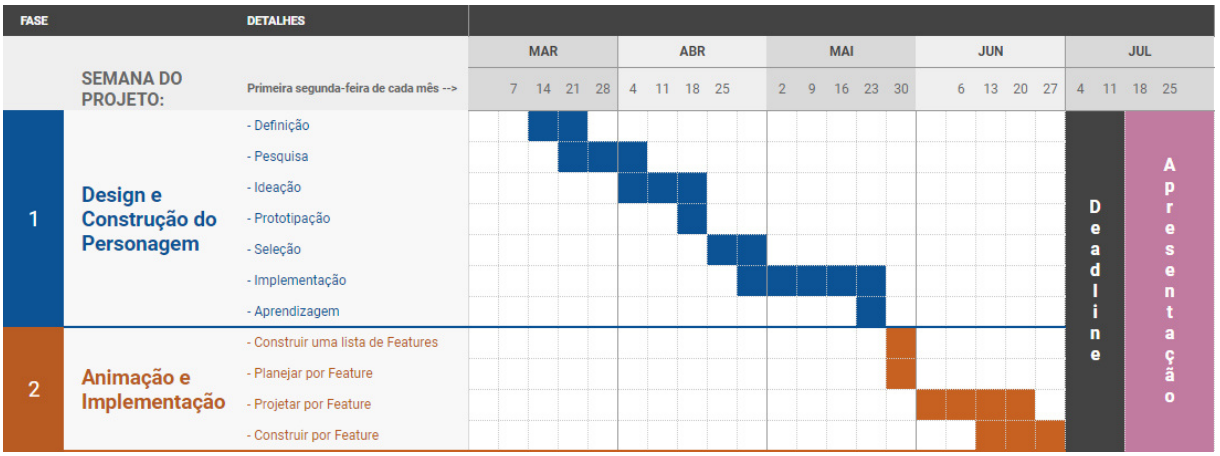
Aplicar a interpolação nas animações e fazer o seu refinamento. A seguir, elas são exportadas para a engine de jogos. Elas serão agregadas à movimentação do personagem através da integração dos novos estados ao *animator*, utilizando de *animation trees* e camadas de animação, criando *avatar mask's* quando for adequado. Se for necessário, pode ser adicionado um componente de IK à armadura, buscando através da proceduralidade trazer um maior nível de acabamento.

Ao final desta etapa é importante, além de testar as alterações feitas no produto, realizar uma inspeção do projeto Unity, para conferir que além de funcionar este esteja organizado e compreensível.

4. Cronograma

O cronograma deste trabalho de conclusão de curso tem início em conjunto com o semestre letivo de 2022.1, no dia 16 de Março. Como já explicado na metodologia, o projeto aconteceu em duas fases. A primeira, de **design e construção do personagem**, ocorreu ao decorrer dos dois primeiros terços do semestre. Já a segunda fase, de **animação e implementação**, sucedeu no terço seguinte. Onde inicialmente, havia mais tempo reservado para a segunda fase, devido a alguns atrasos no cronograma, este teve que ser repensado. Assim, as semanas restantes seriam utilizadas para o desenvolvimento de um ciclo de desenvolvimento de conjuntos de animações(figura 21).

Figura 22 - Cronograma de atividades



Fonte: Elaborada pelo autor.

5. Processo

5.1 *Design Thinking*

O desenvolvimento do projeto tem início com a fase de criação do personagem, a qual foi desenvolvida com a utilização do design thinking de Ambrose & Harris. Esta é dividida em sete etapas.

5.1.1 Definição

O projeto tem início com o processo de *briefing*, ou seja, onde são coletadas as informações e objetivos a serem alcançados; segundo **Ambrose & Harris** o processo de *briefing* tem que responder uma série de perguntas:

- Você compreende o que o cliente está pedindo?
- O cliente entende o que ele está pedindo?
- Você concorda com a definição dos termos?
- O brief tem alguma falha?
- Você consegue comprir com as expectativas do cliente?

Tendo isso em mente, foi decidido que deveria ser desenvolvido um personagem; que este deveria ser completamente modelado, texturizado e *rigged*. O personagem foi inicialmente idealizado como um objeto de estudo, da utilização de técnicas de animação 2D em personagens 3D e sua inserção em ambientes de jogo.

Assim, a malha do personagem não deve ser muito densa, tendo em vista que o processamento da animação é em tempo real, característica inerente da mídia dos videogames. Pois, a cada frame do jogo seria necessário recalcular a posição de cada vértice da malha no ambiente 3D. Isso no contexto onde uma malha densa, que pode chegar às dezenas de milhares de tris[6], implica em um processamento gráfico muito pesado.

Ele não foi desenvolvido tendo uma jogabilidade em mente, e sim como veículo para transmitir, de forma mais convincente, a dinâmica do movimento. Para isto é necessário o uso de ações rápidas, pois nelas o uso de técnicas oriundas da

[6] Conjuntos de três vértices.

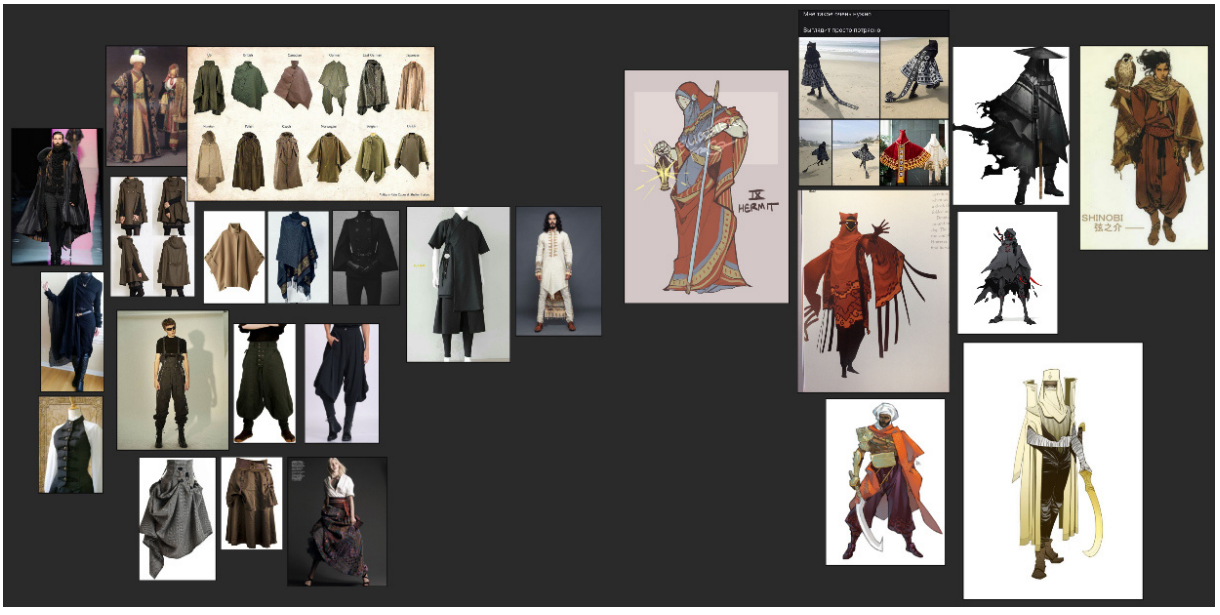
animação tradicional se destacam por representar aquilo que o olho vê e não apenas o que realmente está ali, trazendo consigo o *feeling* do movimento.

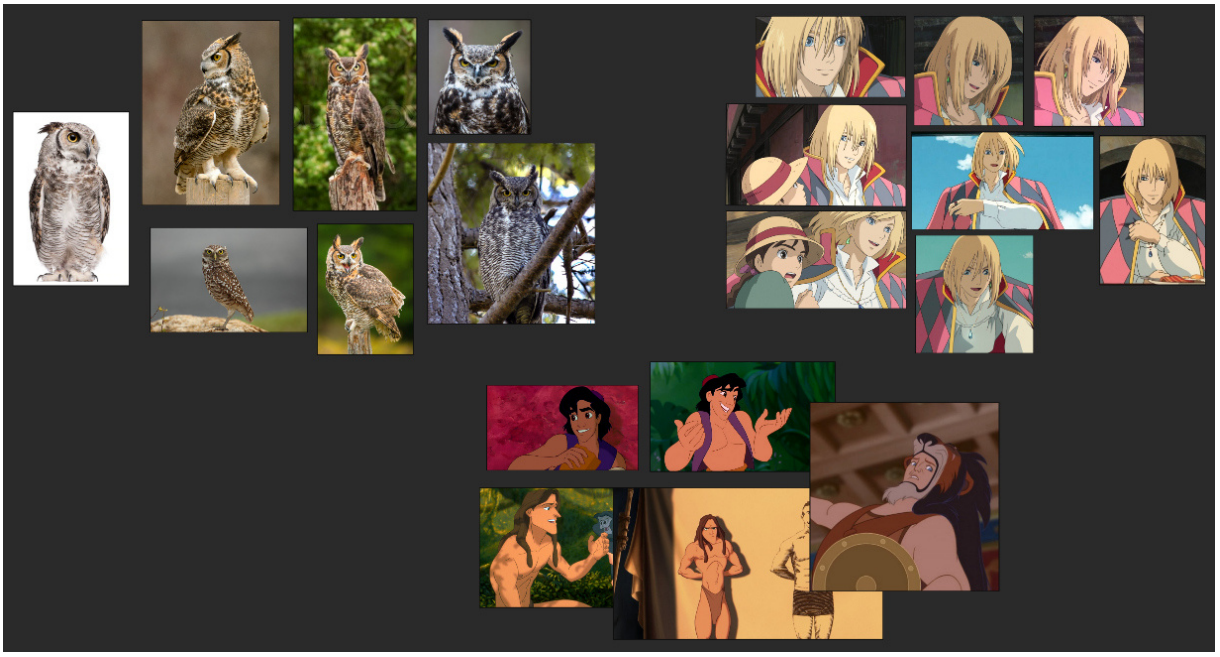
5.1.2 Pesquisa

Com base nas informações que foram levantadas no *brief*, percebemos que haveria uma grande liberdade criativa neste projeto. Partimos então para o desenvolvimento do conceito do personagem. Para isso, foi criado um *moodboard* (figura 23), ou seja, um painel com várias imagens, que seriam adicionadas que seriam selecionadas e adicionadas à medida que fossem pertinentes ao design. Passando por vestimentas, detalhes, estilo e índole.

À medida que foram sendo adicionadas novas imagens, foram surgindo padrões. A ambientação do personagem, então, seria em algum lugar no oriente médio, em um tempo antigo. O estilo do personagem seria remetente ao dos filmes da época do renascimento da Disney, que engloba filmes como Hércules, Aladdin e Tarzan. Por fim, a caracterização do personagem iria se assimilar à de Owl, do filme do estúdio Ghibli, “O Castelo Animado”. Ou seja, ele seria um indivíduo belo, vaidoso, misterioso, sagaz e com alguma ligação com o místico. A partir disso, foi criado o conceito de o personagem ser um djinn, mais popularmente conhecido como gênio; uma criatura mítica associada ao exotérico.

Figura 23 - *Moodboard*





Fonte: Elaborada pelo autor.

5.1.3 Ideação

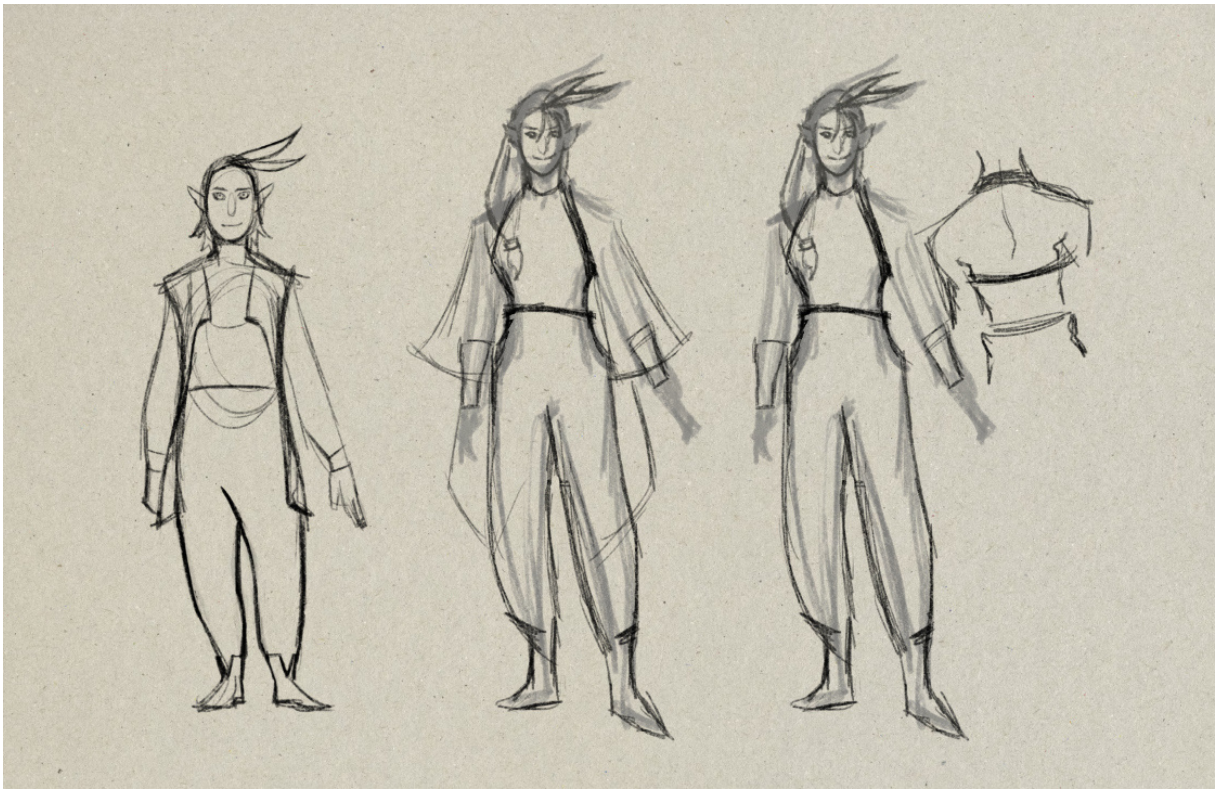
O desenvolvimento do design do personagem passou por três propostas principais (figura 24):

Na primeira, o personagem era um pouco mais jovial, seu traço mais arredondado; Ele tinha os cabelos curtos e arrepiados atrás, para quebrar um pouco com a silhueta. Parte do design que se manteve até o final, sua caracterização principal, os seus olhos amendoados, suas orelhas pontudas e as penas na cabeça, que servem para quebrar a simetria do personagem.

A segunda proposta tenta focar na silhueta do personagem, que traria um visual mais maduro, com seu traço mais alongado. A ideia era que, em conjunto com a sua vestimenta, ele remetesse a uma coruja. Visualmente, quando ele pulasse ou fizesse algum movimento com os braços, a sua roupa (que poderia ser um manto, um poncho ou uma capa) abrisse no formato de asas. Outra adição a esse design é o do cabelo comprido, que adicionava à ideia dele ser uma pessoa vaidosa.

Na terceira, o personagem trouxe consigo as características do anterior, porém o seu físico estaria em maior destaque, este que teria um formato de ampulheta, com os ombros largos e a cintura fina. Esta decisão foi tomada não só para trazer a ênfase à caracterização do personagem como alguém muito vaidoso e que quer expor a sua beleza, mas para evidenciar as diversas deformações decorrentes da animação.

Figura 24 - Três propostas de design do personagem, em ordem



Fonte: Elaborada pelo autor.

Para a escolha de cores foi decidida a utilização de uma harmonia complementar dividida, o vermelho, amarelo e laranja no corpo em contraste com o verde dos olhos.

5.1.4 Prototipação

Com base na *seleção de animações e da blocagem das mesmas*, foi possível definir que os lugares onde haverá maior necessidade da utilização de deformações será nas extremidades. Tendo isso em mente, foi testado a deformação da malha ao utilizar o rig gerado pelo rigify, onde ficou claro que há necessidade de uma certa densidade de malha para que a deformação tenha um aspecto mais natural, não deixando em evidência a topologia da malha.

Além das deformações, foi testado também o uso de duplicatas da malha, não apenas de onde encaixá-las, mas também formas de escondê-las, pois na maior parte do tempo estas não serão visíveis. Chegou-se ao resultado de que ao

parentear[7] a malha a um *bone* auxiliar, ligado ao rig principal, este pode ser escalonado a cerca de 1% de seu tamanho original, ficando recluso dentro da malha original.

5.1.5 Seleção

Foi importante levantar esses pontos previamente para que eles sejam levados em consideração ao desenvolver o personagem, pois a ausência desses testes acarretaria no atraso das etapas. Afinal, o fluxo de trabalho seria interrompido pela necessidade de resolver um problema que já deveria ter sido previsto.

5.1.6 Implementação

Com base nos dados levantados, tem início a fase de desenvolvimento do personagem. Esta irá abranger desde a modelagem, textura, *rigging*, até o *weight painting* e a adição de *shape keys*.

5.1.6.1 Modelagem

A partir do *concept* do personagem, foi possível passar para a etapa da modelagem, que foi toda desenvolvida utilizando o Blender. Esta teve início com a escultura do personagem, criada utilizando o modo de escultura. Para este projeto, o modelo foi criado do zero, para mostrar todas as etapas na produção de um personagem para animação em jogos. Porém, a utilização de *base mesh*, em partes ou no todo, pode ser muito eficiente em um ambiente de produção, onde seja necessário produzir vários personagens em um tempo limitado.

O processo tem início pela blocagem das formas, que é feita a partir do instanciamento e posicionamento das primitivas, sejam estas a esfera, cilindro ou cubo. Nesta etapa não é necessário adicionar detalhes, mas apenas criar uma noção de proporção, da relação entre o tamanho da cabeça do resto do corpo, por exemplo. Mas, sempre é algo que pode ser modificado mais adiante.

A partir dessas primitivas, desde que haja malha disponível, pode-se refinar mais a forma. Quando ocorrer da malha distorcer, de modo aos tris ficarem visivelmente esticados, você pode escolher uma das alternativas a seguir: utilizar a ferramenta de *remesh*, que irá recalcular toda a malha uniformemente e você pode

[7] Embora pareça uma conjugação indevida, este é um uso oficial utilizado pela própria versão portuguesa do manual do Blender, no tópico Parenteamento.

escolher a distância de cada vértice, em outras palavras, a densidade da malha; ou você pode habilitar a opção de *dyntopo* (*dynamic topology*), que vai recalculando a densidade da malha dinamicamente à medida que for sendo esculpido, com base na distância de vértices selecionada.

Com base nisso, o mais indicado é a modelagem segmentada das diferentes partes. Isso implica que cada parte é um objeto separado, pois assim pode-se trabalhar com maior foco e liberdade em cada segmento. Quando um conjunto de partes está pronto para a próxima etapa, eles podem ser agrupados em um único objeto, mas é importante que seja feito o *remesh*, pois dessa forma eles se tornarão uma malha unificada.

Após isso se inicia a etapa de refinamento das formas, dando mais detalhe ao modelo, buscando se aproximar do *concept* o máximo possível. Uma ferramenta muito útil para o detalhamento é a chamada de *box hide*, que oculta a área da malha selecionada, dessa forma podendo focar em uma parte de cada vez. É importante lembrar que não será aplicada a simetria na área oculta, por isso esse método é ideal para a criação de detalhes assimétricos, ou então em conjunto de um modificador de espelhamento.

Para fazer a vestimenta do personagem pode ser selecionada a área desejada e duplicá-la, para então esculpir separadamente até chegar no resultado desejado. Por exemplo, se for uma blusa, deve ser selecionado o tronco e parte dos braços, que serão as mangas. Outro método, que também pode ser utilizado para a modelagem de acessórios e *props*, é a partir de uma primitiva (*box modeling*) ou de um único vértice (*polygon modeling*), alcançar um resultado o mais aproximado o possível da forma desejada. Modificadores como o *mirror* e *subdivision surface* são aliados nessa hora e assim que aplicados podem se adicionar ainda mais detalhes à malha.

5.1.6.2 Retopologia

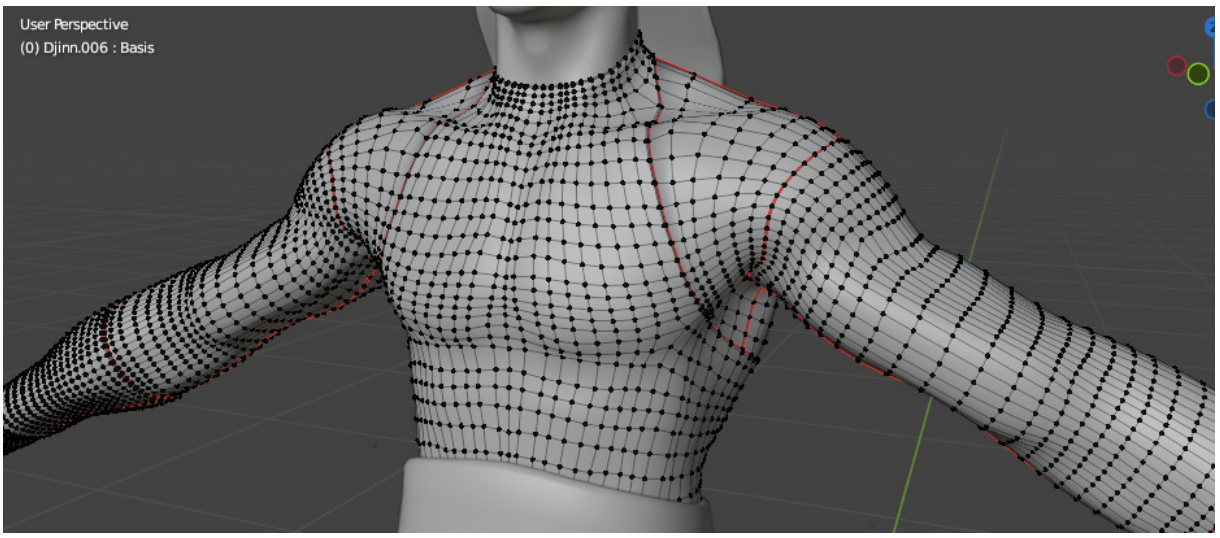
Como já foi mencionado anteriormente, uma malha muito pesada torna o processamento gráfico da animação, que em jogos ocorre em tempo real, algo muito pesado. Para solucionar esse problema, existe um processo conhecido como retopologia, onde é criada uma malha, muito mais leve e fácil de deformar, com base na malha mais densa. Para termos um ponto de comparação, ao utilizar o *remesh* você obteria uma malha uniforme, porém pouco eficiente ao sofrer deformações,

pois elementos auxiliares à animação como os *edge loops* não estão presentes. O que quer dizer que para a finalidade de um objeto que não sofra deformações é uma estratégia viável.

Quando é feita a retopologia de uma área que irá sofrer deformações é muito importante ter em mente o uso de *edge loops*, pois estes irão auxiliar a manter a forma. Quando usado corretamente, os *edge loops* podem simular os músculos em um personagem, produzindo deformações mais convincentes. Um exemplo muito comum é o da retopologia facial, onde deve haver um *edge loop* ao redor de cada olho e outro ao redor da boca. Isso auxilia na movimentação e criação de expressões mais realistas. Um outro exemplo é nos nós dos dedos, onde a presença dos *edge loops* impede que a malha seja deformada de forma tão brusca. (VAUGHAN, 2018)

Um outro tópico importante a ser discutido é a junção de distintas malhas, formando conjuntos. Estes vão variar com as necessidades únicas de cada projeto. Por exemplo, uma prática muito utilizada é a separação da cabeça do resto do corpo, pois geralmente as expressões faciais demandam uma densidade maior na malha. Não é aconselhável a utilização de sobreposição de malha ao redor da dos ombros e peito, pois no contexto da otimização para animação essas áreas estão sob a influência de vários *bones* distintos e pode se tornar muito complicado ajustar os pesos adequadamente. No caso do projeto, esse problema surgiu à tona mais adiante na produção, e exigiu que parte do modelo fosse completamente retrabalhado, tornando o que era o corpo, o top e os braceletes em uma malha única (figura 25).

Figura 25 - Retopologia do tronco em uma malha única



Fonte: Elaborada pelo autor.

5.1.6.3 UV

As UVs são, a grosso modo, uma forma de representar o objeto tridimensional de forma bidimensional. Graças a isso, é possível projetar imagens sobre a malha. Estas imagens podem não ser apenas a cor projetada sobre o objeto, mas trazer em si mais informações a respeito das propriedades físicas do objeto. Este é o princípio utilizado nas texturas PBR (*Physically Based Rendering*).

Para começar a trabalhar com as UVs, é preciso primeiro demarcar onde ocorrerão as *seams* (ou costuras) do objeto, tendo sempre de ocultá-las o melhor possível. Isto porque a *seam* é o lugar de encontro entre duas partes da textura e se não for tomado o devido cuidado, esta ficará aparente. Uma outra ideia é a de dividir a malha em diferentes ilhas, com base nos materiais utilizados, o que facilitará em muito na texturização.

Uma vez feito o processo de *unwrapping,* onde a malha é desembrulhada num plano 2D, é possível alterar a forma e o posicionamento das ilhas UV da forma desejada. É importante, nesse momento, ter em mente qual a importância de cada parte do modelo, para que o seu tamanho, em outras palavras a quantidade de detalhe a sua disposição, seja refletido na UV.

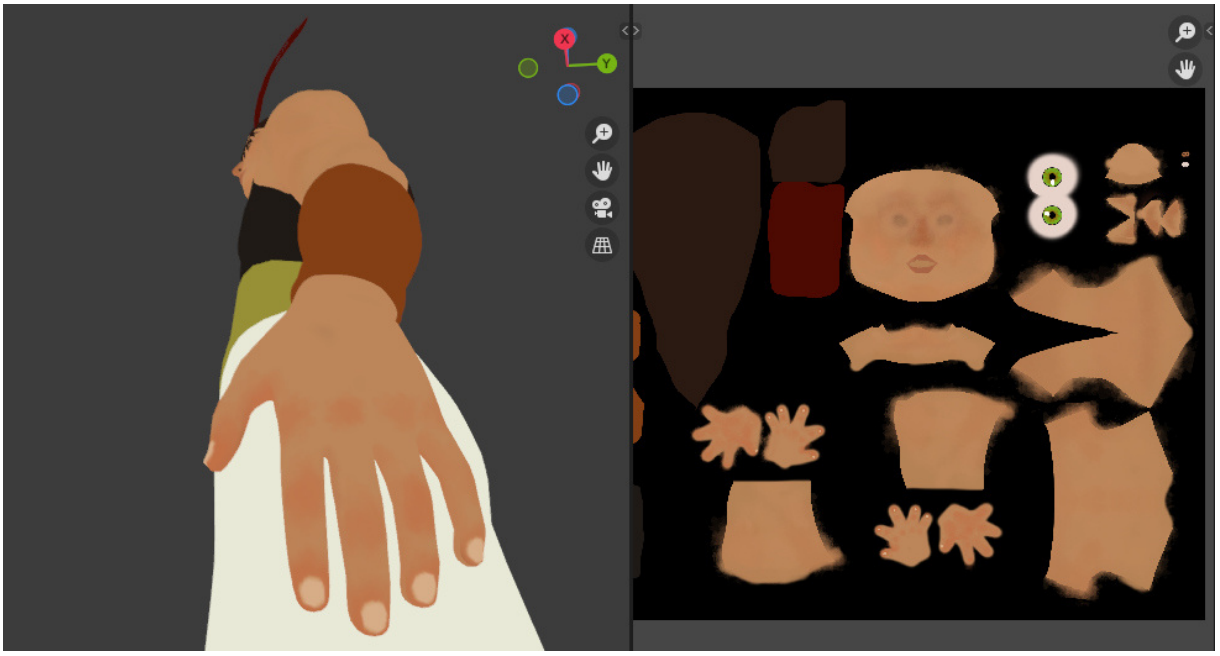
5.1.6.4 Texturização

O processo de texturização do modelo se deu em sua maioria no próprio Blender, em duas etapas:

A primeira foi a de *bake* das normais, que é um processo onde é possível extrair os detalhes da malha inicial e projetá-los sobre a malha de retopologia, imprimindo-os em sua UV na forma de imagem. Graças a isso, é possível alcançar um nível de detalhe muito alto com um custo de processamento bem menor.

A seguir foi feita a cor base, conhecida ainda como albedo ou *diffuse*. Primeiramente ela foi blocada no Blender e a seguir foi levada a um programa de edição de imagens, onde foram adicionados mais detalhes. Porém, ocorreu de que quando trazidas de volta ao Blender, em algumas partes da textura, as *seams* ficaram em evidência. Para solucionar isso, foi utilizado o modo de pintura de textura do Blender, com o viewport display em sólido, a cor em textura e a luz chapada (figura 26). Assim, pode-se trabalhar com clareza total na textura do personagem e assim solucionar todos os problemas de *seam*.

Figura 26 - Texturização do personagem



Fonte: Elaborada pelo autor.

Para os fins deste projeto, não se fez necessária a criação de mapas adicionais, pois o conceito visual é cartunesco. Por esse motivo que na única instância na qual poderia ser necessária a utilização de um mapa metálico, este efeito foi reproduzido através de uma técnica de pintura que tenta imitar os brilhos e sombras que estariam presentes em um objeto metálico.

5.1.6.5 Rigging

Tendo o modelo pronto, pode se passar para a seguinte etapa, que é o rigging. Neste processo, é criada uma armadura que virá a deformar a malha do personagem. Para facilitar o processo, foi utilizado o *Human Meta Rig*, que é uma das predefinições que vêm junto da extensão Rigify. É importante lembrar que os *rigs* do Rigify são modulares, o que quer dizer que se for necessário adicionar ou remover alguma parte do rig, o processo é bem direto. No caso deste projeto, foram adicionadas à armadura base três instâncias de tentáculo, uma para o cabelo e duas para as penas na cabeça, e ambos foram parenteados ao *bone* da cabeça.

Fora isso, alguns cuidados devem ser tomados ao ajustar o rig ao corpo do personagem. Por exemplo, o ângulo no qual as juntas irão ser deformadas deve ficar implícito no seu posicionamento. Quer dizer que o joelho deve estar angulado para a frente, e o cotovelo para trás. O mesmo se aplica aos dedos, porém eles são mais delicados, pois qualquer rotação a mais irá resultar nos dedos dobrando de forma esquisita. Para solucionar este problema de forma efetiva, é necessário deixar visíveis os eixos dos *bones*, em seguida, no menu de edição, a orientação da transformação para afetar as normais e o ponto de pivô para *bounding box center*. Tendo feito isso, pode-se selecionar cada dedo e ao escalonar a largura (eixo x) para zero, o dedo se endireitará.

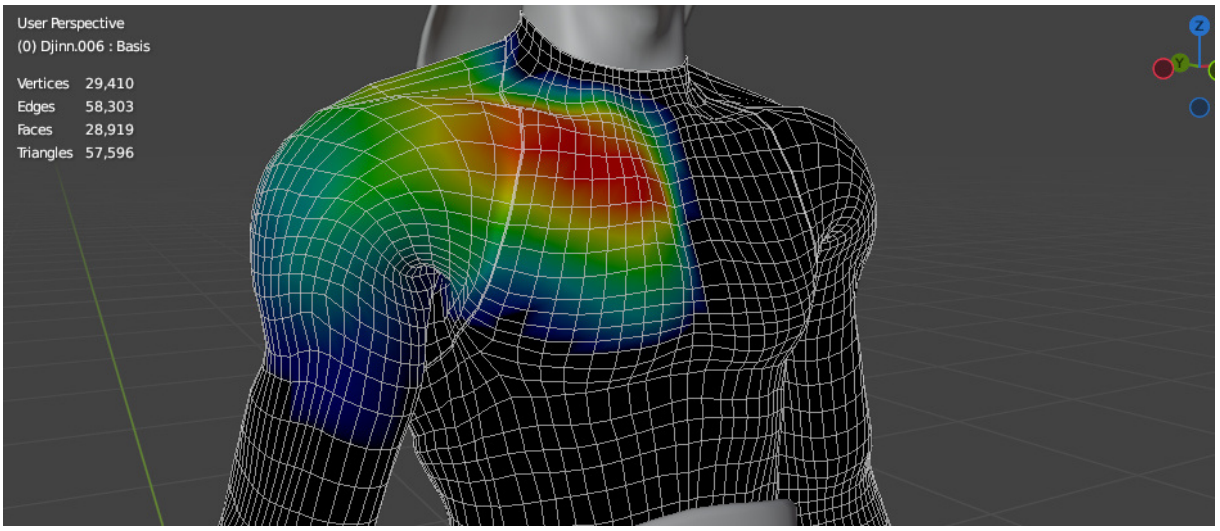
Após esses preparativos, a partir do meta rig pode ser gerado o rig, que trará uma série de funcionalidades extras, como a implementação de um sistema que pode intercalar entre o FK (*forward kinematics*) e IK (*inverse kinematics*) individualmente em cada membro. Além disso, o rig também proporciona uma série de *tweak bones*, que são *bones* especiais que permitem ajustar a deformação da malha. O personagem então deve ser parenteado a esse rig com pesos automáticos.

5.1.6.6 Weight paint

A partir dos pesos automáticos criados ao parentear o objeto à armadura, é muito importante passar pelo processo do *weight painting*, uma fase de ajuste dos pesos, pois são eles que regem a influência dos *bones* sobre a malha. Em outras palavras, a deformação que compõe o movimento na animação depende completamente do peso dos *bones* e, embora os pesos automáticos possam servir para um uso mais simples, é ideal que estes sejam refinados.

Malhas desconexas, como mencionado anteriormente, devem ter o seu peso igualado nas áreas onde houver contato. Isto é mais simples em áreas que sofram influência de apenas um ou dois *bones*. No projeto, um exemplo disso foi a conexão entre o tronco e a cabeça, onde ao igualar o valor da influência do *bone* da espinha foi possível que a transição entre uma parte e outra quando posicionado fosse imperceptível. Em contrapartida, o também mencionado caso das malhas do tronco e o top, onde não foi possível encontrar uma solução viável apenas com o *weight painting*. Uma solução que foi explorada para tentar resolver este problema foi a utilização de *shape keys*, tópico no qual entraremos em mais detalhes em seguida. O caso é que essa solução também não obteve os resultados desejados. A solução que foi implementada com sucesso, como já mencionada, foi a de completamente retrabalhar completamente a malha na área do tronco do personagem, de modo a que esta fosse uma malha única. A partir dela, foi possível achar uma solução para a deformação do ombro apenas com o *weight painting* (figura 27).

Figura 27 - *Weight paint* do ombro do personagem



Fonte: Elaborada pelo autor.

A qualidade da malha irá influenciar diretamente na deformação alcançada pelo weight paint, não só pela já mencionada interação entre malhas distintas, mas por poder se aproveitar dos *edge loops* para alcançar as deformações desejadas com maior facilidade.

5.1.6.7 Shape Keys

As shape keys são uma forma muito útil de controlar como a malha é deformada. Ela funciona definindo uma base e criando cópias, que podem ser editadas separadamente. Cada cópia tem um valor referente à sua influência sobre a forma base e este valor pode ser atribuído a um *gadget*, se tornando parte do próprio rig, ou criando um *driver*, que automatiza o valor da variável com base na transformação de um *bone* selecionado. No caso do projeto, foi decidido utilizar *shape keys* para a deformação dos cotovelos e joelhos, sendo os cotovelos conectados por drive à rotação dos antebraços e os joelhos à rotação das canelas.

5.1.7 Aprendizagem

Tendo passado por todo esse processo, que foi marcado por uma série de percalços, ficou claro que, independente dos problemas que surgirem, o importante é persistir e seguir em frente. Pois caso nos deixemos abater, se tornará muito difícil de progredir. É importante termos a humildade de reconhecer que não conseguimos fazer alguma coisa e então pedir ajuda ou buscar outras saídas.

5.2 Animação

Nessa etapa, o modelo já está pronto para ser animado. Mas antes disso, é importante fazer um planejamento de quais animações serão trabalhadas.

No blender existe o *action editor*, dentro do menu *dope sheet*, onde podem ser criadas as ações individualmente. Ex: caminhar, correr e pular são ações separadas que podem ser combinadas em uma animação única ou exportadas para uma engine onde serão utilizadas conforme o jogador se movimente.

5.2.1 Blocagem

Toda animação tem início com a fase de blocagem, pois nela é possível experimentar com as poses principais (*keyframes*) e o espaçamento entre elas. Assim, ganhando uma noção da estrutura e do timing em um estágio muito inicial da animação, para que caso haja a necessidade de retrabalhar ou refazer alguma parte não haja muita perda de tempo. Este processo é análogo à animação *pose to pose*. Para facilitar a visualização neste processo, no Blender é possível mudar o modo de interpolação para constante, no menu editor de gráficos.

Para adicionar a isso, o Blender possibilita salvar qualquer pose utilizada durante as animações na biblioteca de poses, para depois serem reutilizadas. Estas podem ser do *rig* completo ou apenas de parte dele, como por exemplo a posição de um dos braços.

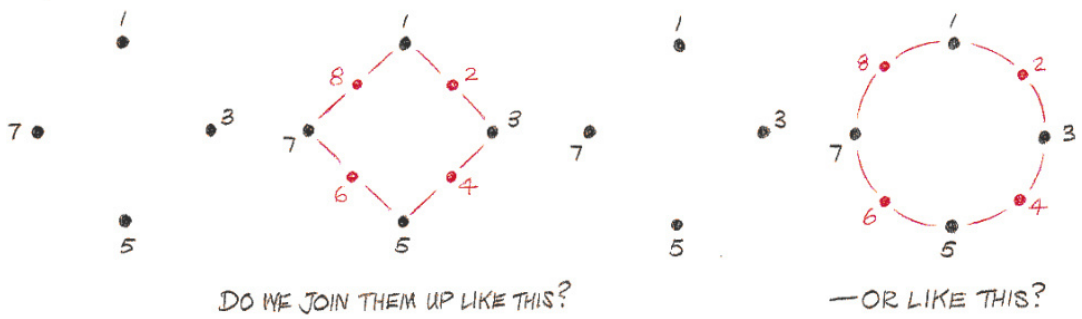
5.2.2 Entre frames

Com base nos *keyframes* e *extremes* elaborados durante a etapa de blocagem, é possível seguir para os *breakdowns*, que servem para mostrar como um objeto passa de um *extreme* para o próximo. Esta etapa é especialmente importante, pois nela serão trabalhados princípios de animação como curvas, timing, exagero e sequência e animação sobreposta.

Como já mencionado anteriormente, os *breakdowns* irão descrever a natureza do movimento, se este seguirá em reta ou em arco (figura 28). Isto depende da finalidade do movimento, mas é importante ter em mente que a grande maioria dos objetos na natureza se movimentam em arcos. Segundo Frank Thomas e Ollie Johnston, antes da descoberta do princípio dos arcos:

> "Em uma caminhada, os personagens subiram e desceram como partes mecânicas em um motor; agora eles "arqueiam" no topo de seus passos e "arqueiam" na posição inferior. Um golpe ou um arremesso poderia estar em uma linha completamente reta, mas o início da ação foi trazida em um arco e o seu seguimento iniciou um movimento em espiral." (THOMAS e JOHNSTON, 1981, p. 62)

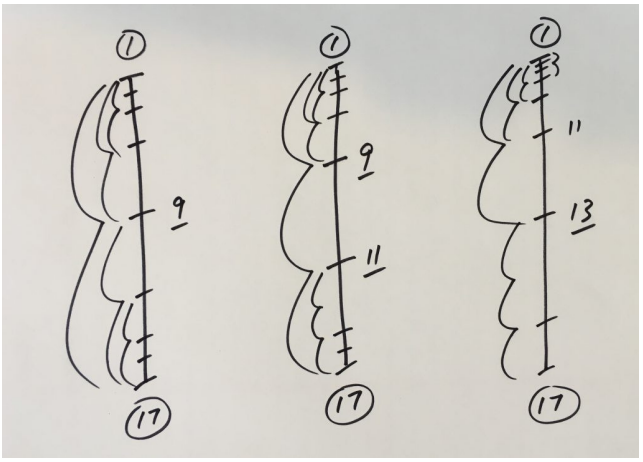
Figura 28 - Breakdowns



Fonte: The Animator 's Survival Kit (2001, p. 91).

Além do posicionamento, o *spacing* também é um fator a ser considerado já nesta etapa. Ele, diferente do timing, não consiste em quantos frames compõem a ação, mas sim na forma como os frames estão dispostos nela (figura 29).

Figura 29 - *Spacing*



Fonte: Animation mentor, 2022.

Ao se utilizar do *spacing* de modo que o posicionamento do objeto em frames subsequentes seja muito distante, embora isso traga um dinamismo para a ação, pode causar também uma sensação de que a animação está entrecortada (STOEBER, 2021). Uma das técnicas utilizadas para combater isso, é a deformação do objeto, a qual pode se utilizar do princípio do *squash and stretch*, que mantém a massa do objeto, ou do *smear*, que não tem essa limitação. *Smears*, ou *elongated in-betweens* podem ser representados de diferentes formas, por exemplo: eles podem se utilizar de linhas de ação; podem esticar o objeto, seja todo ou em parte; e também, pode duplicar os membros, feições . É claro que todos estes métodos podem ser combinados, o limite é a imaginação.

Para tentar reproduzir esse efeito em 3D, procurei exemplos em filmes de animação como o Homem-Aranha no Aranhaverso (2018), onde embora seja um filme de animação em 3D, ele traz consigo uma série de decisões que visam borrar a linha entre animação 2D e 3D. Além de ter um visual que imitava o aspecto da impressão dos quadrinhos, também trazia a utilização do *framerate* como elemento narrativo e a utilização de diferentes formas de *smear frames* (figura 30).

Figura 30 - Homem-Aranha *smear frame*

Fonte: Homem-Aranha no Aranhaverso, 2018.

Para reproduzir esse tipo de efeito no Blender, foram duplicadas duas vezes cada membro e então parenteados à uma série de *bones*, dois a dois. Por sua vez, cada conjunto de *bones* foi parenteado ao *bone* do antebraço ou da canela, respectivamente, onde permanecerão escondidos dentro da malha em escala reduzida (1% da escala original) até serem necessários para a animação. É importante ressaltar que a utilização de *smears* deve ser contida a um número pequeno de *frames*, pois se ficar muito visível para o espectador pode gerar estranheza.

No vídeo a seguir pode ser visto como a escala foi aplicada aos membros extras: https://drive.google.com/file/d/1c9GsWSzvCe-o_PaJIZnYEGxecqyr3B-s/view?usp=sharing.

Uma outra alternativa seria substituir a malha por outra diferente em determinados frames da animação. O jogo Crash Bandicoot para o Playstation foi um dos primeiros a se utilizar disso num ambiente 3D (figura 31) e este efeito foi replicado em seu remake Crash Bandicoot N. Sane Trilogy (figura 32).

Figura 31 - Crash Bandicoot *smear frame* original

Fonte: Crash Bandicoot, 1996.

Figura 32 - Crash Bandicoot *smear frame remake*

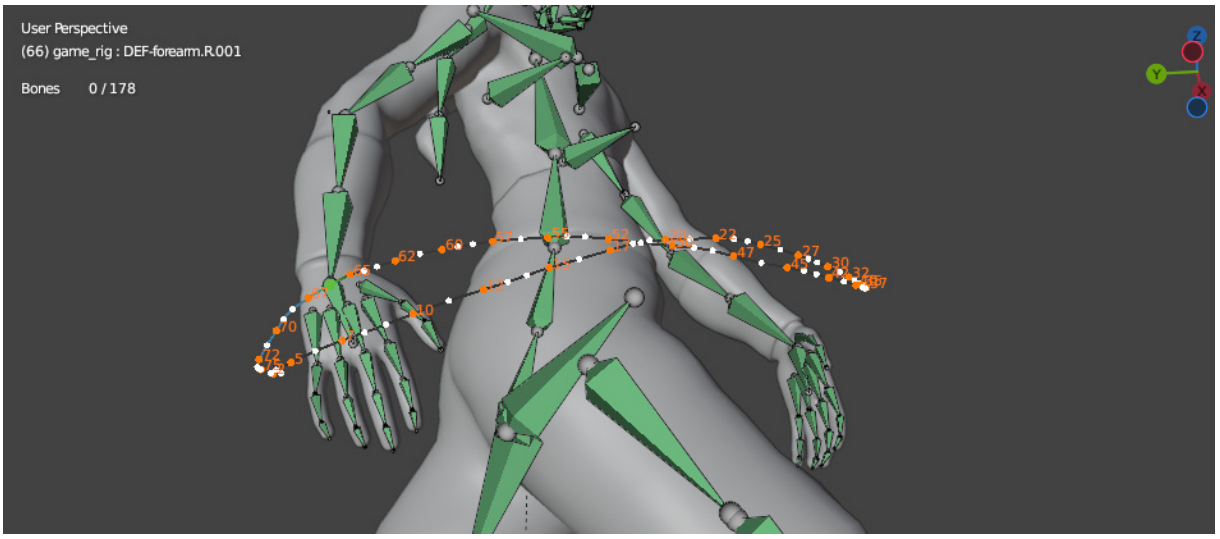
Fonte: Crash Bandicoot N. Sane Trilogy, 2017.

5.2.3 Refinamento

Uma ferramenta muito útil para esta etapa é a de visualização das curvas, pois ela facilita muito a análise da animação, tanto no que se refere ao princípio de

arcos quanto o espaçamento entre os frames, deixando em maior evidência em quais frames há necessidade de utilizar alguma forma de deformação do objeto. No Blender, ela pode ser ativada do modo de pose, apenas selecionando o *bone* desejado indo no menu *dropdown* superior, pose, e dentro da aba *motion paths* selecionar *generate* (Figura 33). Pode ser desabilitado a qualquer momento através do mesmo caminho, mas selecionando a opção *clear*.

Figura 33 - Visualização da curva de movimento do braço



Fonte: Elaborada pelo autor.

Quando já estiver satisfeito com o resultado conjunto dos *keyframes*, os *extremes e os breakdowns*, então é chegada a hora da etapa de Interpolação das animações. Para retornar ao estado que anterior ao da blocagem é necessário fazer o mesmo procedimento, porém mudando o modo de interpolação para bézier, no menu editor de gráficos. Assim, o Blender terá gerado automaticamente todos os *in-betweens*. A partir daí podem ser editadas as curvas de modo a estabelecer com clareza onde será utilizado o *slow in* e o *slow out*.

5.2.4 Preparando para exportar

O Blender é um programa de modelagem e de animação e o modo que ele opera é otimizado para esse propósito, entretanto este é diferente do modo que as *game engines* atuam. Alguns dos principais problemas encontrados ao tentar exportar um *rig* sem o devido preparo são: a má hierarquia dos bones; bones desnecessários; e deformações na malha durante uma transformação de escala,

decorrentes do parenteamento dos *bones* pelo *rig*, o que aplica as transformações como herança. Este tipo de transformação é a essência do efeito de squash e stretch. Ocorre que o *rig* gerado pelo *rigify* é bem complexo, tendo não apenas FK e IK, mas também os *tweak bones*.

Tendo em vista solucionar esse problema, Todor, um artista 3D *freelancer* e dono do canal CGDive no Youtube lançou uma série de vídeos educativos chamada *Bridging the gap*, onde busca aproximar o processo de animação do Blender com o de produção de jogos, independente se é Unity, Unreal ou Godot. O conceito base foi desenvolvido por Pierrick Picaut, da empresa P2design, em seu curso "*the art of effective rigging*". A sua solução, chamada TGT-Solution consiste em fazer uma cópia da armadura de deformação, mudando o sufixo de DEF para TGT (target) e tornar seus *bones* em *non-deforming*. Em seguida, deve-se restringir cada DEF *bone* para o seu correspondente TGT *bone*, para então trabalhar exclusivamente nos TGT bones.

A partir daí, existem duas vertentes: pode-se conectar a hierarquia dos DEF bones, no modo que quiser; ou pode simplesmente desconectar todos os DEF bones, assim criando uma hierarquia plana. Somente assim é possível alcançar o resultado de *squash* and *streatch*, pois é necessário apagar todas as ligações de parentesco e depois fazer um *bake* das animações.

O problema até então é que o processo era mecânico, repetitivo e completamente manual, então havia chances de erro humano. Por isso, Blenderboi desenvolveu um *add-on* para o Blender chamado de *Game Rig Tools* que foi baseado no *workflow* mostrado pelo canal CGDive. Este add-on automatiza quase todo o processo, sendo a única parte manual do processo é o de consertar a hierarquia do rig, pois até o bake das animações está a um clique de distância.

Para finalizar, apenas a malha e o game rig são exportados em fbx, sem a criação de *leaf bones* e fazendo o *bake* das animações no modelo.

5.2.5 Importando na Unity

Uma limitação por parte da Unity, que foi algo que impactou negativamente no processo de implementação das animações, é que, ao importar animações, o modo de interpolação entre os frames escolhido no Blender é ignorado, automaticamente gerando interpolações entre os keyframes. Isso normalmente poderia ser ignorado, porém no caso do uso dos *smear frames* de duplicatas, onde é utilizada a escala

para que as partes apareçam e desapareçam em questão de *frames*, essa interpolação estraga o efeito desejado.

Por isso, é interessante a utilização de um *frame rate* que multiplicado por um número inteiro resulte em 60. Por exemplo, no lugar de animar em 24 frames por segundo (ones), animar em 12 (twos) e multiplicar o resultado por 5 para conseguir 60 frames. Ou então, animar em 30 frames por segundo e apenas multiplicar por 2 para alcançar os 60. Porém, como as animações já haviam sido produzidas utilizando os 24 quadros por segundo, foi necessário convertê-los para 60. Nesse caso, a animação teve que ser escalonada por 2.5, o que pode ocasionar distorções no timing da animação, assim prejudicando no resultado final. Por isso foi necessário fazer alguns ajustes para alcançar um resultado satisfatório.

Uma vez importado para a Unity, ao selecionar o personagem e ir na aba rig, selecione o tipo de animação para genérico, a definição do avatar para “criar a partir deste modelo” e o Root como DEF spine, e então aplicar.

A seguir, é necessário criar um *animator controller*, que será adicionado como componente para o objeto personagem. Ao editar o animator, é importante criar algumas variáveis que servirão como gatilho para as diversas animações. É preciso começar com um estado padrão, geralmente o idle, do qual as outras animações vão partir e para a qual irão retornar. Outra possibilidade é utilizar as *blend trees*, que fazem uma transição suave entre as distintas animações. Por exemplo, uma *blend tree* onde quando o valor da velocidade for 0 o personagem esteja em idle, quando for 0.25 esteja caminhando e quando quando for 1 esteja correndo.

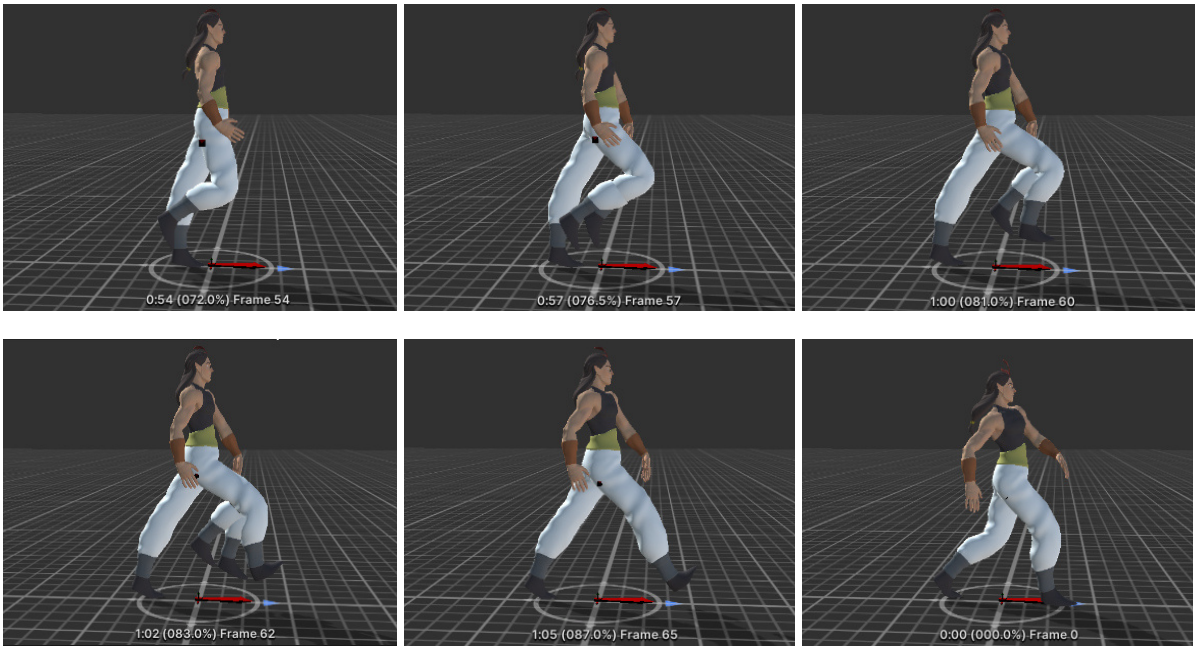
6. Resultados

O resultado alcançado durante a produção deste projeto foi a criação de um personagem, de seu conceito, à modelagem 3D, texturização e rigging; a elaboração de três animações, criadas com base nos princípios de animação e a combinação de técnicas oriundas da animação tradicional, como no caso dos smear frames; e sua exportação e integração à engine Unity.

As sequências de imagens a seguir demonstram o funcionamento das animações dentro da Unity. Onde a primeira é referente à caminhada, a segunda à corrida e a terceira ao pulo.

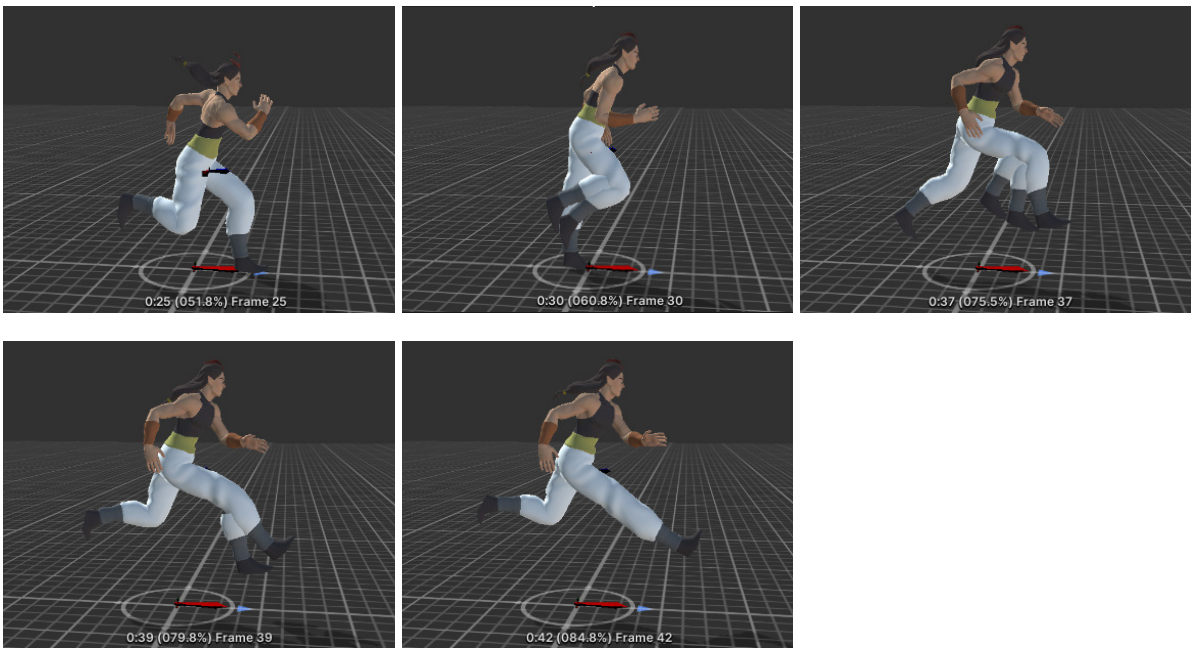
O vídeo com as animação pode ser acessado com o seguinte link: https://drive.google.com/file/d/1V73r0UoVpt0j_ZBtLtrzuHKbgFrfZMNm/view?usp=sharing.

Figura 34 - Primeira sequência de imagens, caminhada



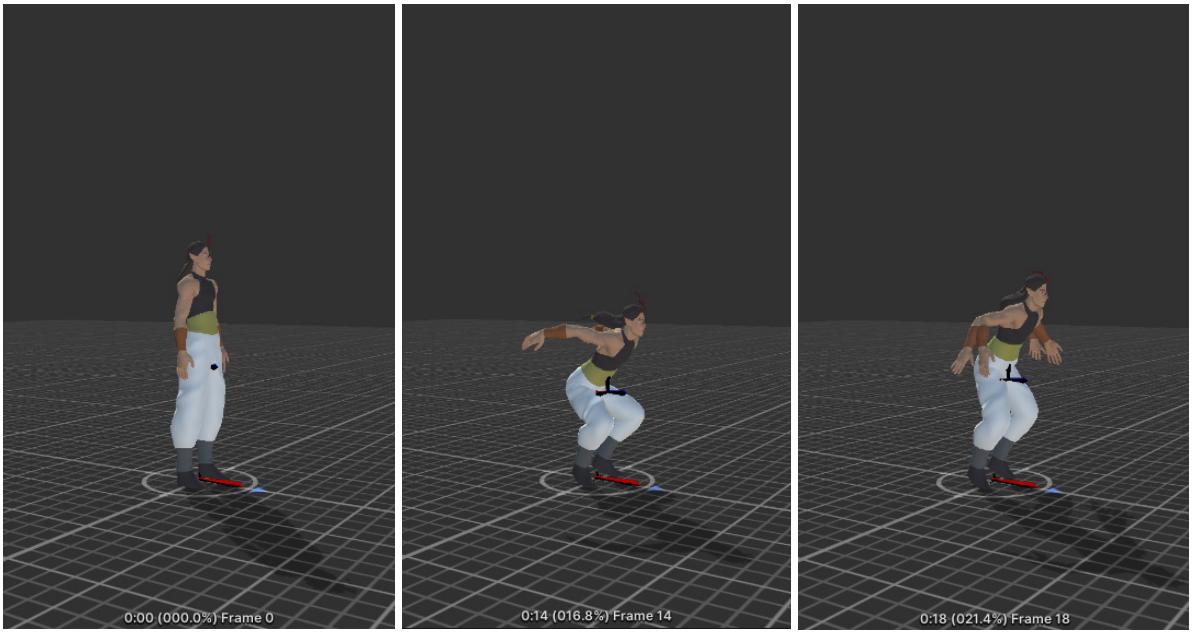
Fonte: Elaborada pelo autor.

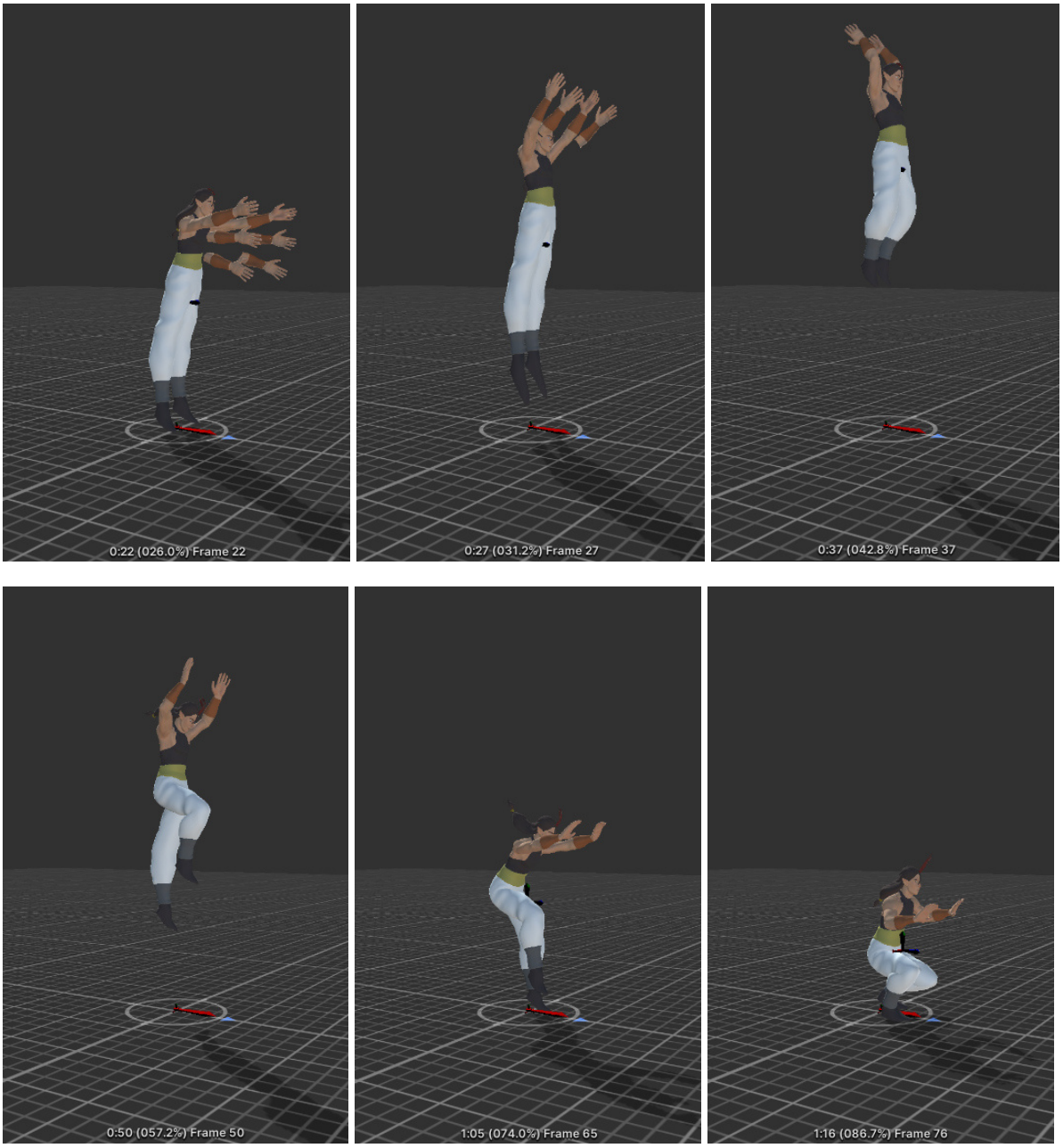
Figura 35 - Segunda sequência de imagens, corrida



Fonte: Elaborada pelo autor.

Figura 36 - Terceira sequência de imagens, pulo





Fonte: Elaborada pelo autor.

Também é interessante termos um comparativo da animação com e sem a utilização dos *smear frames*, para termos uma noção de como o efeito afeta a sensação da ação. Pelo link a seguir pode ser visto o comparativo lado a lado no pulo: https://drive.google.com/file/d/15ctVnzaXxdrorD36JJPMSq09_OqZpUB_/view?usp=sharing.

7. Conclusão

O meio de animação é muito rico e a sua adaptação de forma expressiva e cartunesca à mídia de jogos 3D é uma vertente muito interessante.Por isso, a busca por entender quais passos devem ser tomados para alcançar sua adaptação, de forma expressiva e cartunesca, à mídia de jogos 3D é importante. À medida que novos *workflows* vão surgindo vão se abrindo as portas para que novos artistas possam se aventurar nesse meio, forçando os limites daquilo que é possível.

A utilização do *workflow* de CGDive abre várias possibilidades para a utilização de deformações na malha para jogos, em um ambiente onde este tipo de transformação não é traduzida corretamente de forma nativa. No caso deste projeto, foram desenvolvidas três animações que se utilizam de características cartunescas, como a utilização de *smear frames*.

Os princípios de animação foram sendo aplicados ao decorrer das etapas. Como por exemplo, o atrativo do personagem foi trabalhado ao criar o seu design, tentando criar algo interessante para o espectador se interessar. O desenho sólido fica implícito, pois o personagem é de fato 3D. Antecipação, ação secundária e exagero também foram trabalhados ao fazer a blocagem da animação. Apenas à encenação não foi dado o devido cuidado, pois ainda na etapa do planejamento das animações poderiam ter sido escolhidas um conjunto de animações que dessem ênfase à dinâmica do movimento.

O desenvolvimento do projeto passou por alguns problemas, inicialmente relacionados à malha e a deformação que os *bones* exerciam sobre ela. Porém, isso ocasionou uma perda muito grande de tempo e essa perda exigiu um replanejamento do restante do trabalho. Por consequente, embora a intenção inicial era a de produzir uma série de conjuntos de animações, o escopo teve de ser drasticamente reduzido e se restringir a apenas um.

A escolha das animações foi feita com a intenção de aplicá-las a um personagem jogável, por isso é uma movimentação básica. Porém, em retrospecto deveriam ter sido escolhidas animações mais dinâmicas para dar maior ênfase aos efeitos aplicados. Além disso, inicialmente havia sido planejado muito maior foco na distorção da malha, se utilizando dos *tweak bones*, porém no final foi decidido pela utilização principal da duplicação de membros.

A criação do modelo do personagem foi bem desenvolvido, tendo tido um grande cuidado com a deformação da malha, incluindo o processo de modelagem e de pintura dos pesos. Porém, devido à precaução de manter a forma da malha durante as deformações associada à escolha da técnica de duplicação de membros causou um grande aumento na quantidade de polígonos no personagem. Isso pode ser viável para um único personagem, porém no contexto de um jogo com vários personagens esse método se torna inviável. Assim, se torna necessário fazer uma otimização da malha do personagem.

Em relação à animação, as poses poderiam ter sido ainda mais exageradas, passando uma sensação mais dinâmica e a escolha das animações poderia ter sido mais adequada à proposta do trabalho. Principalmente porque os *smears* ocorrem em ações muito rápidas, para ressaltar a dinâmica do movimento, e isso não é algo que geralmente ocorra em um ciclo de caminhada. Entretanto, no exemplo do pulo a utilização tanto dos princípios quanto o uso dos *smears* trouxeram um resultado interessante.

A utilização da *pipeline* de desenvolvimento de animações para jogos Blender-Unity se mostrou bastante funcional. A utilização do Blender para a criação tanto do personagem quanto das animações se mostrou não apenas possível, como bem prático pelo fato de não haver a necessidade de estar migrando de um programa para outro para produzir as diferentes partes do processo. É claro que outras ferramentas, como o ZBrush, são mais especializadas, mas diferente do Blender elas são pagas. Isso torna o Blender uma ferramenta ideal para quem está entrando no mercado. A transição das animações para a Unity foi muito simples graças ao *add-on Game Rig Tools*, que simplificou imensamente o processo. Uma vez dentro da Unity as animações podem ser trabalhadas como qualquer outra, com a diferença que depois de fazer o *bake* estas se tornam exclusivas daquele modelo, não podendo ser reaproveitadas em outro *rig*.

Apesar dos problemas encontrados, foi possível apresentar todo o processo de desenvolvimento de animações 3D cartunescas para jogos. Foram mostradas todas as etapas da produção da animação para jogos, desde a criação do conceito do personagem ao *rigging* e animação, se utilizando dos princípios clássicos de animação, além da implementação na engine de jogos.

Visando os desdobramentos para trabalhos futuros, já que a produção deste trabalho teve um viés mais técnico, poderia ser interessante analisar o contexto das

animações em jogos de um ponto de vista mais estético e artístico. Outra possibilidade é a de focar em apenas uma das etapas de produção, como por exemplo apenas a produção das animações cartunescas, tendo desde o início do projeto um personagem pronto para animar. Algo que poderia ter sido mais explorado é como trabalhar com as animações dentro da Unity e até mesmo como animar direto dentro da engine.

8. Referências

AMBROSE G; HARRIS P. Basics Design 08: **Design Thinking**. Switzerland: AVA Publishing SA, 2010.

PANTOJA, Toniko. **Animation smears - the hand drawn motion blur**. YouTube, 2018. Disponível em: https://www.youtube.com/watch?v=KXuP2_BCmV0. Acesso em: 4 dez. 2021.

BLENDER DOCUMENTATION TEAM. **Blender 2.92 Reference Manual**, c2021. Manual do Blender. Disponível em: https://docs.blender.org/manual/en/2.92/. Acesso em: 4 dez. 2021.

CGDive. **Blender to Game Engines (Unreal, Unity, Godot etc.)**. YouTube, 2021. Disponível em: https://www.youtube.com/playlist?list=PLdcL5aF8ZcJvCyqWeCBYVGKbQgrQngen3. Acesso em: 2 jun. 2022.

CGDive. **Rigify Tutorial Series: "Rig Anything with Rigify"**. YouTube, 2020. Disponível em: https://www.youtube.com/playlist?list=PLdcL5aF8ZcJv68SSdwxip33M7snakl6Dx. Acesso em: 14 mai. 2022.

DACANAY, Sean. **How Crash Bandicoot hacked the original Playstation | War Stories | Ars Technica**. YouTube, 2020. Disponível em: https://www.youtube.com/watch?v=izxXGuVL21o. Acesso em: 4 dez. 2021.

STOEBER, Jenna. **How devs break bones to make animation feel right**. YouTube, 2021. Disponível em: https://www.youtube.com/watch?v=vIdeGmN__Pw. Acesso em: 4 dez. 2021.

PALMER S; FELSING J. **A practical guide to Feature-Driven Development**. 1ª ed. Upper Saddle River, NJ: Prentice Hall, 2002.

POSVOLSKI A. *et al*. **AgiGame: Proposta de uma metodologia híbrida para desenvolvimento de jogos**. SBGames, Porto Alegre, RS, XIII, p. 1075-1084, Nov. 12-14, 2014.

THOMAS F; JOHNSTON O. **The Illusion of Life**: Disney Animation. 1ª ed. New York: Disney Press, 1981.

UNITY TECHNOLOGIES. **Unity User Manual 2020.3**, c2021. Manual da Unity. Disponível em: https://docs.unity3d.com/Manual/index.html. Acesso em: 4 dez. 2021.

VAUGHAN W. **[digital] Modeling**. Berkeley, CA: New Riders, 2012.

VAUGHAN W. **The Pushing Points Topology Workbook**: Volume 01. Clermont, FL: Hickory Nut Publishing, 2018.

WILLIAMS R. **The Animator's Survival Kit**: a manual of methods, principles and formulas for classical, computer, games, stop motion and internet animators. United States: Faber and Faber, 2001.