

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO
TOCANTINS
CAMPUS ARAGUAÍNA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE
SISTEMAS**

CAIO FELIPE DE SOUSA COSTA

**DESENVOLVIMENTO DE JOGOS POR INICIANTE:
A UTILIZAÇÃO DO MIXAMO PARA RIGGING E MECÂNICA PRIMÁRIA
EM JOGOS 3D EM 3ª PESSOA**

**ARAGUAÍNA-TO
2022**

CAIO FELIPE DE SOUSA COSTA

**DESENVOLVIMENTO DE JOGOS POR INICIANTEs:
A UTILIZAÇÃO DO MIXAMO PARA RIGGING E MECÂNICA PRIMÁRIA
EM JOGOS 3D EM 3ª PESSOA**

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Tocantins, como requisito final para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Paulo Ricardo da Silva Pontes

ARAGUAÍNA-TO

2022

Dados Internacionais de Catalogação na Publicação (CIP)
Bibliotecas do Instituto Federal do Tocantins

C838d Costa, Caio Felipe de Sousa
DESENVOLVIMENTO DE JOGOS POR INICIANTEs: A
UTILIZAÇÃO DO MIXAMO PARA RIGGING E MECÂNICA PRIMÁRIA
EM JOGOS 3D EM 3ª PESSOA / Caio Felipe de Sousa Costa. –
Araguaína, TO, 2022.
67 f. : il. color.

Trabalho de Conclusão de Curso (Tecnólogo em Análise e
Desenvolvimento de Sistemas) – Instituto Federal de Educação,
Ciência e Tecnologia do Tocantins, Campus Araguaína, Araguaína,
TO, 2022.

Orientador: Me. Paulo Ricardo Silva Pontes

1. Mixamo. 2. Rigging. 3. Mecânica de Jogos. I. Silva Pontes,
Paulo Ricardo. II. Título.

CDD 004.2

A reprodução total ou parcial, de qualquer forma ou por qualquer meio, deste documento é autorizada para fins
de estudo e pesquisa, desde que citada a fonte.

Elaborado pelo sistema de geração automática de ficha catalográfica do IFTO com os dados fornecidos
pelo(a) autor(a).

CAIO FELIPE DE SOUSA COSTA

**DESENVOLVIMENTO DE JOGOS POR INICIANTE:
A UTILIZAÇÃO DO MIXAMO PARA RIGGING E MECÂNICA PRIMÁRIA EM
JOGOS 3D EM 3ª PESSOA**

Monografia de Conclusão de Curso Superior apresentado à Coordenação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Tocantins, como requisito final para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Aprovado em: ____/____/____

BANCA AVALIADORA



Documento assinado digitalmente

PAULO RICARDO DA SILVA PONTES

Data: 16/12/2022 16:08:31-0300

Verifique em <https://verificador.iti.br>

Me. Paulo Ricardo da Silva Pontes
IFTO — *Campus Araguaína*



Documento assinado digitalmente

GILVAN VIEIRA MOURA

Data: 19/12/2022 16:49:15-0300

Verifique em <https://verificador.iti.br>

Me. Gilvan Vieira Moura
IFTO — *Campus Araguaína*



Documento assinado digitalmente

ALEXANDRE COTRIM VILAS BOAS

Data: 19/12/2022 16:21:53-0300

Verifique em <https://verificador.iti.br>

Me. Alexandre Cotrim Vilas Boas
IFTO — *Campus Araguaína*

*Dedico este trabalho ao primeiro
indivíduo abastadamente interessado em
lê-lo com afinco e diligência genuína.
Até lá... Parousia!*

AGRADECIMENTOS

Quero expressar toda a minha felicidade, gratidão e alegria, ao Criador: a Matéria Amorfa. A Substância viva, inteligente e consciente de que é e está em tudo e vive em tudo. Aos meus Pais que sempre investiram na minha educação e muito me incentivaram a seguir e continuar nessa jornada de aprendizado. Sei que é clichê dizer isso, mas vocês são os pilares da minha vida.

Aos meus colegas de aula que contribuíram direta ou indiretamente ao longo de todos os semestres do curso, do começo ao fim dessa inesquecível e coletiva trajetória. Aos meus professores por toda paciência e coerência que tiveram em ensinar-me e em particular ao grande mestre e o meu orientador Paulo Ricardo Pontes, que com sabedoria me conduziu de forma excepcional a organizar, criar e dar luz a esse material tão importante para mim.

Parousia!

*“Qualquer tecnologia suficientemente
avançada é indistinguível de magia.”
(Arthur Charles Clarke)*

RESUMO

A cada ano surgem no mercado ferramentas que auxiliam e minimizam o tempo de criação de elementos gráficos para desenvolvedores de jogos eletrônicos, amenizando a complexidade de se elaborar e produzir um jogo atualmente. Essas ferramentas não só disponibilizam benefícios que outrora levaria bastante tempo para gamers designers desenvolverem, como acaba barateando significativamente os custos de produção de um jogo. O intuito desse trabalho é apurar as possibilidades de criação de rigging e mecânica primária em personagens jogáveis em jogos 3D em terceira pessoa através da utilização de uma dessas ferramentas: a plataforma Adobe Mixamo. Esse trabalho é voltado para gamers designers iniciantes que almejam conceber projetos neste estilo de jogo mas padecem de falta de experiência com modelagem tridimensional, retopologia e captura de movimento e dentro dos padrões de exigência atuais, onde o avatar interage e se articula de forma realista durante a gameplay, além de fornecer base teórica sobre mecânica, estilo gráfico e gêneros de jogos eletrônicos. Ao final serão expostas as considerações sobre este trabalho e sua temática denotando ênfase nos principais mecanismos e resultados atingidos.

Palavras-chave: Mixamo, Rigging, Jogos 3D, Mecânica de Jogos

ABSTRACT

Every year, tools appear on the market that help and minimize the time to create graphic elements for electronic game developers, easing the complexity of designing and producing a game today. These tools not only provide benefits that would otherwise take a long time for game designers to develop, but also significantly lower the cost of producing a game. The purpose of this work is to investigate the possibilities of creating rigging and primary mechanics in playable characters in third-person 3D games through the use of one of these tools: the Adobe Mixamo platform. This work is aimed at beginner game designers who aim to conceive projects in this style of game but suffer from a lack of experience with three-dimensional modeling, retopology and motion capture and within the current standards of demand, where the avatar interacts and articulates in a realistic way during gameplay, in addition to providing a theoretical basis on mechanics, graphic style and electronic game genres. At the end, the considerations about this work and its theme will be exposed, denoting emphasis on the main mechanisms and results achieved.

Keywords: Mixamo, Rigging, 3D Games, Game Mechanics

LISTA DE ILUSTRAÇÕES

Figura 1 — Cena do jogo Doom.....	19
Figura 2 — Cena do jogo Assassin 's Creed Origins.....	22
Figura 3 — Cena jogo Call of Duty: Modern Warfare.....	24
Figura 4 — Cena jogo Just Cause 4.....	25
Figura 5 — Cena do jogo GTA V.....	27
Figura 6 — Cena de God of War.....	31
Figura 7 — Exemplo de malha tridimensional.....	35
Figura 8 — Estrutura de um esqueleto virtual humanóide.....	36
Figura 9 — Cena de Toy Story.....	38
Figura 10 — Captação de movimentos em God of War.....	41
Figura 11 — Criação de rigging no Mixamo.....	44
Figura 12 — Auto-Rigger devidamente posicionado.....	45
Figura 13 — Avatar com as respectivas animações.....	52
Figura 13 — Janela Animator.....	53
Figura 15 — Fluxo de animações no Animator.....	54
Figura 16 — Animações estabelecidas no Unity 3D.....	57
Figura 17 — Avatar com suas mecânicas.....	58

LISTA DE QUADROS

Quadro 1 — Tipos de mecânica.....	30
Quadro 2 — As três etapas de criação MDA.....	32
Quadro 3 — Principais conceitos de rigging.....	37
Quadro 4 — Parâmetros do Animator Controller.....	55

LISTA DE ABREVIATURAS SIGLAS

Triple-AAA: Classificação não oficial de jogos produzidos com grande orçamento.

DAE: No inglês “*Digital Asset Exchange*” extensão de arquivo que armazena modelos e texturas 3D.

FBX: Arquivo criado pela empresa Autodesk para armazenamento de dados em 3D e 3D.

FPS: No inglês “*Frames per Second*” é Quantidade de imagens exibidas em um segundo

MDA: No inglês “*Mechanics, Dynamics and Aesthetics*”. Abordagem formal para entender os jogos eletrônicos desenvolvido por LeBlanc.

OBJ: No inglês “*Object File Wavefront 3D*”. Formato de arquivo usado para um objeto 3D e contém suas coordenadas.

SUMÁRIO

LISTA DE ILUSTRAÇÕES.....	9
LISTA DE TABELAS.....	10
LISTA DE ABREVIATURAS SIGLAS.....	11
1 INTRODUÇÃO.....	13
1.1 Definição do Problema.....	14
1.2 Justificativa.....	14
1.3 Objetivos.....	15
1.3.1 Objetivo Geral.....	15
1.3.2 Objetivos Específicos.....	15
1.4 Estrutura da Monografia.....	16
2 EVOLUÇÃO E ESTILO GRÁFICO DE JOGOS ELETRÔNICOS.....	17
2.1 Evolução gráfica em jogos eletrônicos.....	17
2.2 Jogos em estilo gráfico 2D e 3D.....	21
2.3 Classificação, gênero e sub gêneros de jogos eletrônicos.....	23
3 MECÂNICA, JOGABILIDADE E RIGGING EM JOGOS ELETRÔNICOS.....	28
3.1 Mecânica e Jogabilidade.....	29
3.2 Classificação e níveis de mecânicas.....	33
3.3 Rigging de personagens 3D.....	35
3.3.1 Dificuldades em criar rigging e animações.....	39
3.3.3 Mecânica básica em jogos 3D em 3ª pessoa com o Mixamo.....	41
4 UTILIZAÇÃO DO MIXAMO COMO FACILITADOR DE PROCESSOS.....	43
4.1 Auto-Rigging e personagens do Adobe Mixamo.....	44
4.2 Adicionando animações no Adobe Mixamo.....	46
5 DESENVOLVIMENTO DE JOGOS COM UNITY 3D.....	48
5.2 Sobre o Unity 3D.....	49
5.2 Conceitos iniciais do Unity 3D.....	50
6 CONTROLE DE ANIMAÇÕES NO AMBIENTE UNITY 3D.....	51
6.3 Criando fluxos de transições para estabelecer a mecânica.....	54
7 CONSIDERAÇÕES FINAIS.....	59
REFERÊNCIAS.....	60
GLOSSÁRIO.....	64
APÊNDICE A.....	65
ANEXO A.....	66

1 INTRODUÇÃO

Criar jogos eletrônicos é uma ideia que fascina muitas pessoas. Esse fascínio talvez se deva pelo fato de um jogo envolver os mais diversos tipos de trabalhos artísticos como criações de modelos tridimensionais, ilustrações e música e ainda por ser um produto lúdico. Mas produzir um game não é tarefa simples.

Um jogo por mais básico que seja requer muito empenho por parte dos seus realizadores para se obter um produto final satisfatório no intuito de agradar um público ávido por novidades, inovação e qualidade gráfica. Segundo McShaffry (2012) a complexidade de se produzir um jogo eletrônico é uma tarefa árdua e pode deixar o desenvolvedor completamente aflito, pois os sistemas e ferramentas para o seu desenvolvimento se transformam muito mais do que se devia.

Estas dificuldades segundo Strougo e Wenderlich (2011) existem por ser necessário criar muitos componentes de software que necessitam de um amplo conhecimento de computação gráfica, redes e áudio. Por outro lado, os jogos como todos outros programas possuem recursos que facilitam seu desenvolvimento. Várias empresas de tecnologia nascem com o propósito de desenvolver ferramentas para viabilizar a produção de partes específicas de um jogo como cenários, personagens, materiais, shaders, sistema de câmeras, objetos, dentre outros (STROUGO; WENDERLICH, 2011).

O tema central deste trabalho é a demonstração de uma dessas ferramentas: o Adobe Mixamo. Essa plataforma dispõe de recursos que facilitam uma das tarefas mais difíceis na concepção de um jogo 3D: dar vida aos personagens. Este estudo demonstra seu uso prático na criação de mecânicas para jogos em terceira pessoa utilizando a Game Engine Unity 3D e é destinado à novos desenvolvedores, além de esclarecer as principais diferenças entre mecânicas, os principais estilos gráficos e gêneros de jogos encontrados atualmente.

Ao fim desse trabalho será posto todas as considerações fomentadas e todas as questões pertinentes sobre o tema exposto, destacando os principais pontos colhidos através de pesquisas, mostrando a praticidade e relevância do Adobe Mixamo como ferramenta significativa na produção de jogos eletrônicos.

1.1 Definição do Problema

A produção de um modelo tridimensional destinado a jogos em terceira pessoa é um trabalho árduo e vai além de simplesmente adicionar luz e sombra nos bitmaps de textura. Animá-los é outro processo massante. Atualmente usa-se técnicas de MoCap (*Captação de Movimentos*) e esqueleto virtual na produção de jogos desse gênero visando obter realismo nas animações dos personagens.

A maioria dos gamers designers iniciantes não possuem recursos para essa captação e tão pouco dispõe de conhecimentos sobre modelagem 3D e rigging, tendo em vista que estes conceitos não são o foco de sua profissão. Além da atuação do game designer, a produção de jogos digitais conta com artistas gráficos, level designer, programador e técnico em sonorização.

Com isso é nítido para um desenvolvedor que ao pôr em prática seu projeto ele estará fadado aos limites de sua atuação uma vez que para a realização dessas outras atividades é preciso conhecimento profundo e específico em áreas distintas.

Para tanto, o profissional disposto a produzir um jogo no estilo 3D em terceira pessoa, precisa dispor de ferramentas que facilitem a concepção do seu Avatar e NPCs bem como sistema de rigging para produção de animações em busca de estabelecer mecânicas minimamente relevantes.

1.2 Justificativa

A opção por abordar a plataforma Mixamo deve-se ao seu uso eficiente no desenvolvimento de jogos por iniciantes, propiciando uma biblioteca de personagens e animações realistas, e um sistema de criação de rigging extremamente didático. A importância dessa temática se justifica através da comprovação dessa pesquisa que a plataforma propicia rapidez no desenvolvimento de mecânicas tanto para NPCs quanto em personagens jogáveis.

Como mencionado, o fato de poucos gamers designers iniciantes disporem de conhecimentos de modelagem tridimensional e recursos de MoCap torna a experiência com o Mixamo não só bastante satisfatória como até mesmo decisiva na concepção de um jogo, além de ser uma plataforma totalmente gratuita.

A grande importância desse estudo fica clara também ao se checar o anseio de novos game designers e pequenas produtoras em se firmar no mercado, estabelecendo através de seus jogos uma experiência satisfatória para os usuários, dentro dos parâmetros de exigência de qualidade que eles esperam.

1.3 Objetivos

O intuito deste trabalho é apresentar a aplicação da plataforma Mixamo e seus recursos no desenvolvimento de um jogo em terceira pessoa, utilizando um personagem da plataforma como avatar, dispensando a necessidade de modelagem 3D. A pesquisa também contempla o sistema de criação de rig que a ferramenta possui, poupando tempo e trabalho na criação de um esqueleto digital. Além disso, será mostrado a produção de um fluxo de animações dentro do Unity 3D no propósito de estabelecer a mecânica primária de um personagem.

1.3.1 Objetivo Geral

Dissertar sobre os pontos em que a ferramenta Mixamo pode ser utilizada como facilitadora de processos para o desenvolvedor iniciante destinado a produzir jogos eletrônicos, cujo os parâmetros seja elaborá-los nos moldes de cunho realista.

1.3.2 Objetivos Específicos

- a) Fazer o levantamento bibliográfico e teórico relevante para a compreensão do tema proposto, apresentando um breve resumo histórico da evolução dos jogos eletrônicos, seus principais estilos gráficos e gêneros para uma melhor compreensão do tema aqui proposto.

- b) Mapear as diretrizes para abstração de mecânica e jogabilidade destrinchando seus conceitos a fim de satisfazer a compreensão do uso da plataforma Mixamo bem como demonstrar a utilização prática de seu sistema Auto-Rigger e seus benefícios como biblioteca de animações gráficas.
- c) Estabelecer as diretrizes para criação da mecânica primária em jogos em terceira pessoa usando as animações do Mixamo através do motor de jogo Unity 3D, demonstrando o desenvolvimento de mecânicas controladas por inputs de teclado mediante código.

1.4 Estrutura da Monografia

Esta pesquisa divide-se em capítulos que são de extrema importância para a compreensão do tema proposto. Esse capítulo de introdução apresenta o tema dessa pesquisa, o problema aqui apresentado e sua justificativa, apontando ainda as estruturas e objetivos em que foi realizado.

O segundo capítulo traz uma breve história da evolução dos jogos eletrônicos, tendo como objetivo dissertar sobre os dois principais estilos gráficos de jogos bem como apontar as diversas classificações, gêneros e subgêneros dos jogos atuais.

O terceiro capítulo aborda a distinção de mecânica em jogos eletrônicos, a importância que a mesma exerce e descreve o conceito de rigging e captação de movimento nos jogos 3D contemporâneos. Aponta ainda as dificuldades encontradas pelos gamers designers em conceber as mecânicas de um jogo e como o Mixamo pode ser usado para esse fim.

O quarto capítulo demonstra as principais funções da plataforma e o uso do seu sistema Auto-Rigger. É mostrado também o processo de adicionar e baixar as animações para os mais variados fins.

O capítulo cinco traz uma breve definição sobre Game Engines (*Motores de Jogos*), seu uso e relevância nas mais diversas áreas, além de demonstrar os

conceitos básicos sobre o Unity 3D para melhor compreensão do uso do Mixamo no seu ambiente virtual.

O sexto capítulo demonstra de forma metodológica, como estabelecer e configurar as animações do Mixamo no ambiente Unity 3D, destrinchando a criação de um fluxo de transições através do componente Animator Controller. Estabelece sua manipulação via código e dá luz ao embrião de uma mecânica primária.

O sétimo capítulo destina-se à discussão dos dados referentes a essa pesquisa e todas as considerações encontradas pelo autor a partir dos conhecimentos por ele absorvidos e apresentados. Além de propor estudos complementares para o tema.

2 EVOLUÇÃO E ESTILO GRÁFICO DE JOGOS ELETRÔNICOS

Este capítulo busca apresentar um breve resumo da trajetória de evolução dos jogos eletrônicos, seus diversos gêneros e classificações e especificar conceitos atrelados a seus principais estilos gráficos dispondo como objetivo uma compreensão clara do tema proposto, tendo como foco a demonstração dos principais conceitos da plataforma Mixamo a fim direcionar o game designer iniciante na produção da mecânica primária em um jogo em terceira pessoa.

2.1 Evolução gráfica em jogos eletrônicos

Em 1960, Ivan Sutherland, estudante do MIT (*Instituto de Tecnologia de Massachusetts*) defendeu seu doutorado apresentando o que é considerado hoje o primeiro sistema interativo e o precursor dos atuais geradores para produção de jogos eletrônicos. Em 1962, Steve Russell, Martin Graetz e Wayne Witanem também discentes do MIT, criaram o SpaceWar, sendo esse um dos primeiros jogos eletrônicos de computador a ser criado (CLUA; BITTENCOURT, 2005).

Mas a concepção de console de videogames surgiu só em 1966, ano em que Ralph Baer criou o primeiro protótipo para ser conectado a uma televisão (ALVES, 2005). Na década de 1970, foi desenvolvido os computadores pessoais que forneceram a produção de uma nova concepção de jogos eletrônicos (CLUA; BITTENCOURT, 2005). E baseado no jogo de Russell, Nolan Bushnell elaborou uma versão mais simples de Space War. Essa nova versão representou um salto gigantesco para o mercado de jogos eletrônicos, pois também podia ser conectada a um televisor.

Segundo Clua e Bittencourt (2005) até aquele momento, houve um significativo avanço na criação de games, mas foi por meio da criação do jogo Pong que Bushnell, o futuro criador da Atari conseguiu surpreender o mundo dos games. ao contrário do jogo Spacewar que só era jogável em grandes computadores, o game Pong podia ser distribuído ao grande público em geral.

Esse jogo foi produzido em um console menor que o dos famosos pinballs e disponibilizado em diversos locais como bares, cinemas e lojas, sendo que os jogadores poderiam comprar uma versão para o uso em suas próprias residências (ALVES, 2005). Bushnell criou a Atari, empresa de games que marcou os anos de 1970 e 1980. Em 1972 a Atari lançou o console Odyssey que era bem mais evoluído, com games com designers arrojados, mas leves e com narrativas e temáticas mais complexas, ampliando a comunicação e interatividade entre seus usuários e o jogo, diversificando e ampliando a capacidade dos games (CLUA; BITTENCOURT, 2005).

Com a diminuição dos preços dos microprocessadores foi possível um salto de qualidade na produção dos videogames, tornando possível a criação do revolucionário Sprinter (1984) pela própria Atari, sendo este o primeiro jogo que incorporou um microprocessador em sua base. Esse jogo exigia da máquina mais rapidez e gráficos realistas (ALVES, 2005). Em 1977 surgiram os primeiros consoles em cores tornando esse ano um marco na indústria dos jogos eletrônicos e no futuro nascimento dos consoles domésticos.

No ano de 1981 a IBM criou o microcomputador, hoje popularmente conhecido como PC. Além de oferecerem recursos para facilitar diversos trabalhos, os PCs ajudavam também criadores de jogos independentes que poderiam através dessa máquina, potencializar os aspectos técnicos, estéticos e narrativos nos seus próprios games .

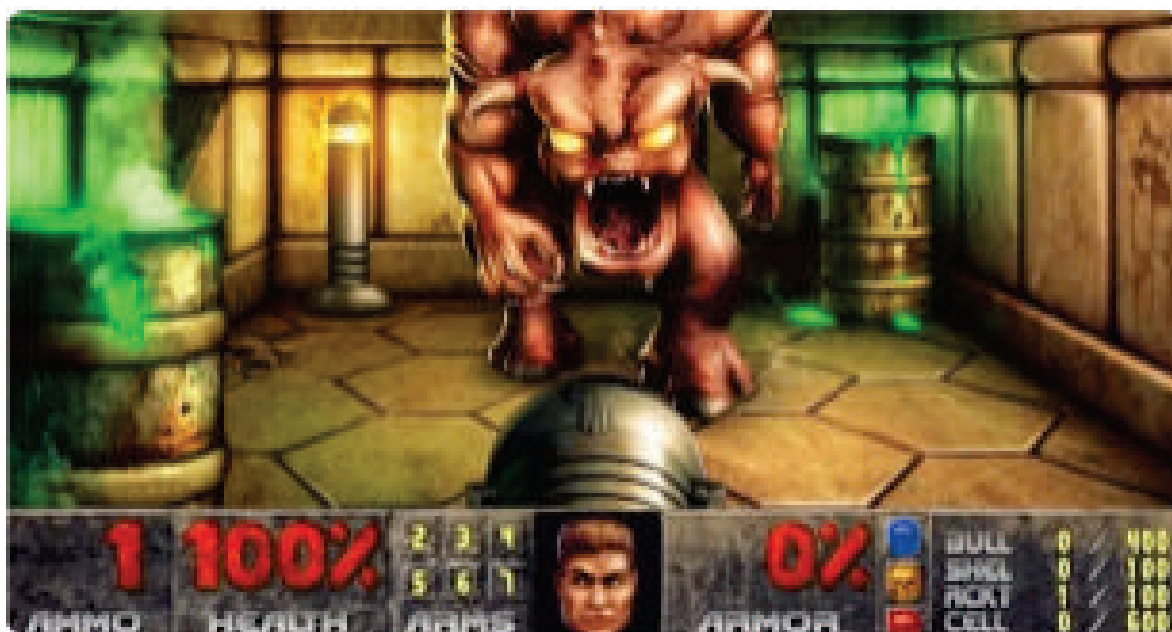
No ano de 1977 a Nintendo entra no mercado com o jogo Game & Watch. Posteriormente lançou o jogo Donkey Kong (1981) apresentando ao mundo pela primeira vez o personagem Mário, que se tornou o ícone da empresa (CLUA; BITTENCOURT, 2005).

Ainda em 1984, a Nintendo lança o Famicom, que apresentava gráficos e sons mais elaborados e superiores em relação aos que existiam na época (ALVES, 2005). Mesmo existindo outros videogames, foi em 1989 que o impacto no mercado de jogos se tornou ascendente quando a Sega lançou o Mega Drive que era até o devido momento, um console superior aos fabricados pela Nintendo. A disputa entre as gigantes dos videogames se tornou mais acirrada com a entrada da gigante Sony

no segmento. A Sony entrou no mundo dos jogos em 1991 lançando o PlayStation, o primeiro console a utilizar CD ROM ao invés dos tradicionais cartuchos.

Porém, segundo Lowood (2014) o game DOOM visto na Figura 1, lançado em 1994, marcou o início dos jogos eletrônicos modernos, sendo esse uma grande conquista no universo da programação, pois permitiu um enorme melhoramento e uma grande série de mudanças na percepção sobre jogos eletrônicos em geral.

Figura 1 — Cena do jogo Doom



Fonte: *techtudo.com* - 2014

Por conta de sua popularidade e dos demais jogos que vieram no encalço de DOOM, houve um enorme avanço na criação de comunidades de gamers e programadores, conteúdo modificável, além de uma grande atração por sons e trilhas sonoras, bem como uma preocupação constante com a representação hiper-realista de personagens e cenários, instituindo um significativo avanço em computação gráfica (LOWOOD, 2014).

Segundo Traina e Oliveira (2004), a computação gráfica é a área da computação que se destina ao estudo da manipulação, geração e interpretação de imagens e modelos utilizando-se do computador como ferramenta essencial. A computação gráfica é integrada a diversas outras disciplinas como a matemática, engenharia e física (TRAINA; OLIVEIRA, 2004). Em se tratando de jogos, essa é a parte que os tornam significativamente atrativos. As manipulações de modelos

gráficos e suas transformações geométricas e as sínteses de imagens dos jogos tanto 2D como 3D são as responsáveis pelas representações visuais a partir de especificações que as compõem.

Hoje no mercado existem uma variedade de técnicas sendo utilizadas para exibição de gráficos eletrônicos e seu conteúdo. Ao passar dos anos essas técnicas se aprimoraram. Esse aprimoramento deve-se principalmente aos avanços de hardware e poder de processamento gráfico. Segundo Traina e Oliveira (2004) é indiscutível que essa evolução se evidencia principalmente na concepção de gráficos realistas comparados aos dos primórdios dos anos 1960. Desde então, os jogos eletrônicos mudaram significativamente, deixando de ser um produto simples para se tornar um dos maiores eletrônicos já criados. Isso se deve também ao seu uso como um produto de hipermídia por grandes corporações.

Segundo Rabaça e Barbosa (2007) um produto é denominado hipermídia quando engloba imagens, sons e vídeos em movimento. Já Gosciola (2008) define hipermídia como um produto de interatividade, navegabilidade e de volume muito maior que os de multimídia. Nessa linha de raciocínio de Gosciola, é possível compreender que os jogos eletrônicos são uma hipermídia devido à interação que eles propiciam através de comandos que funcionam semelhantes à links, o que permite ao jogador escolher a direção, comandando o que quer fazer (chutar, correr, pular).

Ainda segundo Rabaça e Barbosa (2007), todo sistemas hipermidiático precisa ser abstraído como um progresso de estímulos de textos, sons e imagens destinados a quem for utilizá-los. Esse sistema sustenta como base fundamental a qualidade de estímulos sincronizados, interatividade, unidade visual, simulação dinâmica, economia de tempo e uma narrativa concisa e direta. Além de precisarem satisfazer as necessidades do usuário com uma ergonomia adaptativa.

Os jogos eletrônicos são muito apreciados no cotidiano dos jovens e dos adultos e isso repercute em suas formas de agirem e perceberem o mundo ao seu redor (PRENSKY, 2001). O grande fascínio pelos jogos eletrônicos atrai profissionais e consumidores para as áreas de artes gráficas e computação, tornando a indústria de jogos tão próspera quanto a do cinema que também se classifica como entretenimento (SANTOS; VALE, 2006). Os games de hoje em dia são criados

baseados na lógica do modelo de produção capitalista em que estão introduzidos como produtos da indústria cultural, incluindo em si diversos estereótipos, valores, relações de gênero e competitividade.

2.2 Jogos em estilo gráfico 2D e 3D

Existem vários tipos de jogos eletrônicos, mas quase todos se encaixam em dois estilos gráficos: os jogos 2D e os jogos 3D. Há muitas diferenças entre esses estilos. Nos jogos 2D usa-se projeção paralela normalmente com gráficos em bitmap. Nesses jogos são dispostos apenas dois eixos de movimento, tornando possível movimentos para cima e para baixo, bem como para esquerda e direita. Por consequência dessa restrição, jogos 2D costumam ser bem mais acessíveis de se conceber do que os jogos em três dimensões (CLUA; BITTENCOURT, 2005).

Os jogos em 3D utilizam renderização baseada em triângulos. Essa geometria triangular concede aos desenvolvedores a produção de macro cenários, complexos e estruturalmente eficientes utilizando-se de relativamente poucos recursos de arte, dividindo-as em sprites e reutilizando as mesmas de forma rotativa. De acordo com Santee (2005) na produção de jogos, tanto 2D, quanto 3D faz-se necessário a produção de imagens bidimensionais que serão utilizadas como texturas que compoem a interface gráfica outgame e ingame, como, por exemplo, janelas, botões, energia do avatar, munição, dentre outros.

Para produzir estas imagens são utilizados vários programas gráficos como o GIMP e o Adobe Photoshop (CLUA; BITTENCOURT, 2005). As mecânicas dos jogos 2D são, em sua maioria, relativamente simples por conta exatamente do seu sistema de movimento obtuso.

Como o avatar não possui uma gama complexa de movimentos, não é possível obter maior interação com os objetos do cenário, ao contrário dos jogos 3D onde o jogador pode interagir e visualizar os objetos em qualquer ângulo que desejar (ROGERS, 2013).

Outro fato nos jogos 2D é sua câmera que é empregada também de forma simplória em vista dos jogos tridimensionais. Normalmente, não há perspectiva mas diversos jogos bidimensionais utilizam o efeito de rolagem parallax, que movimenta o fundo do cenário em velocidade menor que a do primeiro plano, criando assim uma ilusão de profundidade (ROGERS, 2013)

De acordo com Rogers (2013) a jogabilidade nesses jogos, bem como sua mecânica são fáceis de se programar, por consequência, o jogador que estiver jogando com um controle inclinar o manche para a direita o personagem irá simplesmente se mover para direita, enquanto num título 3D, ao inclinar o manche para direita ele moverá o personagem tendo como base a localização da câmera do jogo e na maneira como ele estará visualizando o jogo no instante momento do comando.

Nos títulos 3D, por outro lado, os movimentos baseiam-se em planos tridimensionais, concebendo ao jogador liberdade de se locomover em um cenário que simula o mundo real, tendo a possibilidade de girar em 360 graus, onde todos objetos possuem invariavelmente as três dimensões: comprimento, altura e profundidade (ROGERS, 2013). Um exemplo é o jogo *Assassin's Creed Origins* (2017) visto na figura 2.

Figura 2 — Cena do jogo *Assassin's Creed Origins*



Fonte: ubisoft.com - 2022

Segundo Rogers (2013) não há dúvidas que os jogos 3D são mais complexos de se conceber, tendo como sua maior diferença a própria perspectiva da câmera, sendo até mesmo possível em muitos títulos, a movimentação da câmera independentemente do personagem, permitindo ao jogador visualizar o mapa do jogo em diversos ângulos e registrar a cena com sistema de fotografia.

Segundo Azevedo (2005) as animações de um personagem em títulos 3D requerem mais elaborações. Essas animações fluem umas às outras, tornando sua aparência mais homogênea e realista se comparada com a sensação rígida encontrada em quase todos jogos bidimensionais.

Nos Jogos 3D, as texturas são renderizadas na superfície da malha tridimensional dos objetos possibilitando a eles a conformação de solidez. Nesses jogos, elementos como luz e som assemelham-se com o comportamento do mundo real. Toda essa elaboração afeta significativamente a sua mecânica, a final de contas os jogos 2D normalmente apresentam objetivos superficiais, diferente dos jogos 3D onde os títulos procuram desafiar o jogador a explorar totalmente o cenário estabelecido, solucionar desafios como quebra-cabeças físicos, enfrentar inimigos que exigem uma mecânica tecnicamente apurada, dentre outras coisas (AZEVEDO, 2005).

2.3 Classificação, gênero e sub gêneros de jogos eletrônicos

Vários autores procuram discutir a evolução e a constituição dos gêneros dos jogos eletrônicos na busca de apresentar classificações para uma melhor compreensão e estudo. Mendes (2005) destaca a importância de se analisar de forma adequada as diferentes classificações e gêneros de jogos eletrônicos pegando por base suas diferenças e peculiaridades, sendo esses, fatores indispensáveis para sua compreensão.

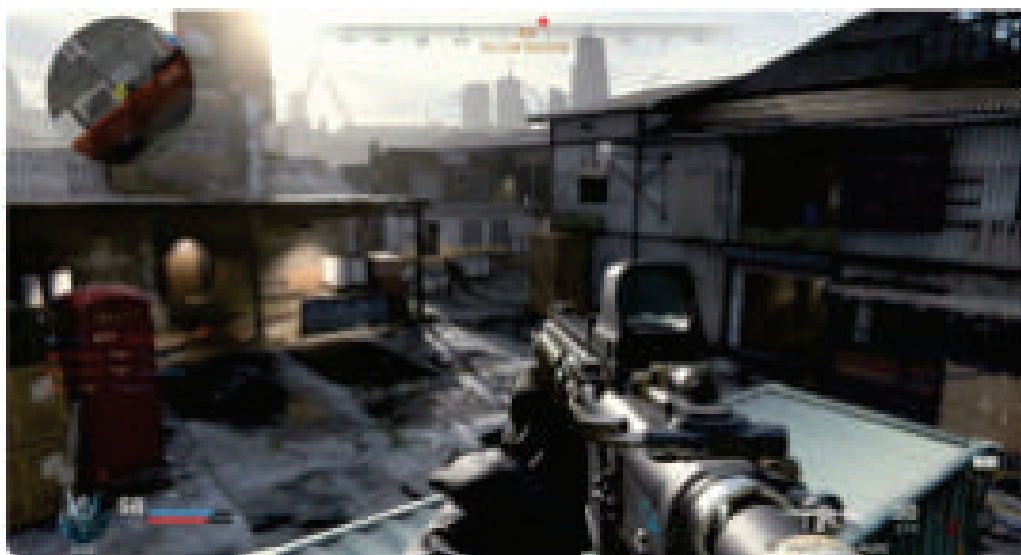
Segundo Arruda (2009) existe atualmente uma grande quantidade de categorizações em razão dos objetivos de mercado. Embora o tipo de jogo seja justamente o objetivo central da classificação, eles são categorizados levando em consideração os mais distintos aspectos como sua temática, habilidades necessárias para seu uso, grau de interação e nível de dificuldade.

A classificação dos gêneros de jogos eletrônicos concorda primordialmente com o objetivo principal estabelecido para o jogo a ser produzido. Mesmo que com o passar dos anos os jogos se dividiram em variados gêneros e subgêneros (ROGERS, 2013). Poole (2004) destaca que os gêneros de jogos mudam e se transformam ao longo de sua evolução. Alguns saem de linha por um determinado período, enquanto novos surgem.

Segundo Novak (2010) jogos classificados como de “ação” normalmente gerenciam os braços, mãos e visão do avatar e este gênero no que lhe concerne divide-se em diversos subgêneros, como por exemplo os jogos plataforma em 2D. Esses jogos também se classificam como “Aventura” como *Shadow Of The Colossus* (2005). Também podem ter lutas como principal propósito como os famigerados *Street Fighter V* (2016) e *Mortal Kombat X* (2011).

Jogos classificados como shooter possuem como principal objetivo atirar inimigos, São chamados no Brasil de “jogo de tiro”, e são conhecidos pela sigla **FPS** (*First Person Shooters*) ou “primeira pessoa”. Segundo Rogers (2013) eles pertencem ao estilo gráfico 3D. Nesse gênero, a câmera tem como base a visão do avatar, visualizando assim, apenas seus braços. O jogo *Call of Duty: Modern Warfare* (2019) mostrado na Figura 3 é um mundialmente aclamado jogo que recebe essa classificação.

Figura 3 — Cena jogo Call of Duty: Modern Warfare



Fonte: gogamers.gg - 2021

Segundo Rogers (2013) uma dos problemas desse tipo de câmera é a dificuldade em avaliar distâncias e enxergar objetos a serem recolhidos pelo jogador uma vez que estes precisam estar em uma escala maior para serem avistados. Outro desafio nesse gênero , é o efeito de “*Enjoo de Movimento*” conhecido pela sigla DIMS ou “*Motion Sickness*” (DORIGUETO; KASSE; SILVA, 2022).

Segundo os autores essa sensação de enjoo tende a ocorrer por que, em algumas situações, os olhos registram o movimento, mas o ouvido interno não, causando assim o mal estar no jogador. O efeito DIMS pode ser amenizado de forma simples, como deixar a câmera em velocidade baixa e adicionar objetos grandes para o jogador ter um ponto de foco constante.

Um dos gêneros mais jogados atualmente são os **TPS** (*Third Person Shooter*), ou jogos em “terceira pessoa” que são o foco deste trabalho. Neste gênero, o jogador visualiza seu avatar e a câmera o segue. Títulos como *Tomb Raider* (2011) e *Just Cause 4* (2018) mostrado na Figura 4 são exemplos consagrados de jogos em terceira pessoa. As câmeras em jogos TPS minimizam o já mencionado efeito DIMS, isso porque o player está sempre focado e visualizando o avatar de seu personagem.

Figura 4 — Cena jogo Just Cause 4



Fonte: psverso.com.br - 2021

Quando o jogo é classificado como de aventura, normalmente tem como foco recolher itens, organizar e gerenciar o inventário do avatar, bem como a resolução de problemas e desafios diversos. O sub gênero RPG “*Role-Playing Game*” e os jogos de “*Sobrevivência*” recebem essa classificação.

Segundo Novak (2010) nos jogos RPG o jogador interpreta um papel específico com o seu avatar. As habilidades deste personagem podem ser aprimoradas no decorrer da gameplay. Jogos como *The Elder Scrolls V* (2011) e *Fallout 4* (2015) são exemplos de jogos que recebem essa classificação.

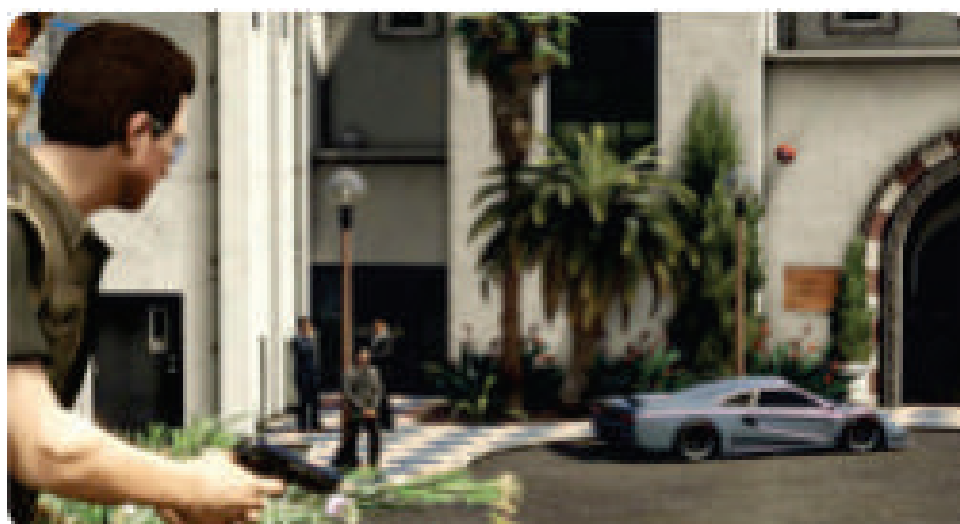
Ainda segundo Novak (2010) Os jogos da série *Silent Hill* e *Resident Evil*, também são classificados com aventura e são em TPS, mas no sub gênero “*Survival Horror*”, ou “*Jogos de Terror*”. Jogos desta categoria geralmente possuem uma temática bem estruturada com o intuito de guiar o avatar por um cenário representado normalmente como repugnantes, claustrofóbicos e obscuros. Esses jogos buscam flertar com a estética dos filmes de terror e ficção científica, onde o jogador enfrenta criaturas do imaginário sobrenatural.

De acordo com Rogers(2013) desde os primeiros jogos, os críticos e as produtoras de jogos buscam conceber classificações considerando a experiência do jogador, bem como as ações necessárias para a interatividade de um jogo. O autor destaca que as resoluções e propostas de classificação dos jogos em gêneros e do tema podem ser deveras apropriadas para a uma análise iconográfica semelhante a de filmes de Hollywood. Porém, sendo a interatividade uma parte primordial da estrutura de um jogo eletrônico, Novak (2010) considera a sua análise como a forma mais essencial para definir o gênero de um jogo.

Segundo Poole (2004) embora as diferenças existentes entre os mais variados gêneros de jogos eletrônicos sejam plausíveis de classificação, estes apresentam sempre como fundamento a simulação dos mais variados acontecimentos com atributos comuns. De acordo com Rogers (2013) existem ainda vários outros gêneros e subgêneros como simulações de esporte, vida real, corridas, estratégia e voo. Há ainda jogos que podem desempenhar vários gêneros ou subgêneros.

Ainda de acordo com Rogers (2013) o jogo *Grand Theft Auto V* (2013), mas conhecido como GTA é um exemplo disso. Além de ser classificado como um Sandbox, ou “Mundo Aberto”, esse jogo apresenta ainda uma considerável combinação de gêneros e sub gêneros como aventura, tiro em terceira e primeira pessoa, simulação de direção, vida, voo e mergulho, concebendo ao jogador a liberdade em realizar as ações desejadas não se atendo apenas ao roteiro devidamente premeditado pela narrativa do jogo.

Figura 5: Cena do Jogo GTA V



Fonte: store.steampowered.com - 2022

O entendimento das classificações e nomenclaturas, bem como as características utilizadas pelo mercado para se referir aos jogos eletrônicos, auxiliam na construção de parâmetros para a análise, discussão e produção de um jogo eletrônico. Ainda de acordo com Rogers, tendo em vista a contínua evolução dos jogos e dos suportes tecnológicos a eles relacionados, torna-se atualmente uma tarefa difícil defini-los e classificá-los.

3 MECÂNICA, JOGABILIDADE E RIGGING EM JOGOS ELETRÔNICOS

De acordo Alves (2020) na física, o termo “mecânica” é empregado para designar o estudo dos movimentos dos corpos sem considerar suas causas, sendo assim, o conceito intrínseco de mecânica é, portanto, a de um movimento considerado em si. Abstrair mecanicamente envolve uma abordagem de máquina e separada de um fluxo vivo.

No conceito de jogos segundo Rogers (2013) a mecânica é projetada para um determinado fim e montada passo a passo como um encaixe de peças, portanto, diz respeito à designação do game designer. O game designer é o responsável pela: movimentação do personagem, regras, restrições e objetos que o mesmo projetou para afetar a experiência do jogador.

No entanto, a experiência do jogador vai muito além do que o game designer estabelece e responde de forma mais orgânica com o jogo. Essa experiência é estabelecida no fluxo das sensações do jogador, sendo o resultado da imersão e diversão do mesmo. Na concepção de Rogers (2013) a mecânica em jogos são objetos que criam a sua *gameplay* quando o jogador interage e se comunica com elas. Ainda segundo o autor, toda mecânica de um jogo deve ser adequada ao tema central proposto.

Quando o usuário se depara pela primeira vez com um jogo, seu reflexo inicial é buscar a resolução de várias dúvidas sobre como jogá-lo e essas dúvidas geralmente remetem à mecânica do jogo em si. Porém, o primeiro ponto que o jogador se atém é sobre a temática do mesmo, isto é, a sua narrativa, pois é justamente o tema que define e organiza o eixo da imersão e experiência, afastando ou despertando o interesse no jogador.

O tema de um jogo como o *Tetris* (1984), é estabelecido em volta de desafios de lógica, já a proposta do já mencionado GTA envolve o confronto com a lei e os perigos e mazelas de se cometer crimes. Por limitações tecnológicas, os jogos de décadas atrás tinham como foco basicamente suas mecânicas, tendo, geralmente, uma única narrativa como pano de fundo. No entanto, mesmo um jogo não necessitando necessariamente de uma história, essa está implícita de alguma forma nele. Jogos como Tetris não exigem uma narrativa para que o jogador se interesse

em jogá-lo, porém ele possui de fato, uma narrativa por que se for observado, mesmo que um jogo não possua uma história propriamente dita, ele segue invariavelmente uma ordem lógica e cronológica de circunstância (ROGERS, 2013).

Com as grandes inovações, principalmente no campo do designer gráfico, surgiram elementos de narrativa cada vez mais espessos. Hoje temos jogos com mecânicas até simplórias em seu contexto, porém, com histórias e narrativas profundas como o aclamado *The Last of Us* (2013).

Um dos fatos que essa pesquisa evidencia é que um game designer não necessita focar em criar uma mecânica robusta e inovadora, mas essa, precisa de fato, está intrinsecamente adequada à narrativa do jogo a ser produzido.

3.1 Mecânica e Jogabilidade

É de extrema importância para um game designer saber distinguir os parâmetros referentes à mecânica e os que se referem à jogabilidade. Quando o usuário se encontra diante de um jogo pela primeira vez, ele tende a fazer confusão sobre esses dois conceitos, pois, no geral, se empregam em relação ao mesmo propósito: as ações que o jogador executa no jogo.

Segundo Rogers (2013), existem quatro tipos de mecânicas com as quais pode-se trabalhar, são elas: mecânica, perigos, quebra-cabeças e suporte. O Quadro 1 mostra a definição de cada uma delas conforme o autor.

Quadro 1 — Tipos de mecânica

MECÂNICA	Movimentos que o player executa. Alguns tipos mais comuns de mecânica são correr, andar, golpear, abrir portas, pular, escorregadios, manipulação de plataformas móveis e itens.
PERIGOS	Mecânicas programadas com o intuito de criar problemas durante a gameplay como fogos, pisos com espinhos, buracos e armadilhas.
QUEBRA-CABEÇA	São mecânicas que se movem num espaço de tempo cronometrado. São usadas normalmente para gerar perigo como blocos esmagadores, lâminas giratórias, portões e paredes ativas etc.
SUPORTES	Mecânicas que não se movem, mas são criadas para bloquear o caminho do player. Elas podem estar presentes no game na forma de degraus, âncoras, postes, ou prédios e precisam ser contornados ou pulados para o player poder seguir no jogo.

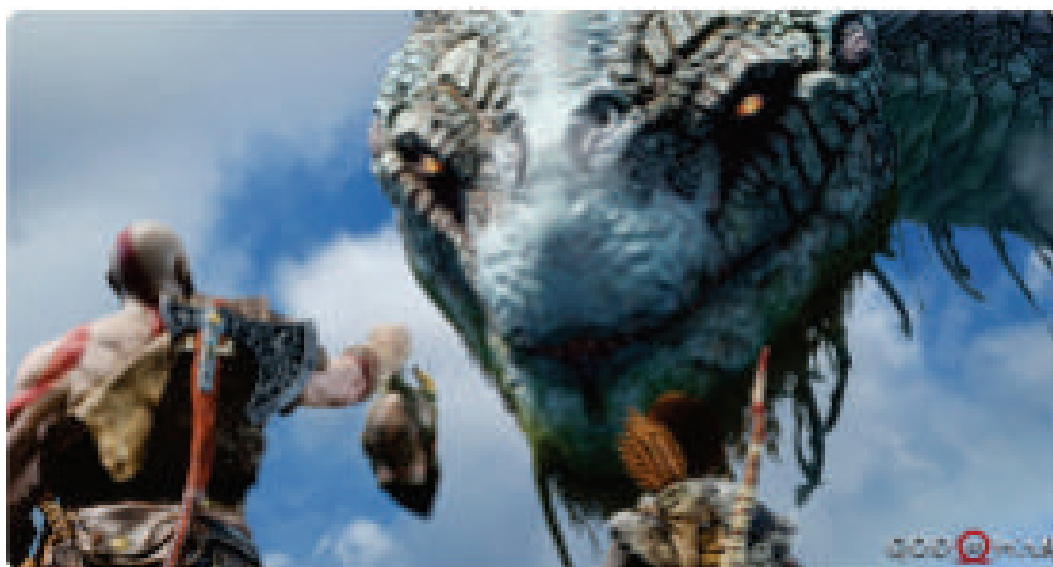
Fonte: adaptado de Rogers - 2013

Analisando o quadro acima, pode-se entender como mecânica, todas ações ou interações que o jogador exerce para se comunicar dentro de um determinado jogo. De acordo com Rabaça e Barbosa (2007) não existe processo de comunicação sem interatividade, pois a comunicação exige participação e interação no trocar de mensagens, permitindo assim, um estímulo imediato de quem recebe a informação, sobre um determinado produto ou marca.

Os autores afirmam que interatividade é a característica de um programa ou sistema de funcionar com a participação do usuário por meio de comandos e opções ao seu dispor. São ainda caracterizados pelo processo de comunicação e resposta instantânea do usuário com a máquina. Normalmente em um jogo o avatar anda, pula, voa, ataca, corre, atira, executa combos especiais e não apenas isso. Mas existem muitos estilos de jogos que o jogador não necessariamente controla um personagem, é o caso do já mencionado Tetris. Nesse jogo, as peças descem e as mecânicas que o jogador deve executar é a de rotacionar e movimentar as mesmas.

Por outro lado, temos jogos com mecânicas mais complexas como em *God of War* (2018), visto na Figura 6, onde o jogador deverá empurrar caixas, escalar, nadar, abrir baús, isso tudo como parte de sua mecânica. Ou seja, toda ação que o jogador executa no game são consideradas mecânicas, independente do jogo em questão (ROGERS, 2013).

Figura 6 — Cena de God of War



Segundo Schuytema (2008) por outro lado, a jogabilidade pode ser compreendida como um grupo de mecânicas que interagem entre si e para uma análise dos aspectos de jogabilidade nos jogos, são consideradas diversas mecânicas. Ainda tomando como exemplo o título God of War, o jogador encontra mecânicas para atacar os adversários, e se caso o jogador golpeia um inimigo, esse, no que lhe concerne, tem a opção de revidar e se voltar contra o avatar forçando o jogador a utilizar a mecânica de esquiva para não receber dano e se defender do contra ataque.

Ao observar todas essas mecânicas interagindo entre si, pode se afirmar que as mesmas trouxeram uma jogabilidade específica para o jogador, que no que lhe cabe definir se essa experiência lhe concede ou não satisfação em jogá-lo. Segundo Schuytema (2008) se analisarmos dessa forma, podemos dizer que em God Of War, as mecânicas de escalar, nadar, arrastar caixas, por exemplo, são usadas para fins diversos, o que não limita o jogador em explorar o level em que se encontra, bem como também servem para resolver puzzles e batalhas e combinam mecânicas de movimento, ataques e exploração o que no fim formam uma jogabilidade.

Ainda no raciocínio do autor, se o jogador observar God of War de forma sensata, perceberá que sua missão no jogo não passa de atravessar de um ponto A para um ponto B do level em que ele se encontra. Porém, isso seria de fato, extremamente tedioso. No entanto, é justamente nesse entendimento que a fusão entre mecânicas e jogabilidades entram em ação para tornar o jogo interessante.

Porque ao longo da trajetória, God of War nos apresenta diversas mecânicas que tornam a jogabilidade plausível, tornando consequentemente a experiência desse jogo relevante para quem o joga. A cada novo level, o jogador encontra novos desafios e precisa obter uma forma de superá-los, utilizando-se das mais variadas mecânicas que o jogo propicia.

Os levels de um game dizem respeito a uma estrutura ou a uma localização que apresentam os desafios do jogo enquanto o jogador avança em direção a seu objetivo (SCHUYTEMA, 2008). Enquanto o game designer é encarregado de elaborar as mecânicas em si, o *Level Designer* se encarrega de criar sentido entre as mecânicas e jogabilidade de um jogo.

Geralmente os jogos contemporâneos são criados dentro do esquema conhecido como MDA, mostrado no Quadro 2, desenvolvido e ensinado pelo game designer Marc LeBlanc como parte do Game Design and Tuning Workshop na Conferência Game Developers, de San José, 2001-2004.

Quadro 2 — As três etapas de criação MDA

Mechanics	Descreve as componentes específicas do jogo, no nível de representação de dados e algoritmos.
Dynamics	Descreve o comportamento da mecânica quando ela é executada pelas ações do jogador e cada um dos resultados ao longo do tempo.
Aesthetics	Descreve as respostas emocionais desejáveis evocadas no jogador, quando ele interage com o sistema do jogo.

Fonte: Adaptado de LeBlanc - 2004

Como especificado no quadro, a dinâmica de um jogo tem como alicerce sua mecânica e jogabilidade na busca de conceber uma experiência estritamente pessoal de entretenimento no jogador.

3.1.2 Classificação e níveis de mecânicas

Ainda de acordo com Hunicke, LeBlanc e Zubek (2004) em geral, as mecânicas podem ser classificadas basicamente em: mecânicas primárias(*core mechanics*) e mecânicas secundárias(*lesser mechanics*). As mecânicas primárias constituem o núcleo central da jogabilidade e são a essência no progresso do jogo. Salen e Zimmerman (2012) consideram que todo jogo tem uma mecânica primária e essa mecânica é a atividade primordial que os jogadores desempenham num jogo. Segundo eles, a definição de mecânica é estritamente direcionada para as ações do personagem principal.

Segundo Hunicke, LeBlanc e Zubek (2004) as mecânicas secundárias, porém, são mais complexas, podendo também alterar uma mecânica primária ou mesmo influenciar nos mais variados aspectos de um jogo. Sendo a mecânica um sistema de regras e objetivos que conduz a ação do jogador sobre o que pode ou não ser executado, pode se afirmar também que esse sistema é induzido pelo

tempo, isto é, são criados “estágios” de ação para o jogador “progredir” durante o desenvolvimento de sua jornada (HUNICKE; LEBLANC; ZUBEK, 2022).

Esses estágios normalmente baseiam-se em segundos, minutos e horas e em alguns jogos, até mesmo em dias, determinando assim, diferentes “níveis” de mecânicas. Todo jogo, portanto, impõe seus limites às escolhas que o jogador pode tomar, em menor ou maior grau, em contraponto, novas possibilidades e opções são criadas pelos gamers designers e exploradas pela indústria dos games atualmente. (HUNICKE; LEBLANC; ZUBEK, 2004).

A elaboração de uma mecânica eficaz e seus aspectos técnicos tendem a considerar os desejos do jogador e como elas podem influenciar nas suas escolhas e no seu modo de jogar, considerando a sua opção por determinados tipos de personagens.

Salen e Zimmerman (2012) acredita que um jogo se faz da combinação de pequenos componentes que se influenciam de forma mútua, e nesse processo específico é que a mecânica se torna o papel decisivo. Já Sicart (2008) define a mecânica como algo que conecta as ações dos jogadores, com a determinação do jogo e seus principais desafios. Ressalta também que são métodos invocados por agentes para interagir com o mundo do jogo, tendo como encargo, as regras estabelecidas. Segundo Fabricatore (2014) são subsistemas atômicos e interativos, baseados em regras capazes de receber, gerenciar e reagir produzindo efeitos.

De modo geral, os parâmetros definidores da mecânica de um jogo tomam resolução durante o processo de produção, considerando aspectos gerais do jogo, como seu gênero, seu público-alvo e o grau de dificuldade esperado. Em se tratando de personagens, as mecânicas não dizem respeito somente aos elementos técnicos, como também aos diferentes graus de identificação entre o próprio jogador com o avatar. Segundo Rogers (2013) quando um jogo é centrado no jogador, sua criação, bem como sua produção, precisam levar em consideração aspectos como: suporte a experiência de jogo, facilitar seu aprendizado e fornecer graus de complexidade na experiência.

O game designer ao conceber um jogo precisa levantar as considerações do seu estilo e público-alvo. A mecânica de um jogo precisa ser projetada para contemplar as habilidades do jogador. Da mesma forma, os erros e fracassos também devem ser imputados ao jogador e não por fatores que se encontram fora de sua influência ou controle (ROGERS, 2013).

De acordo com Fabricatore (2014), a jogabilidade é o elemento crucial e decisivo na escolha e avaliação de um jogo, tendo em mente que para a maioria dos jogadores ela é mais importante que a narrativa, o contexto ou as próprias determinações do jogo. Compreender de forma analítica a mecânica possibilita ao game designer conceber e pré-ordenar cursos de interação, porém não consegue prever como o jogo será jogado, ou qual será o resultado dessa experiência para quem o joga (SICART, 2008).

Abstrair corretamente a mecânica fornece ao game designer os parâmetros para elaborá-las de forma conveniente com o gênero e estilo de jogo pretendido. Se o intuito for criar um jogo em terceira pessoa, essa abstração prévia torna-se crucial para a elaboração do sistema de rigging adequado para o avatar.

3.2 Rigging de personagens 3D

Segundo Clua e Bittencourt (2005) rigging é uma técnica de animação tridimensional que consiste em adicionar movimentos a um personagem 3D por meio da construção de um “esqueleto virtual” que serve para simular a estrutura de um corpo e empregar nele movimento fisicamente realista. Esse esqueleto é também denominado de “Armature” em muitos softwares especializados em rigging. Essa técnica é empregada numa enorme quantidade de segmentos hoje no mercado, seja em jogos, filmes ou comerciais de TV e se tornou indispensável na produção de animações de personagens tridimensionais.

Geralmente um profissional em modelagem 3D cria um personagem utilizando uma “malha” tridimensional para ser inserido no game. A modelagem tridimensional é a área da computação gráfica que tem o objetivo de criar objetos em três dimensões, animar imagens e renderizar cenas. A modelagem 3D é responsável por conceber e dar vida aos personagens, objetos e cenários de um

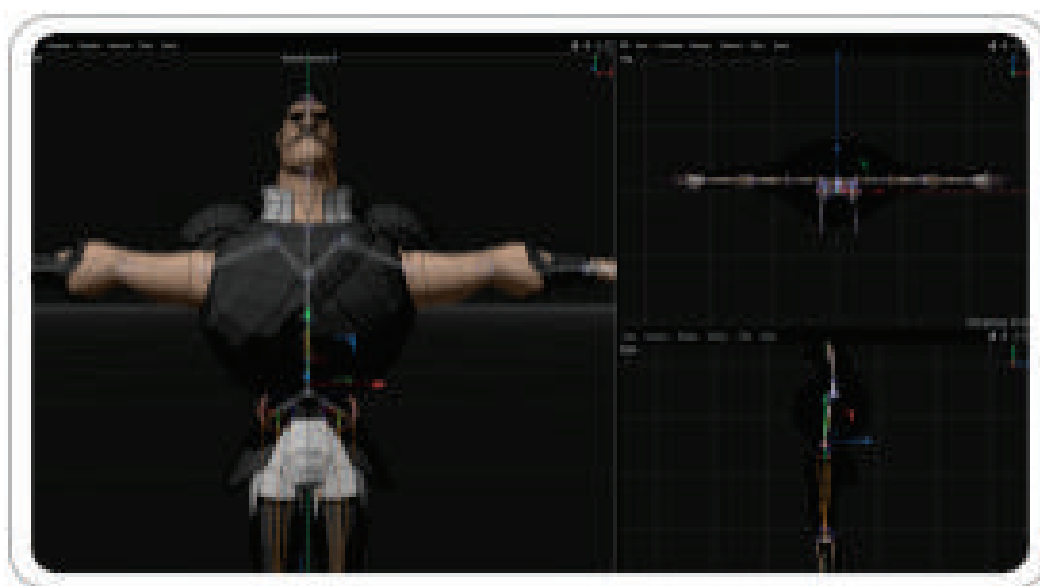
game (CLUA; BITTENCOURT, 2005). A Figura 7 mostra um exemplo de malha 3D do modelo **Warrior Base Mesh**, disponível gratuitamente no site Free 3D.com (FREE 3D, 2022). A Figura 8 mostra o rigging criado pelo autor do estudo para este personagem utilizando a plataforma Adobe Mixamo, tema central desta pesquisa que será abordado no capítulo seguinte.

Figura 7 — Exemplo de malha tridimensional



Fonte: adaptado do Cinema 4D pelo autor do estudo

Figura 8 — Estrutura de um esqueleto virtual humanóide



Fonte: adaptado do Cinema 4D pelo autor do estudo.

A animação esquelética é primordial para a otimização do trabalho de animação, tornando esse processo menos enfadonho e dificultoso. O processo de rigging transforma o personagem estático em uma “marionete”. Assim, o personagem obtém estrutura necessária para ser articulado, evitando um resultado deformado ou distorcido em suas animações.

Por meio da estrutura dessa armadura digital e das articulações do personagem, os profissionais são capazes de inserir movimentos que auxiliam na produção das animações, deformando assim a malha em questão, gerando o movimento pretendido. No Quadro 3 é mostrado os principais conceitos e termos sobre esse tema segundo os autores.

Quadro 3 — Principais conceitos de rigging

RIGGER	Denominação do profissional especializado em rigging de personagens 3D
SKELETAL ANIMATION	É uma técnica de animação por computador em que uma personagem é representada por duas partes: superfície que define aspectos da personagem (skin) e um conjunto hierárquico de ossos (bones) interligados (skeleton ou armature) utilizados para animar a malha.
RIG	É o resultado do trabalho do rigger, é a armadura ou esqueleto que irá ser utilizado pelo animador para animar a personagem.
SKINNING	É o processo de associação da armadura (skeleton) à malha (skin).
JOIN HIERCHY	É a estrutura da armadura, a hierarquia entre bones. Existem uns bones que são child (filho) e outros que são parent (pai), o movimento destes últimos afeta o dos primeiros.
ROOT BONE	É o nome do bone que está em primeiro lugar na hierarquia (não tem de ser o primeiro a ser criado). Se mover o root bone, todos os outros bones também são afetados porque diretamente ou indiretamente são childs.
TIP, BODY e ROOT	São nomes dos elementos que constituem um bone.

Fonte: autor do estudo.

Como visto no quadro acima, o profissional responsável pelo rigging é denominado “Rigger”. Ele precisa se especializar na área e obter um sólido conhecimento no software em que deseja trabalhar.

O rigger insere a partir do desenho estático do personagem, os movimentos desejados a partir de uma noção do desempenho dos movimentos orgânicos, tudo isso claro, utilizando um software específico para essa função. Para a execução de uma simples animação, é preciso que a Armature gire, se movimente ou caminhe de forma a simular os movimentos do mundo real.

A técnica de rigging não se limita a criação de movimentos apenas em modelos “humanóides”, ela pode ser usada para deformar a malha de qualquer tipo de personagem como dinossauros, monstros e animais. Existe atualmente uma abundância de softwares dedicados a esta função como o Blender, Maya, 3D Max e o Cinema 4D (CLUA; BITTENCOURT, 2005).

Essa técnica revolucionou a forma de se produzir animações em três dimensões, tendo como marco o lançamento do primeiro filme da série *Toy Story* (1995) visto na Figura 9. Toy Story foi o primeiro longa metragem completamente digital, e todos seus personagens foram animados através dessa técnica e desde então a indústria dos games também passou a utilizá-la maciçamente na produção de jogos 3D (ÉPOCA NEGÓCIOS, 2019).

Figura 9 — Cena de Toy Story



Fonte: pixar.com - 2022

Se o rigger deseja que seu personagem execute um determinado movimento, este precisa ser previamente concebido e para cada um desses movimentos, precisa ser gerada uma animação. Através do sistema de rigging obtém-se resultados mais satisfatórios e realistas. Inserir o esqueleto no personagem é provavelmente o processo mais básico na produção de rigging, bastando ao rigger introduzir os bones (ossos) nas articulações exatas onde devem estar na anatomia do personagem em questão.

Tendo suas disposições devidamente estabelecidas, para o processo funcionar corretamente, os ossos e articulações precisam obedecer uma hierarquia. A primeira articulação é chamada de “Root Bone” ou “Raiz” ou até mesmo de “Pai”. Uma vez estabelecido o root bone, todas as outras articulações estarão subordinadas direta ou indiretamente a ele.

Ao rotacionar e movimentar uma articulação pai, todos os outros ossos vinculados a ela também se movimentam. Sendo assim, movimentar um osso do braço por exemplo, movimenta também todos os ossos da mão e do antebraço ligados a ele, isto é, os ossos, considerados "filhos".

Segundo Schuytema (2008) um aspecto essencial para a construção de uma armadura de rigging é a cinemática inversa. IK (*Inverse Kinematic*) é uma técnica de programação de movimentos que tem como objetivo garantir que os personagens se conectem de forma física com o ambiente do jogo, para que a interação entre esses dois elementos se assemelhe mais orgânica.

Na cinemática inversa, o rigger não precisa definir a rotação e posição de cada articulação de forma individual, o que resultaria num enorme esforço. Nesse caso, o profissional se incube de manipular apenas a ação, uma vez que o algoritmo do software seleciona os ângulos apropriados para os membros que estão participando do movimento em questão.

3.2.1 Dificuldades em criar rigging e animações

Em geral, segundo Sicart (2008), a tarefa de estruturar e movimentar um esqueleto não é fácil mesmo para os mais gabaritados profissionais. Para desenvolver essa etapa é preciso um vasto conhecimento em software a ser utilizado, topologia, bem como conhecimentos específicos de anatomia e composição humana, suas articulações e possíveis movimentos. Isso somente em detrimento de rigging em personagens humanoides. Em personagens não humanoides, como, por exemplo, dragões e monstros, ou ainda em animais como felinos e répteis, o profissional precisa de um vasto conhecimento em anatomia animal e de um vasto estudo sobre os movimentos desses personagens em questão.

Se o rigger, como exemplo, decide que seu avatar retire uma espada de suas costas, desejando com isso obter um grau satisfatório de realismo, não basta apenas mover as articulações da mão do personagem até ela, pois isso geraria um movimento impreciso e truncado ao olho humano. Neste caso, é preciso calcular a posição necessária do tronco e dos membros, bem como o punho, cotovelo e ombro que participarão da ação para que essa mão se localize próximo à bainha.

O nosso cérebro é capaz de fazer esse cálculo de forma automática, porém, se o rigger deseja obter veracidade, ele precisa ter em mente todos esses detalhes para obter animações satisfatórias. O profissional nessa área precisa sempre estar atualizado em processos de computação gráfica e design 3D, aumentando assim, a precisão e realismo na elaboração de suas animações. Para concepção de animações existem hoje várias técnicas e equipamentos destinados à captura de movimentos para os mais diversos fins.

3.3 Captura de Movimento

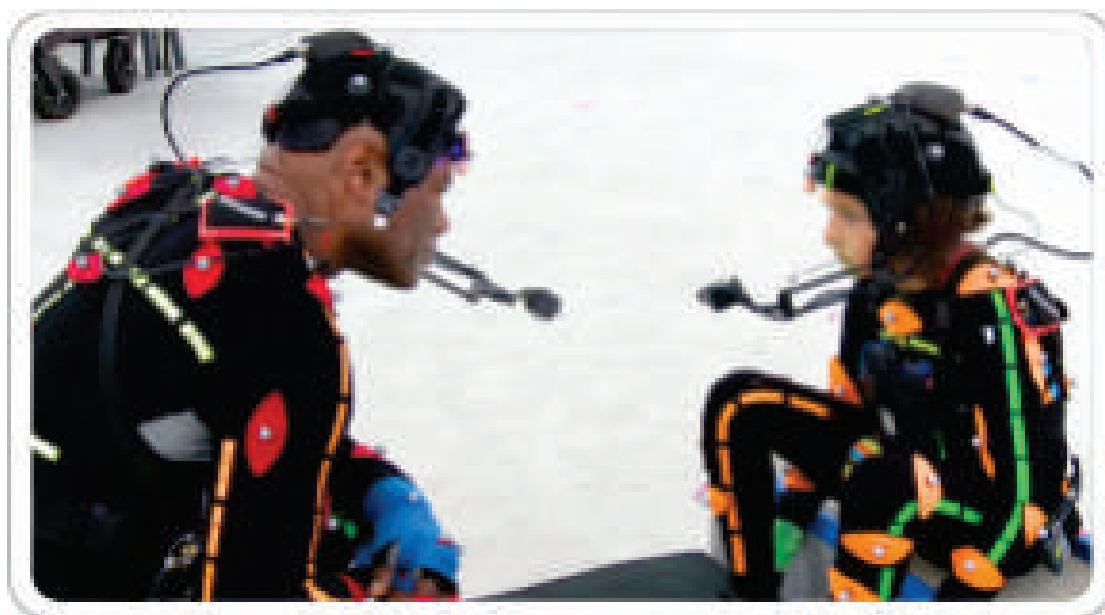
Nas décadas de 1980 e 1990 era preciso bastante improviso e criatividade para se chegar a um resultado mais realista e satisfatório em animações de ações relativamente simples. Existia uma carência por uma tecnologia que permite-se obter o realismo desejado para o mundo digital.

O surgimento da técnica de Motion Capture (MoCap), ou captura de movimentos, proporcionou um nível de realismo jamais visto antes nos jogos eletrônicos. O game designer Jordan Mechner foi o precursor desta técnica. Ele filmou seu irmão correndo, pulando e saltando e inseriu a resolução desses movimentos no clássico jogo Prince Of Persia de 1989 (CLUA; BITTENCOURT, 2005).

Essa ideia revolucionária foi considerada um avanço na produção de animações para jogos ainda no início da década de 1990. Desde então, a captura de movimentos evoluiu exponencialmente e possibilitou o surgimento de jogos com uma gama de realismo em suas animações indistinguíveis da vida real.

Hoje em dia, a captura de movimentos funciona com a utilização de várias câmeras e sensores de movimento que rastreiam as ações de atores e objetos. Os sensores são colocados em tudo que precisa ter seus movimentos capturados. A Figura 10 mostra o sistema de MoCap utilizado no jogo God of War.

Figura 10 — Captação de movimentos em God of War



Fonte: *Hardware.com.br* - 2022

Os dados obtidos pelas câmeras são devidamente enviados para computadores munidos de um programa que projeta o já mencionado esqueleto

virtual no personagem, no qual, os movimentos dos atores, bem como dos objetos, são reproduzidos em tempo real. Com esses dados em mãos, o game designer e seus demais responsáveis obtêm uma visão inicial do desempenho dos atores em cena, o que os possibilita de fazer devidos ajustes nos movimentos se for necessário antes mesmo de enviar para pós-produção os modelos em questão.

Além disso, outros processos e técnicas são necessários para garantir melhores resultados. Esses processos em jogos *Triple-AAA* podem durar anos para serem concluídos.

3.4 Mecânica básica em jogos 3D em 3ª pessoa com o Mixamo

Como visto no capítulo 2, os jogos em terceira pessoa fazem parte do gênero e estilo gráfico 3D. Nesses jogos, o jogador visualiza seu avatar num ângulo de 45 graus ou de cima, e passa a ter a sensação de estar dentro do jogo. Segundo Rogers (2013) nesse gênero, o realismo é maior do que os shooters em duas dimensões, por conta dos seus gráficos tridimensionais, mecânica e jogabilidade.

Para produzir um FPS, os desenvolvedores precisam se preocupar com a movimentação da câmera e as partes do avatar que ela capta (SCHUYTEMA, 2008). Porém, como já mencionado, nos jogos em terceira pessoa o personagem jogável é visível, o que torna esse tipo de jogo mais difícil de se produzir, pois é preciso conceber todo o design do avatar, além da necessidade de uma quantidade considerável de animações para dar vida aos seus movimentos.

Existem inúmeras ferramentas disponíveis no mercado que disponibilizam animações pré estabelecidas no intuito de facilitar o desenvolvimento de TPS. A proposta deste trabalho como foi descrito no capítulo 1 é justamente destrinchar o uso de uma dessas ferramentas: o Adobe Mixamo. As animações do Mixamo também podem ser utilizadas para criação de mecânicas para NPCs “*Non-Playable Character*”, para os diversos gêneros de jogos. Neste trabalho, seu uso será demonstrado apenas em detrimento das mecânicas primárias do avatar de um jogo em terceira pessoa.

Com o Mixamo o game designer poderá criar sua mecânica dispensando a necessidade de técnicas de Motion Capture bem como a participação efetiva de um profissional especialista em rigging, diminuindo significativamente os custos de produção do seu projeto. Poderá também dispensar a participação de um modelador 3D para criação de seu avatar, caso utilize um dos diversos personagens disponibilizados na plataforma.

Mecânicas secundárias também podem ser criadas com as animações do Adobe Mixamo, mas seu uso não será abordado nesta pesquisa.

4 UTILIZAÇÃO DO MIXAMO COMO FACILITADOR DE PROCESSOS

O Adobe Mixamo é uma plataforma online de personagens e animações. O Mixamo tem sede em San Francisco, Califórnia e em 2015 foi adquirida pela gigante Adobe. Desde então, seus recursos foram integrados ao Creative Cloud, o pacote de programas criado e comercializado pela própria Adobe (ADOBE, 2022).

O Mixamo permite adicionar e manipular dados de captura de movimento, provendo uma imensa variedade de personagens e suas diretrizes para uso direto nos mais variados fins. Além de fornecer animações realistas que podem ser adicionadas em personagens principais ou secundários de um game. Conta ainda com um banco de dados de rastreamento de movimentos tridimensionais, bem como a construção de rigging no padrão humanóide (ADOBE MIXAMO, 2022).

Os personagens 3D disponibilizados pelo Mixamo podem ser usados nos mais variados tipos de trabalho, como desenhos, fotografias e filmes. O game designer tem a opção de importar um personagem da plataforma para um programa que suporte modelos 3D e aplicar animações diversas nele.

Um exemplo de programa que suporta modelos 3D é o Adobe Photoshop. Um personagem do Mixamo pode ser importado como um arquivo DAE para esse programa. O profissional poderá aplicar efeitos diversos, mudar o background da composição de modo a dar mais personalidade e vida ao personagem e produzir imagens estáticas bem com animações.

Diferente dos demais aplicativos da Adobe, seu uso não precisa de uma assinatura. Para ter acesso ao conjunto completo de recursos do Mixamo, o game designer deve se inscrever com uma conta gratuita da Adobe (ADOBE MIXAMO, 2022). A imensa biblioteca de animações e seu uso didático torna o Mixamo uma sólida escolha no experimento de modelos tridimensionais em jogos eletrônicos.

4.1 Auto-Rigging e personagens do Adobe Mixamo

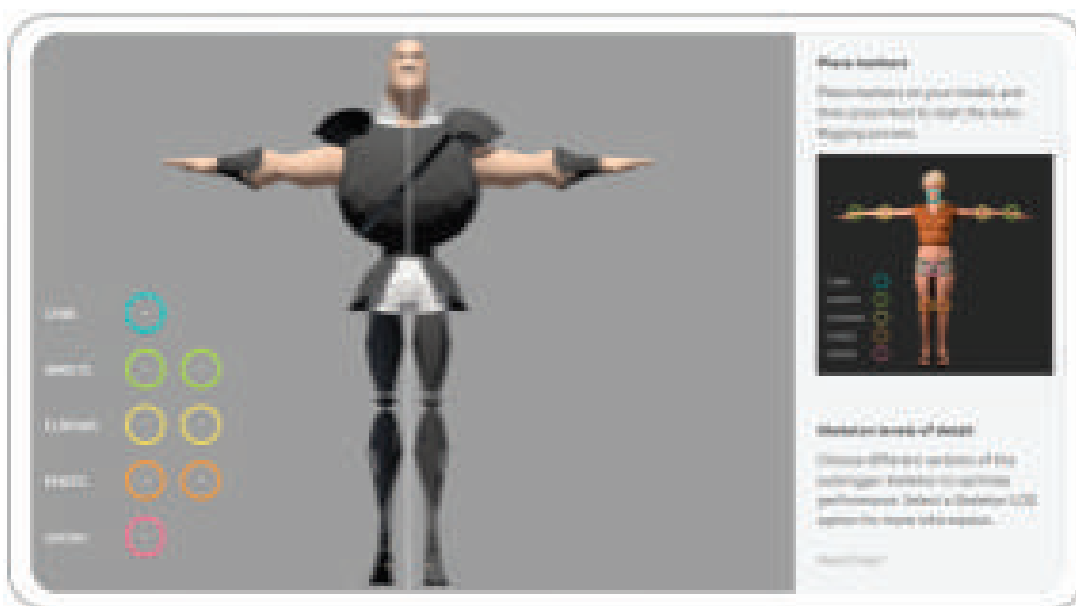
Conforme dito anteriormente, para ter acesso às ferramentas do Adobe Mixamo, o game designer deve criar e fazer login com sua conta da Adobe em

Mixamo Homepage (ADOBE MIXAMO,2022), se o game designer não possui um personagem modelado à sua disposição, ele pode usufruir de vários personagens pré definidos, clicando na guia “Characters” na parte superior da tela e escolher um de sua preferência.

Se o game designer já possuir um personagem modelado e deseja utilizá-lo, deverá clicar no botão “Upload Character” no lado direito da tela. O Mixamo aceita arquivos OBJ ou FBX, bem como ZIP.

Uma vez escolhido o arquivo, é necessário arrastar para a janela de download para iniciar o processo de importação. Para que o aplicativo funcione corretamente, o modelo precisa ser modelado em T Pose (*topologia em T*), mostrado na Figura 11.

Figura 11 — Criação de rigging no Mixamo



Fonte: adaptado do Mixamo pelo autor do estudo

Como visto, até o momento dessa pesquisa, a plataforma não aceita modelos que não sejam compatíveis com a anatomia humana. Para essa pesquisa foi utilizado o já mencionado personagem **Warrior Base Mesh** para fim de demonstração do Auto-Rigger do Mixamo.

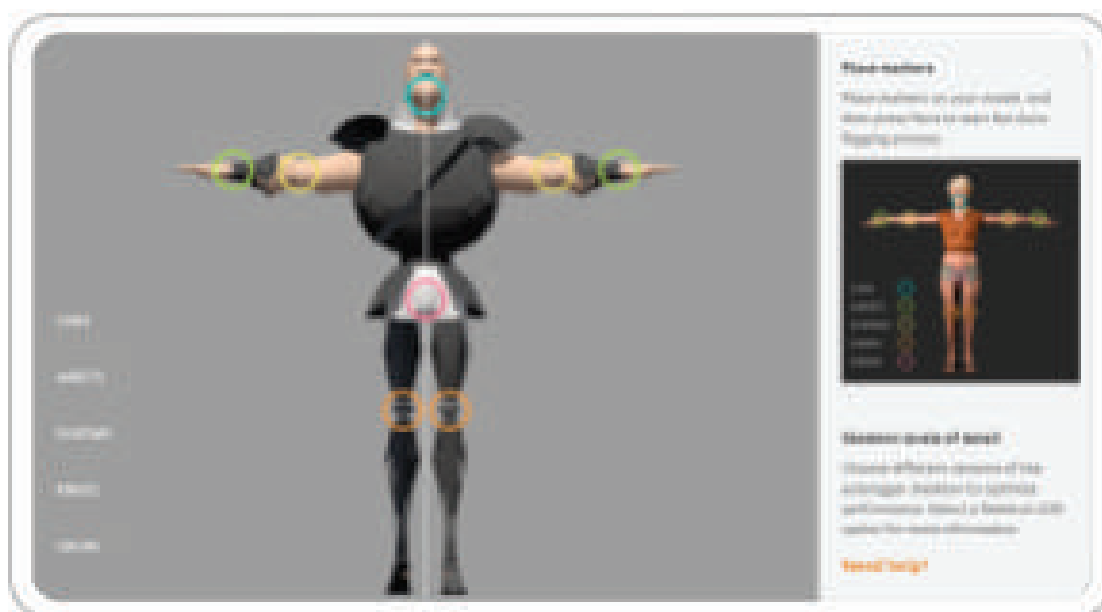
Quando carregado o modelo 3D, o software abre a janela Auto-Rigger onde o game designer será solicitado a girar o personagem fazendo com que fique de frente

no aplicativo. Para isso é necessário utilizar os botões na parte inferior esquerda do visualizador. Feito isso, é preciso informar à plataforma os pontos exatos de introdução do esqueleto, arrastando os círculos de articulação para as áreas do queixo, pulsos, joelhos, cotovelos e virilha do personagem 3D (ADOBE MIXAMO, 2022).

Tendo os círculos devidamente posicionados, basta clicar em “Next” para finalizar o processo. O Mixamo faz cálculos em sua base de dados e gera um esqueleto compatível com suas animações, o que leva segundos de processamento.

Concluído o processo, a plataforma carrega o modelo com uma animação pré-definida. A Figura 12 demonstra o devido posicionamento dos círculos de articulação no Auto-Rigger do Mixamo.

Figura 12 — Auto-Rigger devidamente posicionado



Fonte: adaptado do Mixamo pelo autor do estudo

Se o usuário estiver satisfeito com o resultado, deverá clicar em “Next” para continuar. Caso sejam observados movimentos indesejados ou erros, basta voltar e reposicionar os círculos novamente. Uma vez estabelecido o rigging, o game designer deve clicar na guia Moves localizada na parte superior da tela para começar a animar seu personagem. Para visualizar o modelo de todos os ângulos é preciso clicar e arrastar ao redor do mesmo para movimentá-lo através da câmera

do Mixamo. É possível, ainda, utilizar o botão de rolagem do mouse para diminuir ou aumentar o zoom.

4.1.2 Adicionando animações no Adobe Mixamo

Com seu personagem estabelecido no Mixamo, o game designer poderá personalizar a maneira que ele se movimenta ou fica de pé. Tendo ainda a opção de aplicar imagens animadas ou estáticas a ele. Na janela esquerda existe um catálogo com pesquisa de gifs e poses. Os modelos em vermelho são para projetos femininos e os azuis masculinos.

Essas diferenças alteram significativamente a maneira de se comportar do modelo. O usuário pode explorar uma variedade de opções e encontrar dezenas de gifs de movimentos como correr, dançar, caminhar, empunhar armas, dentre outros (ADOBE MIXAMO,2022).

Encontrando uma animação desejável, é preciso clicar em cima do gif para adicioná-lo ao personagem. Usando o botão de "Play", e "Pause" pode-se controlar a animação em andamento. Pode-se ainda clicar e arrastar o indicador de reprodução do círculo para voltar ou avançar rapidamente a animação. Uma vez aplicada a animação, obtém-se acesso a controles deslizantes no menu à direita responsáveis pelos ajustes das configurações da animação. Muitas animações possuem suas próprias configurações individuais. Para configurá-las, basta manipular os controles disponíveis para a mesma.

Citando como exemplo, se o usuário escolhe a animação "Capoeira" para fazer seu modelo gingar como um capoeirista, o controle deslizante "Posture" dessa animação determina quanto o personagem movimenta as pernas e tronco durante a ginga. Como foi mencionado, é possível encontrar diferentes configurações para cada animação, porém, existem duas configurações pertinentes em praticamente todas elas: Overdrive e Trim.

Enquanto a opção Overdrive ajusta a velocidade da animação, a opção Trim permite excluir partes da animação que o usuário não deseja para seu projeto. Tendo estabelecido a animação, o usuário poderá baixá-la e importá-la para seu

projeto no programa desejado, clicando no botão "Download" no lado direito da tela (ADOBE MIXAMO,2022).

As animações estão disponíveis no formato FBX ou, como mencionado anteriormente, em DAE. Se o game designer pretender baixar seu modelo e animações para um programa de ilustração ou designer gráfico deverá escolher o este último, pois é o único formato compatível com esses tipos de programas.

No entanto, se o intuito for descarregar o mesmo numa Game Engine como o Unity 3D ou programas de modelagem tridimensional como o Cinema 4D, é recomendado baixar em FBX. Se o game designer pretende usar seu modelo no Unity é preciso baixar no formato FBX for Unity. Ao clicar em baixar, aparecerá também uma "lista suspensa" com as opções de FPS (*taxa de quadros por segundo*). O usuário deverá escolher a taxa que mais se adequa ao seu projeto.

5 DESENVOLVIMENTO DE JOGOS COM UNITY 3D

Neste capítulo o objetivo é elucidar os principais conceitos sobre games engines e destrinchar os passos fundamentais para o uso do Unity 3D em jogos a fim de propiciar uma melhor compreensão do tema proposto.

5.1 Conceito de Motores de Jogos

Game Engine, ou motor de jogo, são uma classe de softwares que possuem um conjunto de bibliotecas para o desenvolvimento de jogos eletrônicos e aplicações gráficas. Dentre as funções encontradas em uma game engine inclui: um motor gráfico para renderização; motor de física para detecção de colisões; efeitos sonoros; suporte para criação de animações e inteligência artificial. Além disso, as games engines contam com suporte para linguagens de programação diversas como C# (C Sharp), Lua e JavaScript (GREGORY, 2018).

O jogo DOOM, como dito no capítulo 2, causou uma revolução no mundo dos games. A partir dali, os novos programadores perceberam que não precisavam mais construir um jogo do início. Ao invés disso, eles utilizaram os núcleos dos jogos-base de suas próprias criações.

Segundo Oliveira (2013) embora a denominação Game Engine tenha sido usada somente a partir da década de 1990, alguns dos primeiros motores de jogos já tinham sido concebidos em meados da década de 1980 como o Adventure Game Interpreter, conhecido pela sigla AGI. O autor afirma ainda que os motores de jogos permitem a reutilização de softwares, gerando um aumento significativo em produtividade, praticidade e foco direto nas funcionalidades, regras e características e fundamentos na produção de um jogo (OLIVEIRA, 2013).

Anteriormente, os códigos de um game eram escritos linha por linha, bem como cada um dos pixels dispostos na tela, fazendo-se necessário o esforço de uma grande equipe de programadores para desenvolver um único título. Para cada novo jogo eram escritos uma nova biblioteca de códigos, exigindo assim, bastante tempo para serem desenvolvidos. Os desenvolvedores tinham que refletir sobre todas as

limitações de sua época, mantendo os códigos o mais simples possível na busca por otimização do hardware.

Jason Gregory (2018), afirma que o mercado atual de jogos é uma indústria de entretenimento de centenas de bilhões de dólares, competindo diretamente com Hollywood em popularidade e tamanho. Segundo ele, as games engines são os softwares responsáveis por esse impulso de popularidade e grandiosidade, tais como Valve Source Engine, Quake, Doom Da id Software, Epic Unreal Engine 4 e o Unity 3D.

Elas se tornaram kits de desenvolvimento reutilizáveis de software, com recursos de criação e licenciamento capazes de dar vida a qualquer jogo imaginável. Essa contínua evolução dos motores de jogos estabeleceu uma invariável divisão entre, script, renderização, arte e level design (GREGORY, 2018).

Com o avanço das games engines e das tecnologias em sua integridade, elas não são mais utilizadas apenas para desenvolvimento de jogos. São concebidas na área de simulação de arquitetura bem como na medicina, em casos de treinamento cirúrgico simulado (NIEMINEN, 2021).

5.2 Sobre o Unity 3D

O motor de jogo Unity 3D, foi utilizado no desenvolvimento de diversos jogos, como *Praey for the Gods* (2019) e *Osiris: New Dawn* (2020) e atualmente se encontra na versão 5.

Segundo Corrêa (2022) essa engine permite desenvolver jogos para quase todas as plataformas disponíveis no mercado atual, como Linux ou Windows. Além disso, o seu uso também é empregado em plataformas móveis como Android, e iOS, bem como para consoles como Xbox, Nintendo Wii e Playstation, utilizada também para produção de games online.

Segundo Nascimento (2022), o Unity 3D possui inúmeras vantagens em detrimento das outras engines no mercado. Uma delas é a possibilidade de usufruir elementos produzidos por outros profissionais.

A loja da Unity disponibiliza uma variedade de elementos gráficos, alguns gratuitos e outros pagos como texturas, materiais, modelos de personagem e cenário e pequenos jogos em versão Beta.

A loja oficial do Unity se chama Asset Store, é por ela que o usuário pode obter seus recursos. Essa loja pode ser acessada de dentro do Unity 3D, bastando ao usuário conectar-se à internet. O Unity dispõe também de um tutorial de aprendizado bastante robusto para o desenvolvedor. No site da plataforma o usuário encontra disponível também a documentação necessária para o desenvolvimento e utilização das classes do Unity em seus scripts.

5.2.1 Conceitos iniciais do Unity 3D

Os conceitos principais do Unity 3D são: Scenes, Game Objects, Terrains, Models, Materiais e Texturas. O Unity procede com o conceito de cenas, as quais levam a denominação de Scenes dentro do programa.(NASCIMENTO,2022)

De acordo com Corrêa (2022) dentro dessas cenas, são inseridos os Games Objects. Qualquer elemento ou objeto inserido na cena recebe essa denominação, como terrenos, luzes, câmeras, personagens e objetos. São utilizados materiais, shaders e texturas para criar profundidade nos modelos gráficos. Os materiais são baseados em shaders e texturas. Os shaders são o que define como será desenhado, enquanto as texturas determinam o que será desenhado na superfície do material.

Outro elemento de grande importância é o terreno. Ele é utilizado para construção de todo tipo de cenários dentro de uma cena e por ele o avatar percorre o jogo (UNITY DOCUMENTATION, 2022). O Unity permite a implementação de modelos e materiais diversos, o que possibilita um maior grau de realismo do jogo como grama, água, árvores, pedras e outros.

Outro conceito bastante utilizado nos projetos desenvolvidos no Unity 3D são os prefabs *objetos “pré-fabricados”* sendo estes conjuntos de modelos armazenados para serem utilizados (UNITY DOCUMENTATION, 2022). Um ponto também de

grande importância é a utilização de luzes e câmeras. No Unity 3D, cada projeto pode se valer de várias câmeras e luzes conforme a necessidade do desenvolvedor.

Como visto no capítulo 4, as animações e personagens do Adobe Mixamo podem ser utilizados em praticamente todos os motores de jogos disponíveis no mercado. Nesta pesquisa será mostrado seu uso através do Unity 3D.

6 CONTROLE DE ANIMAÇÕES NO AMBIENTE UNITY 3D

Este capítulo foi desenvolvido mediante pesquisa, usabilidade e experiência do autor com as ferramentas aqui apresentadas a fim de demonstrar os recursos do Adobe Mixamo dentro do ambiente Unity 3D. As informações estabelecidas foram coletadas através de conhecimentos de programação adquiridos pelo mesmo no curso Análise e Desenvolvimento de Sistemas no Instituto Federal de Educação, Ciência e Tecnologia (IFTO) e os conhecimentos na área de jogos foram extraídos de diversos cursos, bem como em dados compartilhados em comunidades de gamers e fóruns de programação voltada para desenvolvedores.

Além disso, foram aplicadas como base teórica os Termos de Uso Gerais da Adobe e o Manual do Usuário do Unity 2021.3 (LTS). No Apêndice deste trabalho será disponibilizado o código para o controle das animações citadas nesta pesquisa e no Anexo encontra-se o Termo Adicionais do Mixamo. O código foi elaborado em C# (*C Sharp*) pelo autor deste estudo.

6.1 Ajustes de animações no Mixamo

Como visto no capítulo 4, o Mixamo dispõe de um sistema de Auto-Rigger. o usuário pode escolher um personagem da plataforma, ou fazer upload do seu próprio personagem bastando clicar no botão “Upload Caractere”. Uma vez estabelecido o modelo na plataforma, o usuário encontra uma biblioteca de animações dispostas na aba Animations e baixa as que mais convêm para seu projeto gerenciando-as antes, através da câmera do Mixamo.

O usuário deve utilizar a barra de busca localizada no canto superior esquerdo da tela para filtrar a pesquisa por animações. É possível buscar uma animação específica digitando uma palavra-chave como “Jumping” ou “Standing” ou até mesmo utilizar as tags da plataforma para filtrar sua pesquisa. O Mixamo possui as seguintes tags: Combat, Adventure, Sport, Dance, Fantasy, Superhero e Skinning Test.

Segundo Nascimento (2022) é possível ainda ajustar certos parâmetros de animação antes de baixá-las, como velocidade de execução, intensidade, espaço entre membros, postura do personagem dentre outras. Se o usuário deseja que seu personagem execute uma animação sem se deslocar no jogo, ele pode marcar a opção “In place” para esta animação, isso permite com que ela seja apenas reproduzida, mantendo o personagem no mesmo ponto onde for executada (NASCIMENTO, 2022).

6.1.2 Controle das animações no Unity 3D

Para controlar as animações do Mixamo no Unity 3D, foi utilizado nessa pesquisa o componente Animation Controller. Este componente é o asset que armazena informações sobre o gerenciamento das animações de um personagem ou de objetos não jogáveis (UNITY DOCUMENTATION, 2022). Para este trabalho foi utilizado o personagem Vanguard disponível na plataforma. Para criar sua mecânica, foram utilizadas as seguintes animações: **Idle** (parado), **Walking** (andando), **Punching** (golpeando) e **Kicking** (chutando) como visto na Figura 13. Essas animações foram baixadas no formato FBX for Unity. Para o cenário foi baixado o pacote **Demo Grass Flowers** disponível gratuitamente na Asset Store (UNITY ASSET STORE, 2019).

Figura 13 — Avatar com as respectivas animações



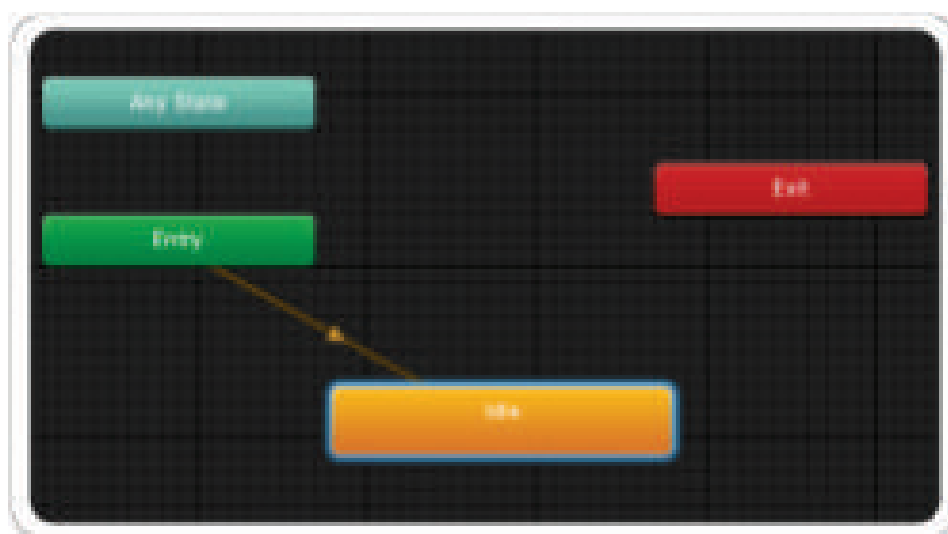
Fonte: adaptado do Mixamo pelo autor do estudo.

A primeira animação precisa ser baixada com sua “Skin” (Malha) escolhendo a opção “With Skin” (Com Malha), isto faz com que a malha tridimensional do personagem seja incluída no Download. As demais animações precisam ser baixadas com a opção “Without Skin” (Sem Malha), disponibilizando apenas seus movimentos e parâmetros (NASCIMENTO, 2022).

Segundo Corrêa (2022) no ambiente Unity 3D, o usuário pode importar as animações e organizá-las em uma pasta a fim de obter um melhor fluxo de trabalho. Após criar uma “Scene” é preciso inserir a animação que possui a malha 3D. O personagem será reconhecido como um objeto, pois como visto no capítulo 5, todos os elementos de uma cena no Unity 3D recebem essa denominação.

Visualizando a janela “Inspector”, pode-se observar que esse objeto possui um componente incluso gerado automaticamente pelo Unity. Esse componente é justamente o Animator Controller. Porém, não existe ainda um “Controlador” atrelado a ele. Para tanto, faz-se necessário selecionar o personagem na janela Assets, clicar com o botão direito do mouse na opção “Create” e depois em “Animator Controller”, desta forma foi criado um “Controller” para o componente do personagem, que neste trabalho foi renomeado para **ControllerAvatar**. Clicando duas vezes nesse objeto é possível acessar a janela “Animator”, vista na Figura 14.

Figura 14 — Janela Animator



Fonte: adaptado do Unity 3D pelo autor.

Através dessa janela o usuário controla o sistema de fluxo de animações do seu personagem, inserindo as animações pretendidas dentro dela. Arrastando a animação **Idle** para a janela Animator, pode-se notar a criação de uma seta laranja conectando o estado de entrada de cena **Entry** para **Idle**.

Essa seta é denominada “Transition” e através dela é possível controlar individualmente cada uma das animações que devem ser executadas e definir o momento que elas precisam ocorrer. Apertando “Play” para visualizar a animação, foi observado que quando chega ao seu final, a animação **Idle** simplesmente para, porém, essa animação precisa continuar em loop infinito. É necessário clicar sobre a janela Assets e através do Inspector na janela Animations, marcar a opção de “Loop Time” (UNITY DOCUMENTATION, 2022).

6.1.3 Criando fluxos de transições para estabelecer a mecânica

Clicando em **Idle** com o botão direito do mouse, foi escolhida a opção “Make Transition” e conectada a seta Transition com a animação **Walking**. Também foi executado o inverso, criando uma transição de **Walking** para **Idle**. Fazendo isso, obtêm-se as transições da animação **Idle** para **Walking**, bem como de **Walking** para **Idle**. Neste ponto pôde ser observado que após a animação **Idle** termina, a animação **Walking** foi rodada logo em sequência, porém no jogo com certeza não será isso que o desenvolvedor almeja.

Para o gerenciamento das transições de forma satisfatória, isto é, executá-las separadamente e no momento específico que o jogador desejar, faz-se necessário utilizar os parâmetros embutidos no próprio Animator Controller (NASCIMENTO, 2022). O Animator Controller possui quatro parâmetros: Trigger, Bool, Integer e Float, especificados no Quadro 4. Esses parâmetros são responsáveis por controlar as transições entre as animações (UNITY DOCUMENTATION, 2022).

Quadro 4 — Parâmetros do Animator Controller

INTEGER	Representa um número inteiro
FLOAT	Representa um número com uma parte fracionária
BOOL	Representa um valor verdadeiro ou falso (representado por uma caixa de seleção)
TRIGGER	Representa um parâmetro booleano que é redefinido pelo controlador quando consumido por uma transição (representado por um botão circular)

Fonte: Adaptado de Manual do Usuário do Unity 2021.3(LTS). - 2021.

Neste caso precisa ser adicionado um novo parâmetro do tipo Bool para cada animação e definir se ela será ou não executada. Para a melhor compreensão e fluxo no desenvolvimento, é recomendado renomeá-los. Neste trabalho os parâmetros foram renomeados para **EstouParado**, **EstouAndando** e **EstouGolpeando**.

Segundo Corrêa (2022) nessa etapa é preciso condicionar a transição **EstouParado** para **EstouAndando**, selecionando a transição referente na janela Inspector, encontrando a propriedade “Conditions” e selecionando o parâmetro **EstouAndando**. Executando isso, o parâmetro controla o momento exato em que a animação **Idle** deve ser executada.

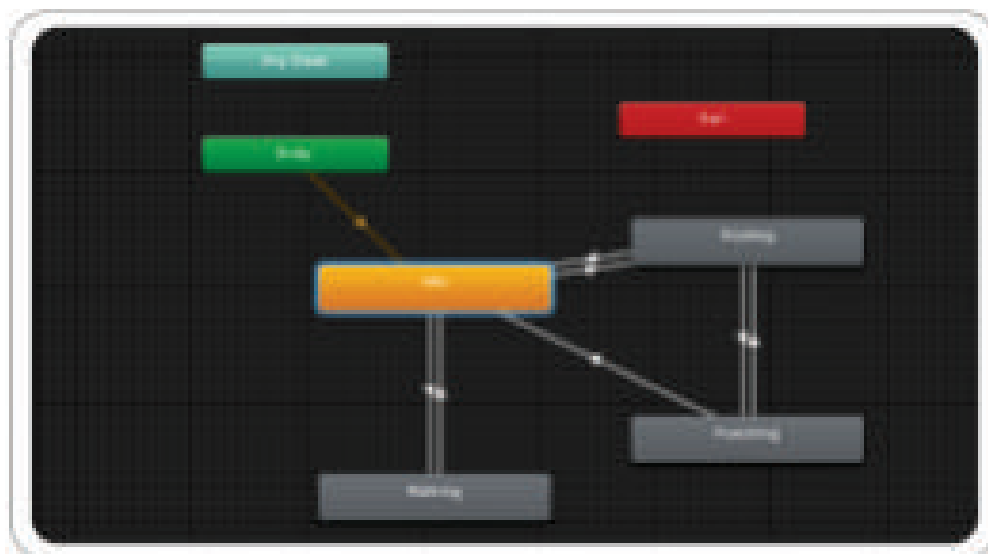
A animação **Walking** ocorre apenas quando o parâmetro **EstouAndando** for true(verdadeiro). Esse controle, porém, precisa ser feito via código. Terminado esse processo, o script pode ser adicionado ao personagem. Foi realizado o mesmo processo na transição **EstouAndando** para **EstouParado**, porém utilizando o parâmetro **EstouParado**.

Precisa ser criada também as transições de **Idle** para **Punching**, utilizando o parâmetro **EstouGolpeando**, e uma de **Punching** para **Idle**, utilizando o parâmetro **EstouParado**. Para que **Kicking** seja efetuada, o parâmetro **EstouGolpeando** também precisou ser condicionado para a reprodução dessa animação. Também foram elaboradas transições de **Punching** e de **Kicking** para **Idle** utilizando-se do parâmetro **EstouParado** que recebe true.

Para que a animação **Kicking** seja executada depois de **Punching** caso o jogador continue atacando, faz-se necessário também conceber uma transição de

Punching para **Kicking**, utilizando o parâmetro **EstouGolpeado** que recebe true novamente. Elaborando todo esse processo foi obtido o fluxo visto na Figura 15.

Figura 15 — Fluxo de animações no Animator



Fonte: adaptado do Unity 3D pelo autor.

Testando o jogo novamente constatou-se que, embora estabelecido o fluxo completo, o personagem só executa a animação **Idle**. Isto ocorre porque ainda deve ser gerenciado os estados dos parâmetros dentro do **ControllerAvatar**.

Esse gerenciamento é feito através do código. Analisando o código, pode-se observar que na linha "var EstouMovendo = Input.GetAxis ("Vertical") a intenção não é fazer com que o personagem execute a animação **Walking**, esse código apenas armazena uma validação se o jogador está ou não pressionando as teclas de movimentação pela vertical, neste caso, as teclas S e W do teclado.

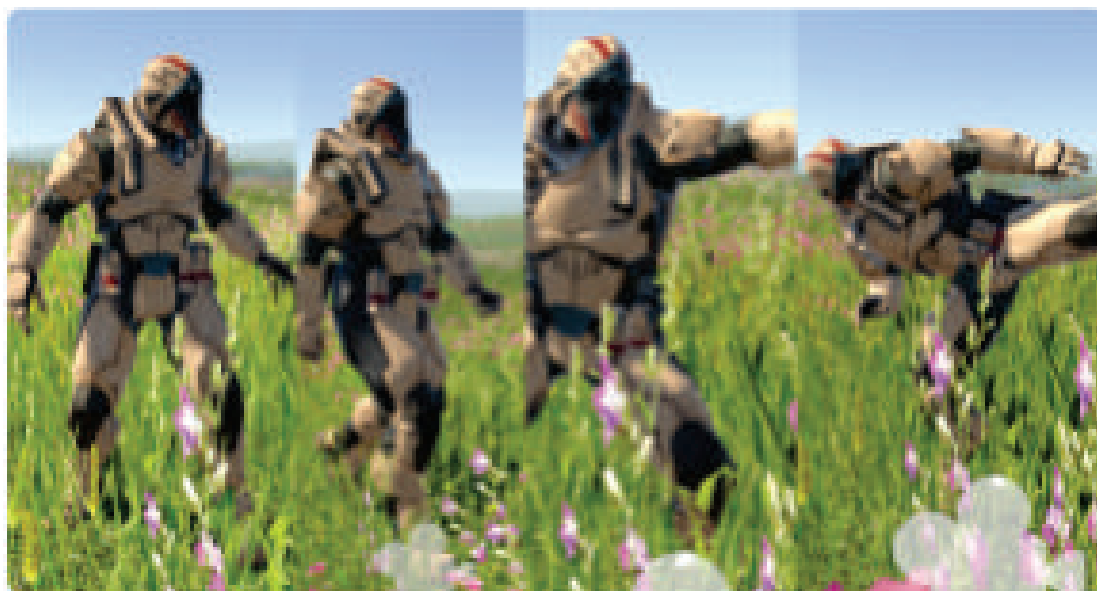
No código, o componente Animator está sendo referenciado através do objeto "animParousia" tornando possível obter acesso aos seus parâmetros mediante o método "SetBool", que necessita como argumento, o nome do parâmetro e o estado em que ele deve se comportar, neste caso, true ou false.

Sendo assim, o código checa se o jogador está pressionando as teclas referentes aos eixos verticais, se caso estiver, ele determina que o parâmetro **EstouAndando** receba true e **EstouParado** receba false.

Se o jogador soltar essas teclas, o personagem volta para animação **Idle** e o parâmetro **EstouParado** recebe true, enquanto o parâmetro **EstouAndando** recebe false. Para rodar a animação **Punching**, o código verifica se o jogador está ou não pressionando a tecla correspondente, se caso esteja, o parâmetro **EstouGolpeando** recebe true, em contrapartida **EstouParado** recebe false, evitando assim indesejáveis conflitos. Quando o jogador solta essa tecla, **EstouGolpeando** receberá false e o personagem volta para a animação de **Idle**.

Apertando play neste ponto foi constatado que todas as animações estão estabelecidas dentro do Unity 3D. Na Figura 16 é mostrada as animações ocorrendo. No entanto observou-se que mesmo pressionando as teclas correspondentes, as animações demoram a acontecer, pois somente quando termina uma, a próxima se inicia.

Figura 16 — Animações estabelecidas no Unity 3D



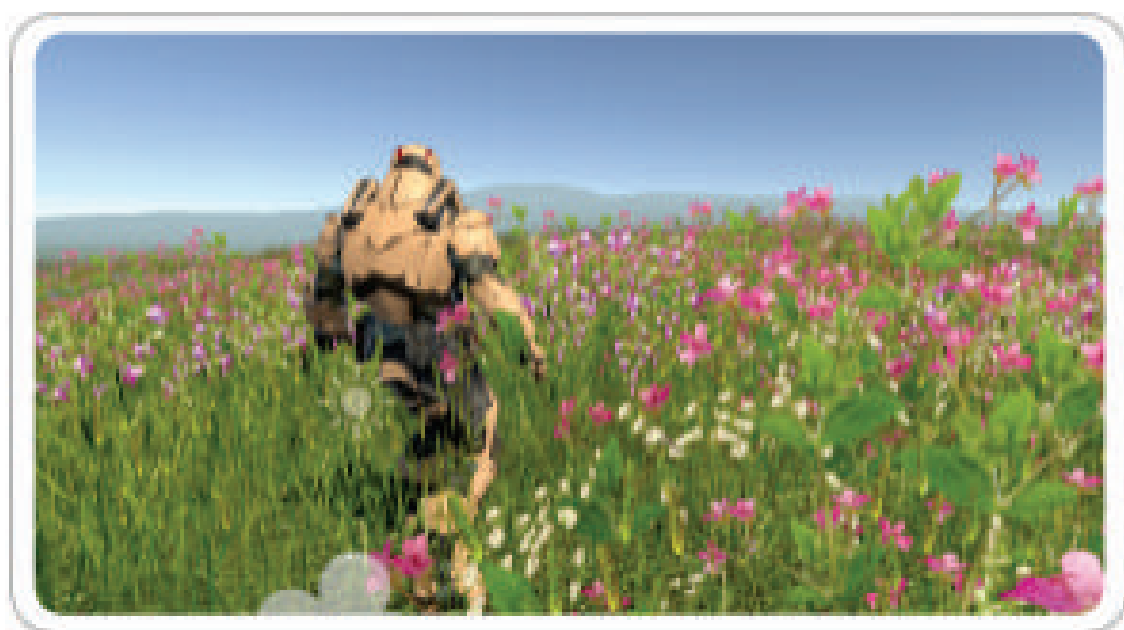
Fonte: adaptado do Unity 3D pelo autor.

Esse problema foi resolvido selecionando as transições **EstouParado** para **EstouAndando**, **EstouAndando** para **EstouParado** e **EstouParado** para **EstouGolpeando** através do Inspector e desmarcada a opção “Has Exit Time”.

Essa propriedade faz com que a próxima animação só se execute se a anterior for finalizada. Desmarcando-a, o parâmetro quando satisfeito interrompe a animação independente do frame em que se encontra (UNITY DOCUMENTATION, 2022).

Testando o jogo novamente, foi denotado que as animações do avatar ocorreram de forma desejável, e respondem aos comandos estabelecidos mediante o código. A Figura 17 demonstra a mecânica em funcionamento.

Figura 17 — Avatar com suas mecânicas



Fonte: *adaptado do Unity 3D pelo autor.*

A partir desse ponto, o desenvolvedor obtém os conhecimentos relevantes para estabelecer suas mecânicas primárias no Unity 3D usufruindo das animações do Adobe Mixamo. Para incrementá-las, faz-se necessário selecionar o restante das animações que condizem com a proposta do avatar que o mesmo almeja desenvolver e seguir os passos que essa pesquisa evidenciou.

7 CONSIDERAÇÕES FINAIS

A opção por abordar a plataforma Adobe Mixamo deve-se ao uso eficiente de seus recursos no desenvolvimento de jogos eletrônicos, propiciando uma biblioteca de personagens e animações realistas, um sistema de criação de rigging extremamente didático que dispensa a colaboração de profissionais como riggers, designer gráfico e modelador digital além de ser compatível com praticamente todas as engines disponíveis no mercado.

A importância desse estudo fica clara também ao se checar o anseio de novos gamers designers em estabelecer em seus jogos animações realistas. Como visto, os iniciantes na área em geral não dispõem de conhecimentos de modelagem tridimensional e recursos de MoCap. Isso torna a experiência com o Mixamo não só bastante satisfatória como determinante na criação de um jogo.

Para os interessados nessa área cabe ainda pesquisar o desenvolvimento de e mecânicas secundárias através do sistema de Blend Trees do Unity 3D para complementar os conhecimentos aqui adquiridos.

É proposto ainda o estudo sobre o comportamento dos NPCs através de ferramentas de inteligência artificial do próprio Unity 3D aliadas às já estudadas animações do Mixamo, a fim de estabelecer a jogabilidade entre eles e o avatar.

REFERÊNCIAS

ADOBE MIXAMO. **Mixamo**. [S. l.], 14 nov. 2022. Disponível em: <https://www.mixamo.com/#/>. Acesso em: 11 nov. 2022.

ADOBE MIXAMO. **Upload and rig 3D characters with Mixamo**. [S. l.], 2022. Disponível em: <https://helpx.adobe.com/creative-cloud/help/mixamo-rigging-animation.html>. Acesso em: 17 set. 2022.

ADOBE. **Termos Adicionais do Mixamo**. [S. l.], 22 mai 2021. Disponível em: https://www.images2.adobe.com/content/dam/cc/br/legal/servicetou/Mixamo-Addl-Terms-pt_BR_20210623.pdf. Acesso em: 1 nov. 2022.

ALVES, Lynn. **Game Over: Jogos Eletrônicos e Violência**. [S. l.]: Futura, 01/01/2005. 256 p. ISBN 8574131989.

ALVES, Nickolas. **Mecânica Clássica**. São Paulo: Instituto de Física, 2020. https://alves-nickolas.github.io/pdf/Mec%C3%A2nica_Cl%C3%A1ssica.pdf.

ARRUDA, E. P. **Jogos Digitais e Aprendizagem**: O jogo Age of Empires III desenvolve idéias e raciocínios históricos de jovens jogadores. Orientadora: Lana Mara de Castro Siman. 2009. 237 p. Tese (Pós Graduação) - Faculdade de Educação/UFGM, Belo Horizonte-MG, 2009. Disponível em: <https://docplayer.com.br/3551166-Eucidio-pimenta-arruda-jogos-digitais-e-aprendizagens-o-jogo-age-of-empires-iii-desenvolve-ideias-e-raciocinios-historicos-de-jovens-jogadores.html>. Acesso em: 6 jul. 2022.

ASSETSTORE.UNITY.COM/. **Grass Flowers Pack Free**. Standard Unity Asset Store 05/02/2019. Extension Asset. Disponível em: <https://assetstore.unity.com/packages/2d/textures-materials/nature/grass-flowers-pack-free-138810>. Acesso em: 13 ago. 2022.

AZEVEDO, Eduardo. **Desenvolvimento De Jogos 3D E Aplicações Em Realidade Virtual**. [S. l.]: Campus, 2005. 320 p. ISBN 8535215697.

CLUA, E. W. G; BITTENCOURT, J. R. **Desenvolvimento de Jogos 3D**: Concepção, Design e Programação. [S. l.], 2005. Disponível em: https://www.academia.edu/15907500/Desenvolvimento_de_Jogos_3D_Concep%C3%A7%C3%A3o_Design_e_Programa%C3%A7%C3%A3o. Acesso em: 7 ago. 2022.

CORREA, Gunnar. **Desenvolvimento de jogos 3D com Unity 2017 + 2018**. Udemy. Disponível em: <https://www.udemy.com/course/desenvolvimento-de-jogos-3d-com-unity-2017/>. Acesso em: 09 nov. 2022.

DORIGUETO, R. S.; KASSE, C. A.; SILVA, R. C. **Cinetose**: Motion Sicknes. Revista Equilíbrio Corporal e Saúde, São Paulo, p. 51-56, 4 dez. 2022. Disponível em: <https://seer.pgsskroton.com>. Acesso em: 13 out. 2022.

ÉPOCA NEGÓCIOS. **Toy Story 1 foi marcante como 1ª animação toda produzida em computador.** [S. l.], 16 jun. 2019. Disponível em: <https://epocanegocios.globo.com/Tecnologia/noticia/2019/06/epoca-negocios-toy-story-1-foi-marcante-como-1a-animacao-toda-produzida-em-computador.html>. Acesso em: 13 jul. 2022.

FABRICATORE, Carlo. **Gameplay and game mechanics design**: a key to quality in videogames. Conference Paper, OECD-CERI Expert Meeting on Videogames and Education, p. 4-16, 30 maio 2014. DOI 10.13140/RG.2.1.1125.4167. Disponível em: https://www.researchgate.net/publication/236168267_Gameplay_and_game_mechanics_design_a_key_to_quality_in_videogames. Acesso em: 19 ago. 2022.

FREE3D.COM. **Free Warrior Base Mesh.** baq studio 20/11/2019. Personagem 3D. Disponível em: <https://free3d.com/3d-model/base-mesh-warrior-free-3d-model-26122.html>. Acesso em: 12 jul. 2022.

GOGAMERS.GG. **Principais gêneros de jogos e suas características.** 16 maio 2021. fotografia 1280 x 720. page 2. Disponível em: <https://gogamers.gg/gamepedia/principais-geros-de-jogos/>. Acesso em: 3 jul. 2022.

GOSCIOLA, Vicente. **Roteiro para Novas Mídias**: do game à tv interativa. São Paulo-SP: Senac, 2008. 280 p. ISBN 8573597119.

GREGORY, Jason. **Game Engine Architecture.** 3. ed. [S. l.]: A K Peters, 17/08/2018. 1240 p. ISBN 1138035459.

HARDWARE.COM.BR. **O que é motion capture?**: 6 jogos que ficaram marcados pelo uso desta tecnologia. 1 jun. 2022. fotografia 1060 x 596. page 3. Disponível em: <https://www.hardware.com.br/artigos/o-que-e-motion-capture-6-jogos-que-ficaram-marcados-pelo-uso-desta-tecnologia/>. Acesso em: 27 out. 2022.

HUNICKE, R.; LEBLANC, M.; ZUBEK, R. **MDA**: uma abordagem formal para o design e a pesquisa de jogos. Modelo MDA, [s. l.], p. 1-11, 4 dez. 2004. Disponível em: https://edisciplinas.usp.br/pluginfile.php/4575137/mod_resource/content/2/MDA_traduzido.pdf#:~:text=O%20MDA%20%C3%A9%20uma%20abordagem,a%20pesquisa%20de%20jogos%20t%C3%A9cnicos. Acesso em: 13 out. 2022.

LOWOOD, Henry. **Game Engines and Game History.** Games Engines, History of Games International Conference Proceedings, p. 2-19, jan. 2014. DOI 1916-985X. Disponível em: https://www.kinephanos.ca/Revue_files/2014-Lowood.pdf. Acesso em: 20 out. 2022.

MCSHAFFRY, Mike; GRAHAM, David. **Game Coding**: Fourth Edition. 4. ed. Boston, MA - USA: Cengage Learning, 05/03/2012. 960 p. ISBN 133-77657-4. MENDES, C. L. **Jogar jogos eletrônicos**: que lazer é esse?. Jogos eletrônicos, Licere, Belo Horizonte - Brasil, ano 2005, v. 8, n. 1, p. 52-64. 2005.

MIXAMO, Adobe. **Animate 3D characters with Mixamo**. [S. l.], 2022. Disponível em: <https://helpx.adobe.com/creative-cloud/help/animate-characters-mixamo.html>. Acesso em: 17 ago. 2022.

NASCIMENTO, Wiliam. **Jogos 3D com Unity + modo multiplayer**. Udemy. Disponível em: <https://www.udemy.com/course/jogos-3d-com-unity-2017-modo-multiplayer/>. Acesso em: 29 out. 2022.

NIEMINEN, Topi. **Unity Game Engine in Visualization, Simulation and Modelling**. Faculty of Information Technology and Communication Sciences, jul. 2021. Disponível em: <https://trepo.tuni.fi/bitstream/handle/10024/133433/NieminenTopi.pdf?sequence=2>. Acesso em: 3 ago. 2022.

NOVAK, Jeannie. **Desenvolvimento de Games**. Tradução da 2ª Edição. São Paulo-SP: Cengage Learning, 2010. 472 p. ISBN 9788522106325.

OLIVEIRA, E. R. **O uso de engines para o desenvolvimento de jogos eletrônicos**. Orientador: Roque Mendes Prado Trindade. 2013. 67 p. Monografia (Bacharel em Ciência da Computação) - UESB, Vitória da Conquista, 2013. Disponível em: <http://www2.uesb.br/computacao/wp-content/uploads/2014/09/O-USO-DE-ENGINES-PARA-O-DESENVOLVIMENTO-DE-JOGOS-ELETR%C3%94NICOS-esdras-rocha-de-oliveira.pdf>. Acesso em: 13 jul. 2022.

PIXAR.COM. **Toy Story That Time Forgot**. fotografia 250 x 173. page 2. Disponível em: <https://www.pixar.com/toy-story-toons-1>. Acesso em: 14 nov. 2022.

POOLE, Steven. **Trigger Happy**: Videogames and the entertainment revolution. [S. l.]: Arcade Publishing, 2004. 248 p. ISBN 109887654321.

PRENSKY, Marc. **Digital Natives, Digital Immigrants**. 5. ed. University Press, 2001. Disponível em: <https://www.marcprensky.com/writing/Prensky%20-%20Digital%20Natives,%20Digital%20Immigrants%20-%20Part1.pdf>. Acesso em: 12 jul. 2022.

PSVERSO.COM.BR. **10 melhores jogos de mundo aberto de tiro em terceira pessoa**. 7 fev. 2021. fotografia 1200 x 600. page 6. Disponível em: <https://psverso.com.br/listas/melhores-jogos-mundo-aberto-terceira-pessoa/>. Acesso em: 28 jul. 2022.

RABAÇA, C. A.; BARBOSA, G. G. **Dicionário de Comunicação**. 5. ed. [S. l.]: Campus, 2007. Disponível em: https://www.academia.edu/28561275/Dicion%C3%A1rio_de_Comunica%C3%A7%C3%A3o_5a_Edi%C3%A7%C3%A3o_Carlos_Alberto_Raba%C3%A7a. Acesso em: 7 jul. 2022. ROGERS, Scott. **Level UP**: um Guia Para o Design de Grandes Jogos. São Paulo – SP – Brasil: Edgard Blucher, 01/01/2013. 494 p. v. 1. ISBN 978-85-212-0700-9.

SALEN, Katie; ZIMMERMAN, Eric. **Regras do Jogo**: Fundamentos do design de jogos. 1. ed. São Paulo – SP – Brasil: Blucher, 01/01/2012. 168 p. v. 1. ISBN 978-85-212-0626-2.

SANTOS, Christiano L.; VALE, Frederico S. **Jogos Eletrônicos na Educação**: Um estudo da Proposta dos Jogos Estratégicos. Orientador: Henrique Nou Schneider. 10/10/2006. 110 p. Monografia (Graduação em Ciência da Computação) - Universidade Federal de Sergipe, São Cristóvão -SE, 2006. Disponível em: <https://christianosantos.com/files/pub/monografia.pdf>. Acesso em: 14 jul. 2022.

SANTEE, André. **Programação de Jogos com C ++ e DirectX**. [S. l.]: Novatec, /2005. 400 p. ISBN 85-7522-064-0.

SCHUYTEMA, Paul. **Design de Games**: Uma abordagem prática. 1. ed. [S. l.]: São Paulo-SP: Cengage Learning, 18/04/2008. 472 p. ISBN 8522106150.

SICART, Miguel. Defining Game Mechanics. **The international journal of computer game research**, MIT Press, 2009, v. volume 8, n. 2, p. 1-14, dez. 2008. DOI 1604-7982. Disponível em: <http://gamestudies.org/0802/articles/sicart?ViewType=Print&viewClass=Print>. Acesso em: 9 set. 2022.

STORE.STEAMPOWERED.COM. **God of War**. 14 jan. 2022. fotografia 600 x 337. pag 1. Disponível em: https://store.steampowered.com/app/1593500/God_of_War/. Acesso em: 7 nov. 2022.

STORE.STEAMPOWERED.COM. **Grand Theft Auto V**. 14 abr. 2015. fotografia 600 x 337. Disponível em: https://store.steampowered.com/app/271590/Grand_Theft_Auto_V/. Acesso em: 13 nov. 2022.

STROUGO, Rod; WENDERLICH, Ray. **Learning cocos2D**: A Hands-On Guide to Building iOS Games with Cocos2D, Box2D and Chipmunk. 1. ed. Boston, MA - USA: Addison-Wesley Professional, 07/06/2011. 640 p. ISBN 0321735625.

TECHTUDO.COM.BR. **Doom**: conheça a história do popular jogo de tiro. 1 ago. 2014. fotografia 100 x 604. pag 1. Disponível em: <https://www.techtudo.com.br/noticias/2014/08/doom-conheca-historia-do-popular-jogo-de-tiro.ghtml>. Acesso em: 12 nov. 2022.

TRAINA, A. J. M.; OLIVEIRA, M. C. F. **Apostila de Computação Gráfica**. São Carlos: USP, 17 mar. 2004. Disponível em: http://www.deinf.ufma.br/~paiva/cursos/cg/apost/Apostila_ICMC.pdf. Acesso em: 17 out. 2022.

UBISOFT.COM. **Assassin 's Creed Origins**. 7 jan. 2020. fotografia 960 x 540. Disponível em: <https://www.ubisoft.com/pt-br/game/assassins-creed/origins>. Acesso em: 27 nov. 2022.

UNITY DOCUMENTATION. **Unity Manual**. [S. l.], 25 nov. 2022. Disponível em: <https://docs.unity3d.com/2023.1/Documentation/Manual/AnimationParameters.html>. Acesso em: 3 nov. 2022.

UNITY DOCUMENTATION. **Unity Manual**. [S. l.], 25 nov. 2022. Disponível em: <https://docs.unity3d.com/2023.1/Documentation/Manual/class-Transition.html>. Acesso em: 3 nov. 2022.

UNITY DOCUMENTATION. **Unity Manual**. [S. l.], 25 nov. 2022. Disponível em: <https://docs.unity3d.com/Manual/class-AnimatorController.html>. Acesso em: 4 nov. 2022.

UNITY DOCUMENTATION. **Unity Manual**. [S. l.], 25 nov. 2022. Disponível em: <https://docs.unity3d.com/Manual/class-Prefabs.html>. Acesso em: 4 nov. 2022.

UNITY DOCUMENTATION. **Unity Manual**. [S. l.], 25 nov. 2022. Disponível em: <https://docs.unity3d.com/Manual/class-GameObject.html>. Acesso em: 4 nov. 2022.

GLOSSÁRIO

Avatar: termo usado para se referir ao personagem jogável de um jogo

Gameplay: toda a jornada percorrida em um jogo

Level: fases de evolução da gameplay de um jogo

Sandbox: gênero de jogos conhecidos pelo termo “Mundo Aberto”

Sprites: é um objeto gráfico bidimensional ou tridimensional para jogos

Shaders: conjunto de instruções que definem a superfície de objetos em jogos

Texturas: imagens que servem de superfície de objetos computadorizados

APÊNDICE A

CÓDIGO PARA GERENCIAMENTO DE ANIMAÇÕES EM C#

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AnimationManager : MonoBehaviour {
    Animator anim;

    void Start () {
        animPacusia = GetComponent ();
    }

    void Update () {
        var EstouMovendo = Input.GetAxis ("Vertical");
        if (EstouMovendo != 0) {
            anim.SetBool ("EstouAndando", true);
            anim.SetBool ("EstouParado", false);
        } else {
            anim.SetBool ("EstouAndando", false);
            anim.SetBool ("EstouParado", true);
        }

        if (Input.GetButtonDown ("Fire1")) {
            anim.SetBool ("EstouGoupeando", true);
            anim.SetBool ("EstouParado", false);
        } else {
            anim.SetBool ("EstouGoupeando", false);
        }
    }
}

```

ANEXO A

Termos Adicionais do Mixamo

Válido a partir de 23 de junho de 2021. Substitui todas as versões anteriores.

Estes Termos Adicionais regem o uso que você faz do Mixamo e são incorporados por referência aos Termos de Uso Gerais da Adobe ("Termos Gerais") localizados em www.adobe.com/go/terms_br (estes Termos Adicionais e os Termos Adicionais e os Termos Gerais são coletivamente referidos como "Termos"). Termos com iniciais maiúsculas são definidos no presente documento têm o significado disposto nos Termos Gerais. Referências a "Serviços" nestes Termos Adicionais são pertinentes aos serviços do Mixamo.

1. Restrição a IA/ML. Você não usará os Serviços nem o Software, tampouco permitirá nem instruirá terceiros a usá-los (ou conteúdo, dados, resultados ou outras informações recebidas ou derivadas dos Serviços ou do Software) para, direta ou indiretamente, criar, treinar, testar ou, de outra forma, aperfeiçoar algoritmos de aprendizado de máquina ou sistemas de inteligência artificial, incluindo, entre outros, arquiteturas, modelos ou pesos.

Mixamo Additional Terms pt_BR-20210623