Paulo Henrique Scabeni

APLICAÇÃO DE ANIMAÇÃO EM JOGOS: DESENVOLVIMENTO DE ANIMAÇÃO 2D PARA JOGO ARCADE

Projeto de Conclusão de Curso submetido(a) ao Programa de Graduação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Design.

Orientadora: Profª. Drª. Mônica Stein

Florianópolis
2019

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

Scabeni, Paulo Henrique
Aplicação de animação em jogos: : Desenvolvimento de animação 2D para jogo arcade / Paulo Henrique Scabeni ; orientador, Mônica Stein, 2019.
81 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro de Comunicação e Expressão, Graduação em Design, Florianópolis, 2019.

Inclui referências.

1. Design. 2. Animação . 3. Jogos digitais. 4. Game design. I. Stein, Mônica. II. Universidade Federal de Santa Catarina. Graduação em Design. III. Título.

Paulo Henrique Scabeni

APLICAÇÃO DE ANIMAÇÃO EM JOGOS:
DESENVOLVIMENTO DE ANIMAÇÃO 2D PARA JOGO ARCADE

Este Projeto de Conclusão de Curso (PCC) foi julgado adequado para obtenção do Título de Bacharel em Design e aprovado em sua forma final pelo Curso de Design da Universidade Federal de Santa Catarina.

Florianópolis, 28 de junho de 2019.

Profª. Marília Matos Gonçalves, Dra. Coordenadora do Curso de Design UFSC

Banca Examinadora:

Prof. Clóvis Geyer Pereira, Dr. (Universidade Federal de Santa Catarina)

Prof. Flávio Andaló, Dr. (Universidade Federal de Santa Catarina)

Prof. Rafael Arrivabene, M.Sc.. (Universidade do Vale do Itajaí)

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

AGRADECIMENTOS

O autor expressa aqui seus agradecimentos aos colegas de sala, professores, amigos e familiares. Reconhece também o papel fundamental feito pelo grupo G2E para que o projeto se tornasse viável, assim como a presença da Universidade Federal de Santa Catarina e Breda University of Applied Sciences nesta trajetória.

RESUMO

O trabalho aqui desenvolvido mostra o processo de criação e execução de um protótipo de jogo digital para plataforma de PC, inspirado em jogos *arcade* e no universo *The Rotfather*. O projeto busca evidenciar obstáculos para aplicação de animação em um game, além de soluções para obter uma arte que complemente o jogo sem interferir nas mecânicas do game.

O uso de animações para jogos digitais requer uma ótica diferenciada daquela usada na produção de filmes animados, tendo em vista a presença da interatividade em tempo real com o jogador. Este trabalho utiliza da metodologia *Double Diamond* para produzir um protótipo jogável de jogo digital, aplicando técnicas de animação que permitem uma animação que atue com a interação do jogador.

Palavras-chave: Animação, Jogos *arcade*, *The Rotfather*..

ABSTRACT

The work being developed here shows the process of creation and execution of a videogame prototype for PC platform, inspired by arcade games and The Rotfather universe. The project tries to show obstacles for animation application in digital games, and also brings solutions to get an art which compliments the game without interfere in the game mechanics.

The use of animations for games requires a different point of view if compared to the one used in animated movies production, seen that games have real time interaction with the player. This project utilizes the Double Diamond methodology to produce a playable prototype, applying animation techniques that allow an animation which acts along with the player interaction.

Keywords: Animation. *Arcade* games. *The Rotfather*.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

NPC – Non-playable Character, ou personagem não jogável
HD – High Definition, ou alta definição
NES – Nintendo Entertainment System

SUMÁRIO

INTRODUÇÃO

O uso da animação já não se limita mais apenas a filmes, visto que podemos encontrar sua aplicação para melhoria de experiência de usuário em menus de interface e até mesmo complemento a músicos em um concerto ao vivo. Em filmes, é cada vez mais comum que vejamos retoques ou cenas inteiras compostas por animação, na medida que a tecnologia ao nosso alcance permite níveis de realismo admiráveis.

Uma indústria que teve um crescimento notável nos últimos anos foi a dos videogames, já que sua popularidade ganha cada vez mais peso. Tornou-se perceptível a evolução das animações em jogos digitais, tendo exemplos desde os títulos que buscam trazer realismo visual, como *Uncharted 4* (*Naughty Dog*, 2016), até jogos que buscam reviver a sensação de assistir um desenho animado da década de 30 como *Cuphead* (*StudioMDHR*, 2017). Os jogos digitais começaram com uso restrito de técnicas de animação não somente por conta da forte limitação que a tecnologia da época dispunha, mas também por falta de experiência dos profissionais envolvidos na produção do jogo com relação a animação em games.

Nos primórdios do uso doméstico de videogames, com consoles como Atari 2600 por exemplo, muitos títulos tinham tanto o setor de arte quanto programação e *game design* produzidos pela mesma pessoa. Em muitos casos, o setor da arte era o que ganhava menos profissionais especializados, uma vez que a produção para estes consoles exigia um conhecimento de programação mínimo para conseguir entregar algo suportado pelo hardware.

O autor do título *Adventure*, para *Atari 2600*, relata que "cada jogo era feito por uma pessoa - eles escreviam o código, testavam, depuravam, e decidiam quando estava pronto"(WAWRO, 2015, apud ROBINETT, 2015, tradução nossa). O jogo então foi feito com Warren Robinett encarregado de produzir tudo relacionado ao título, deixando a produção da arte do jogo como baixa prioridade não só pela limitação de hardware, mas também pela curta experiência que a indústria tinha na época.

Com o tempo, animação em games deixou de ser algo secundário para atuar como forma de construir uma identidade e completar a experiência do jogador. O projeto a ser desenvolvido busca criar um protótipo jogável de um game que se passa dentro do universo *The*

Rotfather, com objetivo de aplicar soluções para a animação de personagens e cenários de maneira que estas respeitem as regras estabelecidas pelo segmento de *game design* do jogo.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver o protótipo de um jogo 2D ao estilo arcade inspirado no universo *The Rotfather*, buscando mostrar e aplicar técnicas de implementação de animação em conjunto com *game design* e programação.

1.1.2 Objetivos Específicos

- Analisar a aplicação de animações que complementam e melhoram a imersão em jogos digitais;
- Elaborar um *game design* simples e um *core game loop* para jogo 2D que suportem a aplicação de animações comumente usadas;
- Desenvolver *sprites* e animações em pixel art para assets a serem usados no protótipo jogável;
- Elaborar e utilizar formas que o uso de programação possa incrementar a qualidade da animação nos jogos;

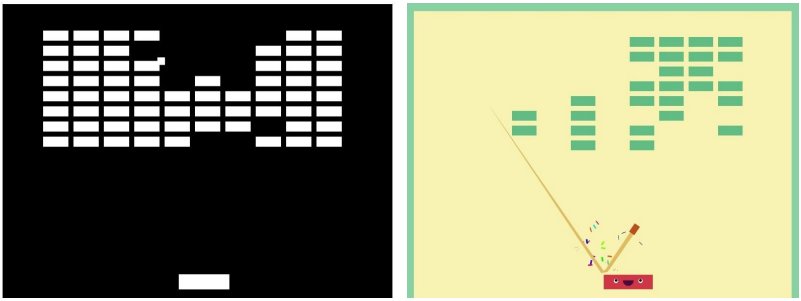
1.2 JUSTIFICATIVA

Animação aplicada em filmes animados com o objetivo de imitar a vida traz uma experiência única para o espectador há décadas, contando histórias e trazendo sentimentos de forma única. Apesar de seu começo simples, a animação desenvolveu uma forma de envolver o público, e aos poucos seu uso se solidificou cada vez mais em meio a outras mídias.

Filmes live action muitas vezes recebem retoques ou cenas compostas totalmente por animação. No título Blade Runner 2049, de 2017, dirigido por Denis Villeneuve, a personagem Rachel pode facilmente não ser reconhecida como uma animação computadorizada, visto sua fidelidade com anatomia e comportamento humano. Não somente como forma de imitar a realidade a animação se desenvolveu: o longa Homem-Aranha: no Aranhaverso, de 2018, produz a animação com uma linguagem completamente diferente de Rachel em Blade Runner. Personagens são mais elásticos, cores distorcem e reagem à iluminação de forma diferenciada, e até mesmo a taxa de quadros se altera propositalmente conforme a cena. Ambos os títulos produzem animação com um resultado estético completamente diferente, e ainda assim sua aplicação foi tão bem planejada e executada de acordo com sua proposta que receberam muito destaque positivo tanto pela crítica quanto pelo público.

No universo dos jogos digitais, a animação aplicada em personagens, cenários e até mesmo menus vem ganhando cada vez mais atenção e cuidado para que possam trazer imersão ao jogador. Se compararmos o título Space Invaders lançado em 1978 com o game Space Invaders Extreme de 2008, fica claro como a adição de partículas de efeito animadas e outros efeitos secundários trazem uma vida nova ao jogo. Martin Jonasson e Petri Purho (2012) fazem um excelente trabalho em sua palestra “Juice it or Lose it” ao criarem um título semelhante a Breakout de Atari 2600, desta vez com a adição de animações e respostas visuais ao jogador sem que as mecânicas do jogo sejam alteradas. O ganho de experiência é claro quando comparamos a versão sem animações do jogo com o resultado final, mostrando como é possível conquistar uma experiência melhorada em um jogo apenas aplicando animações que adicionem ao *gameplay*.

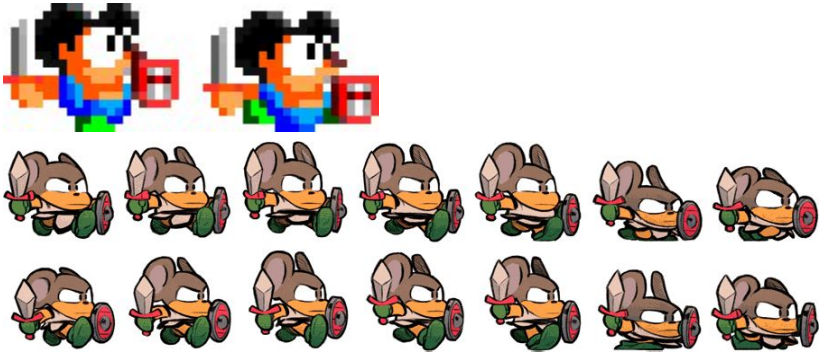
Figura 1 *- Comparação entre configurações do jogo Juicy-Breakout*



Fonte: screenshots retirados pelo autor

Para alguns jogos, a atenção dada à produção de suas animações acaba até mesmo se tornando um motivo de compra para os jogadores. O remake feito de *Wonder Boy* 3 (original de 1987) chamado *Wonder Boy*: *The Dragon's Trap*, lançado em 2017, chamou muita atenção por seus gráficos atualizados que substituem animações simples por *frames* totalmente retrabalhados em HD. Além de muito mais *frames* por animação, o título traz movimentos secundários às animações, como o balanço da cabeça dos personagens que adiciona peso em seu ciclo de corrida, ou pequenas rotações no ângulo do rosto que trazem volume aos personagens, tudo isso mantendo as mecânicas originais intactas.

Figura 2 *- Animações de caminhada dos jogos Wonder Boy III e Wonder Boy; The Dragon's Trap*



Fonte: http://i47.tinypic.com/33ojlog.png
http://www.lizardcube.com/presskit/TheDragonsTrap/

Tendo em vista estes aspectos, o desenvolvimento do protótipo jogável neste projeto de conclusão de curso busca então planejar e aplicar soluções que mostrem uma adição positiva à experiência de jogar em um game digital, podendo usar animação como uma forma de enriquecer a experiência imersiva.

1.3 METODOLOGIA

A metodologia escolhida pelo autor para ser aplicada no projeto é a metodologia *Double Diamond*, desenvolvida pelo IDEO e utilizada pelo *Design Council*. Esta metodologia consiste em 4 etapas:

● *Discover*

É a primeira etapa de expansão, que busca ampliar horizontes e arrecadar conhecimento e ideias que são relacionados ao projeto. Nesta etapa é feito o mínimo possível de cortes, visto que o objetivo é obter quantidade, e não necessariamente qualidade.

● *Refine*

Busca filtrar todo o material provindo da etapa anterior. Ela reduz a quantidade de caminhos a serem seguidos, visando qualidade acima de quantidade para que o escopo de ideias e opções limitem-se àquilo que se mostra viável. Ao fim desta etapa, chega-se à definição do problema em questão.

● *Develop*

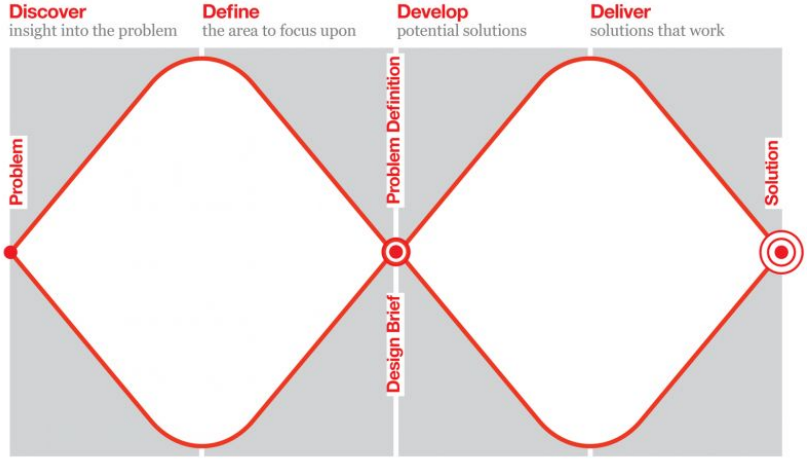
Segunda etapa de expansão, desta vez aplicada em cima do resultado da etapa de *Refine* e da definição do problema. Aqui geralmente ocorre a prototipação, teste de ideias e tentativas de novas ideias para solucionar o problema.

● *Delivery*

Consiste na finalização e polimento do que foi desenvolvido anteriormente para que o produto ou serviço possa ser lançado.

A metodologia double diamond foi escolhida por se mostrar eficiente para solução de problemas tanto de escala grande quanto pequena, podendo descrever o mapa de desenvolvimento do projeto como um todo assim como a de obstáculos mais pontuais a serem vistos adiante.

Figura 3 *- Representação gráfico da metodologia Double Diamond*



Fonte:
https://www.designcouncil.org.uk/news-opinion/design-process-what-double-diamond

1.4 DELIMITAÇÕES

O protótipo jogável a ser desenvolvido será 2D, com estilo visual em pixel art para a plataforma de PC. Será feito o desenvolvimento da base de mecânicas do personagem a ser controlado pelo jogador, sua interação com objetos e atividade de inteligência artificial básica para inimigos, tal como pelo menos um espaço virtual onde estes sejam aplicados. Não serão explorados design de menus e nem interfaces de usuário.

A programação a ser explorada neste projeto é considerada simples, buscando evidenciar um pensamento lógico que aprimore o jogo, e não o uso linhas de código específicas.

O *game design* a ser desenvolvido busca ser minimalista, atuando como contexto para visualização da aplicação das animações feitas, visto que o foco principal deste projeto é o uso e aplicação de animação. Assim, o segmento de *game design* terá um papel de suporte às produções visuais no contexto deste trabalho.

A parte de sonorização não será estudada neste projeto, posto que ela merece uma dedicação a parte e é mais distante do que foi estudado no curso. Todo conteúdo sonoro gerado para este projeto visa apenas complementar a experiência de jogá-lo, atuando de forma coadjuvante nesta etapa de desenvolvimento.

2 DESENVOLVIMENTO TEÓRICO

Para melhor diferenciação das áreas trabalhadas neste projeto, o segmento de desenvolvimento teórico foi dividido entre *Game Design* e Animação, dado que as duas áreas devem ser tratadas de acordo com suas peculiaridades.

2.1 GAME DESIGN

2.1.1 O que é Game Design?

Programação e arte dificilmente compõem a experiência de um jogo por si só. Junto com elas é necessário algo que traga unidade e dê uma meta para suas aplicações, com o objetivo de entreter, emocionar e contar histórias ao jogador. Assim, *game design* busca explorar como será a experiência do usuário, fazendo parte da concepção de tema, mecânicas e regras que compõem o jogo para um nível prático.

Apesar de ser um conceito simples, suas demandas são grandes e, devido à enorme diversidade de gêneros de jogos, é muito difícil estipular o que é considerado um *game design* bem feito. Richard Rouse III (2001), traz:

> Quais são os elementos de *game design* que fazem um jogo realmente bom? Claro, não há resposta definitiva para tal pergunta. Não obstante, como um *game designer* será esperado que você intuitivamente saiba exatamente qual é a resposta. Compreender *game design*, assim como qualquer forma de arte, é em grande parte uma compreensão internalizada, uma reação instintiva, uma "sensação" que você possa ter. Talvez você não será capaz de formar esta resposta com palavras, mas você terá que compreender quais aspectos de *game design* são fortes ou fracos, e como estes podem ser substituídos por aqueles. Experiência tem um grande papel em compreender o que faz um jogo divertido, sendo ela como *game designer* tanto quanto jogador.(ROUSE III, 2001, tradução nossa)

O título *Space Invaders*, por exemplo, acabou trazendo um resultado de *game design* não intencional por conta de limitações do

hardware da época. O jogo dá ao jogador o controle de uma nave que fica na base da tela, podendo mover-se para esquerda e direita, além de atirar para cima. Ao mesmo tempo, pequenos alienígenas surgem do topo da cena enquanto descem lentamente, ficando cada vez mais próximos do jogador. Como descrito por Kaleb Eberhart:

> Uma falha no microprocessador fazia com que a animação dos inimigos aumentasse a velocidade conforme menos deles eram renderizados . Enquanto criar essa curva de dificuldade foi um acidente por conta de problemas técnicos, esta característica foi teve sucesso em criar um desafio otimizado visto que *Space Invaders* vendeu mais de 300,000 máquinas no Japão e mais 60,000 na América por volta de 1980 .(EBERHART, 2019, tradução nossa)

Assim uma curva de dificuldade que geralmente é planejada por *game designers* foi resultado de uma limitação de processamento da plataforma que executava o jogo.

Em alguns títulos, a escolha daquilo que comumente seria considerado um *game design* ruim se torna parte do diferencial do jogo. *I Wanna Be the Guy* (Michael "Kayin" O'Reilly, 2007) e *Kaizo Mario* (T. Takemoto, 2007), por exemplo, são títulos de plataforma que colocam o jogador em situações de derrota sem aviso prévio, muitas vezes usando elementos iguais que não seguem mesmo comportamento. Por outro lado temos *Downwell*, que é exemplar por ter um *game design* que busca trazer múltiplos propósitos para suas mecânicas, como abordado por Mark Brown em seu vídeo "*Downwell's Dual Purpose Design | Game Maker's Toolkit*". O jogo usa, por exemplo, a mecânica de atirar não somente para eliminar inimigos, mas também para o jogador usar o coice dos tiros como forma de manobrar seu personagem durante uma queda livre.

O estudo de *game design* vem se expandindo cada vez mais, ainda que continue sendo algo abstrato para muitos e, como dito por Richard Rouse III (2001), entender *game design* é uma forma de arte, e que com certeza é fortemente influenciado pela experiência vivida pelo designer.

2.1.2 Core Gameplay Loop e mecânicas

Segundo Josh Bycer, "o *core* gameplay *loop* é o sistema do jogo primário ou mecânica que define seu título. É o que você constrói seu jogo inteiro em volta, e geralmente é o qualificador que determina a qual gênero seu jogo pertence."(BYCER, 2019, tradução nossa). Com isso, podemos enxergar o core *gameplay loop* como parte da espinha dorsal de um jogo, ramificando-se para criar as diversas experiências que o game pode trazer baseado em contexto, obstáculos e outros elementos.

Desde os primeiros títulos de jogos digitais desenvolvidos até lançamentos atuais podemos ver uma enorme evolução na quantidade e detalhamento de mecânicas a disposição do jogador. No título clássico *Pac-Man*, lançado em 1980, o jogador contava com a habilidade de se mover em 4 direções enquanto busca consumir todos os pontos espalhados em um labirinto para passar ao próximo nível, já em games como *The Elder Scrolls V: Skyrim* o jogador busca o aperfeiçoamento do seu personagem enquanto pode optar por equipar suas melhores armas para matar dragões ou buscar por plantas específicas para criar poções e subir de nível por conta da sua perícia em alquimia.

Figura 4 *- Core Gameplay Loop de Pac-Man*

Fonte: Desenvolvido pelo autor

Figura 5 *- Core Gameplay Loop de The Elder Scrolls V: Skyrim*

Fonte: Desenvolvido pelo autor

Maior variedade de mecânicas não necessariamente implica em um jogo melhor ou mais divertido. O aclamado título *Super Hot* lançado em 2016, por exemplo, conta com um *core loop* que consiste em esperar, mover-se, eliminar inimigos e passar de fase. As mecânicas do jogo complementam este *loop* com a habilidade de diminuir a velocidade do tempo enquanto o jogador está parado, permitindo um maior planejamento de suas ações futuras.

Games desenvolvidos para *arcade* geralmente trazem uma mecânica principal de forma bem nítida, que acaba por se tornar o elemento que continua atraindo jogadores e muitas vezes tenta ser algo que continue divertido mesmo com repetição. *Super Crate Box*, lançado em 2010 traz um core loop bem reduzido baseado em mover-se, eliminar inimigos e coletar caixas, com pouca variedade de ações que o jogador pode fazer. Seu ritmo bem cadenciado e controles bem executados tornam o jogo pouco enjoativo.

Assim, podemos ver como o *core game loop* dita a direção da experiência que o jogadores vai ter, assim como seu tipo de público alvo.

2.2 ANIMAÇÃO

2.2.1 Histórico da animação

Há muito tempo o ser humano mostra interesse em tentar contar histórias. Muito antes daquilo que conhecemos como escrita atual já buscávamos retratar acontecimentos com imagens que traziam uma narrativa e tentavam imitar a vida. Os autores da obra *Illusion of Life* trazem:

> Vinte e cinco mil anos atrás, nas cavernas do sudoeste europeu, o homem de Cro-Magnon fez desenhos ilustres dos animais que ele caçou. Suas representações não são somente precisas e desenhadas de maneira bela, mas também muitas parecem dar uma vida combinada com sugestão de movimento. (THOMAS, FRANK, Illusion of life, pg 13, 1995, tradução nossa)

Assim, vemos que nossa curiosidade em buscar formas de retratar movimento a partir de imagens estáticas não é algo recente.

Limitados pela tecnologia que os acompanhava em suas épocas, "artistas continuaram a pesquisar por um meio de expressão que iria permití-los capturar aquela faísca de vida ilusória, e no fim dos anos 1800 novas invenções pareceram finalmente tornar isto possível". Objetos como o zootrópio (ou "roda da vida") tentaram imitar movimento a partir de trocas rápidas de imagens em um local de foco. Mais tarde, tivemos a chegada do folioscópio, ou *flipper book*, se se popularizou devido à sua simplicidade de uso: um simples bloco de folhas com vários desenhos que quando folheados rapidamente parecem ganhar vida.

Segundo Richard Williams (2016, pg 15), o antecessor daquilo que conhecemos como desenho animado foi o filme animado produzido por Thomas Edison e James Stuart Blackton, publicado em 1906 sob o título de "Fases cômicas de faces engraçadas". Por mais primitiva que a animação fosse, ela abriu portas para futuras produções como Gertie, o Dinossauro e Gato Félix.

2.2.2 Os 12 princípios da animação

Quando olhamos para as primeiras animações produzidas e as comparamos com o material comercial que é feito na atualidade, fica nítida a diferença de fluidez de movimentos. Isso se deve em grande parte ao estudo e divulgação dos 12 princípios da animação estipulados por Ollie Johnston e Frank Thomas, que propuseram uma lista de conceitos que deveriam ser aplicados de maneira que viriam a aprimorar a qualidade da animação sendo produzida. Como trazido por um dos autores:

> Os animadores continuaram a pesquisar por métodos melhores de relacionar desenhos uns com

> os outros e encontraram algumas formas que pareciam trazer resultados previsíveis. Eles não podiam esperar por sucesso todas as vezes, porém estas técnicas especiais para desenhar um personagem em movimento ofereciam alguma segurança (THOMAS e FRANK, 1981, pg 47)

Estes princípios mencionados são:

- *Squash* e *stretch*: deformações na estrutura do personagem, amassando-o ou esticando-o de acordo com a ação;
- Antecipação: o movimento anterior à ação principal que busca trazer mais peso e impacto a esta;
- *Staging*: clareza na representação visual, sendo com poses dos personagens, seu posicionamento ou composição da cena;
- *Straight ahead* e *pose-to-pose*: relativo à técnica de animação usada, sendo cada quadro desenhado em ordem cronológica ou a execução dos quadros principais e depois os quadros intermediários;
- *Follow through* e *overlapping action*: buscam representar a diferença de movimentação entre duas partes conectadas, alterando o tempo e a natureza do seu tempo de acordo com seu peso e forma
- *Ease in* e *ease out*: aceleração e desaceleração nos movimentos, que pode trazer mais fluidez no conjunto de ações;
- Arcos: movimentos em forma de arco, que de maneira geral deixam as ações dos personagens mais orgânicas;
- Ação secundária: movimentos menores e paralelos a ações principais que ajudam a reforçar o peso destas;
- *Timing*: velocidade em que a ação ocorre, alterando assim a percepção de peso do objeto ou personagem;
- Exagero: extrapolação nas ações da atuação para deixar a cena mais impactante, muitas vezes fugindo de parâmetros realistas;
- *Solid drawing*: referente à qualidade do desenho, que busca trazer clareza e definição nas formas daquilo desenhado;
- *Appeal*: diz respeito à clareza do design daquilo sendo representado, com fim de deixar a leitura mais clara ao espectador.

2.2.3 Animação em jogos

A produção de animação para jogos, contudo, traz o fator da interatividade do jogador durante o processo de exibição das animações. Dessa forma, alguns dos princípios de animação, como a antecipação, não conseguem ser aplicados da mesma maneira que em séries e filmes. Segundo Jonathan Cooper, "entender esses básicos da animação é essencial, então é hora de revisitá-los sob a perspectiva de animação para *videogame*"(COOPER, 2019, tradução nossa). Cooper traz em seu texto vários exemplos que mostram como a animação pode ser aplicada em jogos de maneira que os complemente, também descrevendo limitações que essa interatividade traz. Segundo este:

> Antecipação é um tópico controverso em video games, com o designer frequentemente pedindo o mínimo possível (de antecipação) e animadores forçando para o máximo de *frames* possível. Pouca antecipação e o movimento desejado, como um soco ou balanço de espada, terá pouco peso (um componente chave para o feedback do jogador, não somente estético). Antecipação demais e o movimento irá parecer irresponsivo, removendo agência do jogador e reduzindo o sentimento de controlar o avatar diretamente.(COOPER, 2019, p. 01, tradução nossa)

Não somente na aplicação de animações para o jogador é necessário pensar em como o *game design* se conecta com a animação. Cooper ainda traz como antecipação para ações e ataques de NPCs (ou personagens não controlados pelo jogador) é intencionalmente mais longo para que o jogador possa ler a antecipação e usá-la para planejar suas ações. Como escrito pelo autor, "Não há muita diversão em ter que adivinhar o que um inimigo pode fazer com pouco aviso."

É possível ver no jogo *Wonder Boy: The Dragon's Trap* como os desenvolvedores tiveram o cuidado de adicionar animações de antecipação para ataques de alguns inimigos específicos que permitiam esta adição sem interferir nas mecânicas do game original que inspirou este remake. A possibilidade de ler as ações futuras dos inimigos, ou também conhecido como telegrafar, acaba por trazer ao jogador a

sensação de responsabilidade quando é acertado pelo inimigo e reduzindo assim a sensação de injustiça e frustração se comparado ao dano levado por um golpe inesperado.

Figura 6 *- Animação de ataque do mesmo personagem em Wonder Boy III e Wonder Boy: The Dragon's Trap*

Fonte: Screenshots retirados dos respectivos jogos

Cooper ainda sugere como a animação pode realçar outras ações como o pulo. Segundo o autor, "um personagem pulando para cima pode ser esticado verticalmente durante a porção rápida do pulo para acentuar o vertical, mas pode amassar no ápice do arco de pulo e novamente no impacto com o chão"(COOPER, 2019, tradução nossa). Veremos adiante a aplicação deste tipo de conceito sendo feita no protótipo do jogo a ser criado.

Mariel Cartwright, artista animadora do jogo *Skullgirls*(*reverge Labs*, 2012), traz outros problemas e soluções em sua palestra do evento GDC chamada "*Fluid and Powerful Animations Within Frame Restrictions*". A palestrante apresenta como a interatividade do jogador influencia a maneira que animadores devem pensar em suas animações. Por *Skullgirls* (o jogo usado como exemplo na palestra) se tratar de um jogo de luta, o tempo de cada animação tem grande papel no

balanceamento do jogo, visto que isso cria vantagens e desvantagens entre personagens de acordo com seu tempo de ação. Dentre os pontos levantados por Mariel, vemos como o uso de *smears*, ou borrões de movimento, é eficiente para cobrir a necessidade de animação para ações que sejam muito rápidas, algo que será de grande utilidade no projeto a ser desenvolvido.

> Para manter a responsividade, o animador deve ser capaz de controlar quando o jogador está apto para executar uma ação consecutiva a partir da especificação de um quadro onde o jogador ganha controle novamente antes do fim, permitindo também que o *follow-through* seja executado completamente se não houver nenhum comando do jogador ao invés de cortá-lo na animação em si. *Game engines* que não incorporam tal recurso forçam os animadores a terminar suas ações antes do desejado para manter a sensação de resposta, perdendo uma ferramenta chave para animadores de jogos a fim de trazer tanto boa sensação quanto boa aparência aos personagens. (COOPER, 2019, p. 01, tradução nossa)

Podemos ver que a animação para jogos digitais conta com o obstáculo de lidar com a interatividade em tempo real. Em filmes animados, o tempo de cada ação é elaborado em etapas anteriores à arte final, podendo haver adaptações que permitam uma leitura clara de tudo que deve ser contado. Para a produção de muitos jogos, é necessário pensar na animação de forma diferenciada, levando em consideração todas as interrupções que possam ser feitas pelo jogador.

3 O UNIVERSO *THE ROTFATHER*

O projeto a ser trabalhado pelo autor é baseado no universo do projeto *The Rotfather*, desenvolvido pelo G2E, ou Grupo de Educação e Entretenimento. O grupo multidisciplinar faz pesquisa e extensão dentro da Universidade Federal de Santa Catarina na área de entretenimento transmídia.

3.1 O QUE É O UNIVERSO *THE ROTFATHER*?

A ficção trazida no projeto transmídia *The Rotfather* é baseada nas relações e tramas vividas por personagens antropomórficos das mais diversas espécies como aranhas, baratas, ratos e sapos. Grande parte do conteúdo desenvolvido se passa na cidade fictícia de Faux City, situada nos esgotos de Nova York na década de 1940. Um dos segmentos principais de história do projeto trata do tráfico de açúcar, visto nesta sociedade como um equivalente a drogas como cocaína, e como o império do personagem Al Kane e todos envolvidos se desenvolve conforme suas respectivas ambições e poderes.

O projeto busca contar suas histórias não somente a partir de jogos digitais, mas também jogos analógicos como *Ostentação*, *Intrigas* e *Contrabando*, assim como livros, quadrinhos e séries de animação. Apesar de pouco focado em narrativa, o projeto do game desenvolvido pelo autor usa como base personagens e localizações do universo *The Rotfather*, de forma que alguns momentos da linha do tempo ou leis de funcionamento originais do projeto acabam nem sempre sendo aplicados para favorecer a experiência de *gameplay arcade* buscada neste jogo.

3.2 ESCOLHA DO TEMA E DOS PERSONAGENS

Dentre as espécies já relatadas no universo *The Rotfather* estão os sapos, que têm sua origem no Japão e são trazidos a Nova York pelo sapo Eiji em seu arco de história. A comunidade dos sapos, de forma geral, traz elementos e estéticas comumente vistos na cultura oriental, assim como personagens pertencentes à Yakuza.

O personagem controlado pelo jogador nesta versão de demonstração do jogo a ser desenvolvida é Takeda, um sapo membro da Yakuza. A escolha por este personagem vem por conta da mecânica

principal de combate abordada no jogo a ser feito: o *dash*. Sendo que o "*dash*" é uma movimentação rápida em um curto espaço de tempo em *games*, e a ideia de combate para o projeto em desenvolvimento é que seja feita com arma branca, foi feita a busca de um personagem ágil que se encaixasse nesses parâmetros. Porque dentro do clã dos sapos é possível encontrar indivíduos que lutem de forma semelhante a ninjas, este tipo de personagem se mostrou a melhor opção para aplicação do *dash*. Takeda, por ser um dos personagens com mais presença dentre os sapos, foi escolhido para que o público que já tivesse contato prévio com o universo *The Rotfather* tivesse mais chance de reconhecer e identificar o personagem.

Para os inimigos, foram escolhidas as baratas por serem visualmente contrastante com Takeda, tanto em cor quanto em formas. As baratas apresentadas no jogo em desenvolvimento acabaram por sofrer a adaptação de poder voar livremente por conta de objetivos mecânicos, enquanto as baratas do universo *The Rotfather* têm apenas a habilidade de planar com suas asas.

Como cenário, foi escolhido o tema de esgoto muito presente no *The Rotfather*, desta vez trazendo bastante peso em cores azuis e preto, que serve não somente para reforçar a atmosfera do lugar, mas também realçar o contraste com as cores dos personagens.

***Figura 7** - Splash art para o jogo em desenvolvimento*

Fonte: Desenvolvido pelo autor

4 DESENVOLVIMENTO

A etapa de desenvolvimento deste projeto, assim como em embasamento teórico, foi dividida entre *Game Design* e Arte. O segmento de arte, por sua vez, engloba não somente as animações, mas também a direção de arte feita para o projeto.

4.1 GAME DESIGN

4.1.1 Proposta para o projeto e pesquisa de referências

Inicialmente, a proposta para *game design* do jogo a ser desenvolvido tinha como base um *gameplay* com foco em ação, podendo abrir oportunidade para a exploração do impacto que as animações trazem ao jogo de maneira direta. Além disso, outro fator importante é que o jogo não viesse a exigir muito tempo do jogador para que este pudesse se divertir cada vez que jogasse, algo comum em jogos *arcade*. De forma ideal, uma partida do game a ser desenvolvido poderia conter até mesmo somente 5 minutos, e se expandindo em tempo conforme o jogador progride.

Para a definição do gênero do jogo a ser desenvolvido, fez-se a primeira etapa de expansão presente na metodologia *double diamond*, expondo alguns subgêneros populares nomeados com base na definição de Scott Rogers em *Level Up*. Baseados no gênero de ação, os subgêneros que se destacaram dentre as descrições de Rogers foram aventura de ação, ação *arcade* e *beat'em up*.

Segundo Rogers, o sub-gênero de aventura de ação é uma combinação de gêneros que se baseia em coleção de itens e resolução de quebra-cabeças. Assim, o estilo de *gameplay* se mostra pouco propício para exploração de ação e velocidade, evidenciado por títulos que se encaixam neste gênero como *Tomb Raider* (Crystal Dynamics, 2013) e *Assassin's Creed* (Ubisoft, 2007) trazem uma linearidade e evolução de *gameplay* não condizentes com a experiência buscada no projeto a ser feito, além de um escopo de produção muito grande se comparado com a meta a ser atingida.

Jogos do sub-gênero *beat'em up* "têm jogadores lutando contra ondas e mais ondas de inimigos com aumento da dificuldade" (ROGERS, 2014, pg. 34), deixando assim uma oportunidade de um *gameplay* não tão linear. O sub-gênero não se mostrou como melhor

escolha porque jogos *beat'em up* populares como *Final Fight* (Capcom, 1989) e *Scott Pilgrim vs. The World: The Game* (Ubisoft, 2010) trazem um progresso baseado em diferentes fases, geralmente acompanhados de uma progressão de história, algo que o autor quer evitar para o título em desenvolvimento.

O gênero *arcade* foi definido por Rogers como jogo com "ênfase em *gameplay* de reflexos, pontuação e tempos curtos de jogo"(ROGERS, 2014), mostrando-se o melhor candidato para ser levado à próxima etapa de expansão do desenvolvimento inicial do jogo. Como descrito por Rogers, " gêneros de jogos aparecem em todos os tamanhos e formas. Não tenha medo de misturar e combinar"(ROGERS, 2014). Logo, o game a ser produzido terá também alguns elementos que foram classificados como *beat'em up* como o combate com ondas de inimigos para melhor aproveitamento do jogo.

Games que nasceram na era dos *arcades*, por volta dos anos 80, traziam consigo um grande foco na repetição do uso de uma mecânica, onde o jogador deveria utilizá-la em diferentes contextos e níveis de desafio para que a experiência do jogo continuasse prazerosa e demorasse a tornar-se repetitiva. Muitos títulos foram desenvolvidos para arrecadar dinheiro nas máquinas de *arcade*, em consequência de em cada falha do jogador ser cobrado um valor monetário para que pudesse jogar novamente. Alguns títulos citados pelo autor são *Dig Dug* e *Dinner Dash*, que mostram justamente um *core loop gameplay* simples e curto, porém que se mantém divertido mesmo após várias jogatinas. Como forma de expandir referências e ideias para o desenvolvimento do jogo, deu-se início à pesquisa de referência de títulos que pudessem agregar com mecânicas no game a ser feito.

Em 1983 foi lançado o jogo *Mario Bros*. para *arcade*, publicado pela Nintendo. O título buscava trazer os irmãos Mario e Luigi, com um *gameplay* que consistia em controlar os personagens usando movimentação lateral e pulo para desabilitar e eliminar inimigos que surgiam em pontos específicos do cenário. Cada fase era concluída assim que um número específico de inimigos era eliminado, resultando em uma pontuação que segue o desempenho do jogador.

A partir da inspiração em cima da simplicidade de *Mario Bros*., deu-se início à segunda etapa de expansão dentro do modelo *double diamond*. Nesta etapa, buscou-se avaliar mecânicas e loops de *gameplay* que poderiam fazer parte do projeto a ser desenvolvido, com o conceito

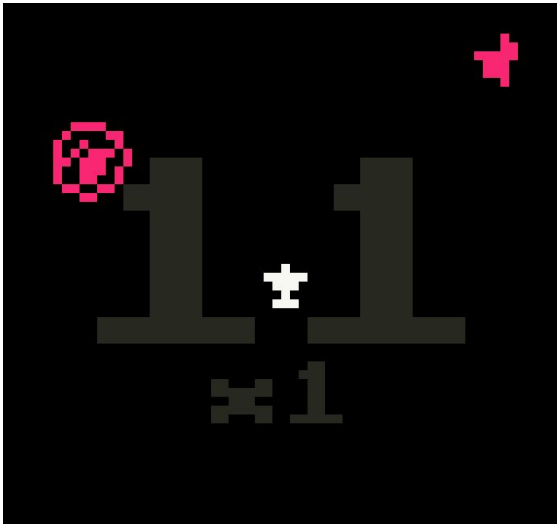
de jogabilidade rápida e repetição de mecânica em mente. O título que se destacou em primeiro momento foi *Super Crate Box*, da desenvolvedora holandesa *Vlambeer*. O jogo se trata de uma ramificação do conceito apresentado em *Mario Bros*.: o jogador controla um personagem com movimentação lateral e pulo, desta vez com o acréscimo de uma arma de fogo em sua mão, e por conta disso, a forma de derrotar os inimigos se torna simplesmente acertá-los com os tiros. O diferencial deste título está no fato que a pontuação do jogador é feita conforme ele coleta caixas que ressurgem em locais aleatórios do cenário cada vez que são coletadas. Sempre que coletadas, as caixas trocam a arma atual do jogador por outra arma aleatória, trazendo assim uma constante variação na forma de jogar e obrigando o jogador a se deslocar não somente para eliminar os inimigos, mas também coletar caixas e pontuar em fases que não possuem final. A partida chega a um fim somente quando um inimigo encosta no personagem do jogador, fazendo assim que ele morra e a pontuação do jogador seja validade para desbloquear novas fases, armas e personagens.

O segundo título que ganhou destaque na pesquisa foi *Dash Craft*, feito em 2017 pelo desenvolvedor TK. Este game traz um conceito minimalista de gameplay, onde a partir de uma visão aérea, o jogador controla um avião que segue a posição do mouse. Toda vez que o jogador aciona o botão de ação, o avião se desloca com grande impulso àquela direção por um curto período de tempo (conhecido também como *dash*). Aviões inimigos surgem pela borda da tela, e são destruídos se acertados pelo jogador durante seu *dash*, fazendo o jogador pontuar. Inimigos mais avançados começam a efetuar o mesmo tipo de ação com o impulso, podendo assim acertar o jogador e resultar num fim de jogo. A busca do jogador é puramente por pontuação, e a velocidade constante do avião do jogador contribui para que ele esteja atento e responsivo ao jogo.

Figura 8 *- Jogo Super Crate Box*

Fonte: https://store.steampowered.com/app/212800/Super_Crate_Box/

Figura 9 *- Jogo Dash Craft*



Fonte: https://itch.io/jam/lowrezjam2017/rate/168082

Spelunky é um jogo que se distancia do gênero *arcade* clássico e bebe da fonte dos jogos considerados "*roguelite*". Jogos *roguelite* são uma variação do estilo *roguelike*, que consiste em jogos baseados no título *Rogue* de 1980. Algumas características do estilo são: morte permanente, fazendo que o jogador perca o progresso de evolução com seu personagem ao morrer; níveis gerados proceduralmente e itens gerados com certa aleatoriedade (KING, 2015).

O título *Spelunky*, desenvolvido por Derek Yu, ganhou destaque na cena independente de jogos por trazer cenários procedurais que mudavam cada vez que o jogador começasse a partida, mantendo-se longe da repetição e exigindo que o jogador se adaptasse ao novo cenário. Este elemento de fases geradas de maneira diferente a cada partida se mostrou então algo muito promissor para manter partidas variadas que se tornassem desinteressantes. A aplicação de fases geradas de maneira procedural podem aumentar a vida útil do jogo, porém o custo de tempo para planejamento, execução e balanceamento deste tipo de sistema se mostra muito alto.

Figura 10 *- Jogo Spelunky*

Fonte: https://www.pcgamer.com/spelunky-review/

Em 2008, foi lançado o título *Call of Duty: World at War*, devolvido pela Treyarch. O título first person shooter traz consigo um

modo extra onde o jogador tem que enfrentar hordas de zumbis que nascem em um ritmo cada vez mais acelerado conforme o jogador sobrevive, tendo fim apenas quando o personagem do jogador morre. Com um cenário limitado, os inimigos surgem de locais específicos no ambiente como janelas e portas, que podem ser reforçadas por barricadas feitas pelo jogador para retardar o ritmo de surgimento dos zumbis. Este modo de jogo teve retorno em títulos seguintes da série *Call of Duty: Black Ops*, trazendo as mesmas mecânicas clássicas com acréscimos de interatividade do jogador com armadilhas e objetos de cenário que poderiam ajudar durante a tarefas de manter-se vivo em meio às hordas. O formato de jogo também recebeu versões em outros títulos como *Gears of War 2*, que possui o modo *Horde*, onde o jogador passa pela mesma tarefa de sobreviver a ondas de inimigos em um cenário limitado e interativo tanto para defesa quanto para ataque.

Figura 11 *- Modo Zombies do jogo Call of Duty: World at War*

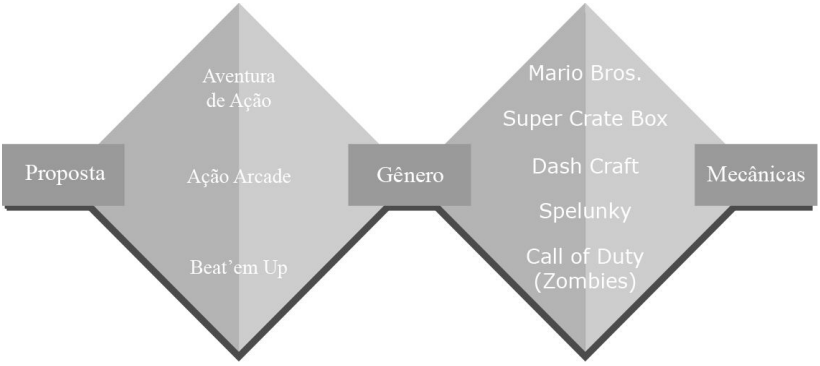
Fonte: http://xbox360media.ign.com/xbox360/image/article/927/927055/call-of-duty-world-at-war-20081111115354250.jpg

Assim, com os títulos estudados, os elementos de *gameplay* que se mostraram de interesse para o projeto foram:

- *Gameplay* em plataforma com cenário limitado de *Super Crate Box*;
- Ataque minimalista em forma de *dash* presente em *Dash Craft*,
- Cenários procedurais em *Spelunky*;
- Sobrevivência do jogador em meio a hordas de inimigos em um cenário interativo.

A complexidade de planejamento e execução de cenários procedurais se mostrou muito alta e poderia acabar por comprometer a produção dos elementos de animação buscados nesse projeto. Junto a esse refinamento, foi-se estipulado que o jogo a ser desenvolvido seria uma proposta com mecânicas de plataforma (movimentação lateral e pulo), somadas à ação de ataque em forma de *dash*.

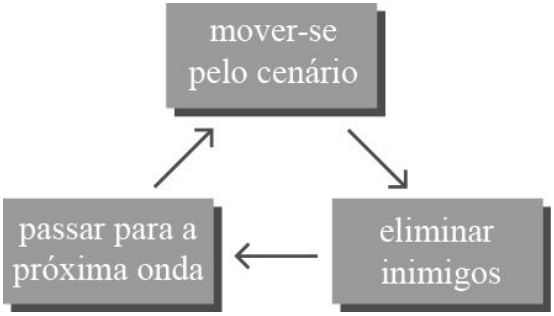
Figura 12 *- Metodologia Double Diamond aplicada no projeto*



Fonte: Desenvolvido pelo autor

O *gameplay loop* viria a consistir no jogador sobreviver a hordas de inimigos em um cenário limitado com elementos interativos que modificam de maneira sutil o *layout* da fase, enquanto elimina os adversários baseado na movimentação e execução do *dash*.

Figura 13 *- Gameplay Core Loop do jogo a ser desenvolvido*

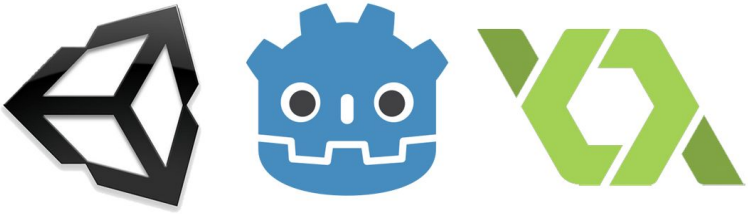


Fonte: Desenvolvido pelo autor

4.1.2 O desenvolvimento do protótipo de mecânicas

Para a prototipação foi necessário escolher um programa, mais especificamente uma *engine* de jogos, que pudesse servir para fácil prototipagem. Tendo em vista o escopo do jogo, as *engines* conhecidas no mercado que se mostraram mais adequadas foram Unity, Godot e Game Maker.

Figura 14 *- Game engines mencionadas*



Fonte: Montagem desenvolvida pelo autor

A *engine* Unity possui uma grande comunidade que fornece diversos tipos de tutoriais e suporte aos desenvolvedores, além de um extenso número de jogos lançados para o mercado em seu catálogo. A *engine*, além de possuir grandes títulos 3D desenvolvidos, tem também jogos como *Cuphead* e *Hollow Knight* (*Team Cherry*, 2017) entre games 2D que chama atenção no mercado. O fator que tornou a *engine* Unity pouco acessível a este projeto foi sua linguagem de programação, que exige um conhecimento maior e mais específico da área por utilizar C#, UnityScript ou Boo. Visto que o foco deste projeto é principalmente desenvolver um jogo estilo arcade baseado na aplicação das animações, o uso a *engine* Unity iria tornar a etapa de programação um obstáculo comprometedor.

Figura 15 *- Jogos Cuphead e Hollow Knight*

Fonte: Montagem desenvolvida pelo autor

Godot é uma *engine* open source que vem ganhando destaque por ter expandido seu alcance de desenvolvimento. Até o momento do desenvolvimento deste jogo, não foram desenvolvidos muitos títulos populares lançados no mercado feitos nesta *engine* justamente por ser considerada relativamente nova com sua versão mais aprimorada. Seu uso poderia também comprometer o desenvolvimento deste projeto fazer com que o entendimento da *engine* em si tomasse muito tempo. A linguagem de programação usada na Godot Engine é mais simples do que as linguagens usadas na Unity, posto que é baseada em Python. Ainda assim, o autor deste projeto busca uma opção que não exija muito conhecimento aprofundado de programação.

Por fim, a *engine* Game Maker se mostrou a mais adequada para o escopo deste jogo por possuir uma linguagem de programação muito simples e acessível, além de um catálogo de vários jogos lançados no mercado que se encaixam com a proposta do game a ser desenvolvido. Entre os títulos famosos desenvolvidos com Game Maker, encontramos *Nuclear Throne* (Vlambeer, 2015), *Undertale* (Toby Fox, 2015), *Hotline Miami* (Dennaton Games, 2012), e os recentes *Swords of Ditto* (onebitbeyond, 2018) e *Forager* (HopFrog, 2019). A *engine* é muito

elogiada e popular entre desenvolvedores independentes por possibilitar uma etapa de prototipagem muito rápida, permitindo ajustes e modificações rápidas no projeto sem que seja investido muito tempo.

Figura 16 *- Jogos Nuclear Throne e Swords of Ditto*

Fonte: Montagem desenvolvida pelo autor

Após a seleção da *engine* Game Maker Studio para a execução deste projeto, deu-se início à criação do primeiro protótipo. Neste programa, temos algumas divisões em relação ao conteúdo criado para o jogo que valem ser esclarecidas. São elas: *sprites*, descritos pelo site da *engine* como representação visual dos objetos criados no jogo; objetos, que são recursos usados para controlar, agir e reagir dentro do jogo; e *scripts*, que são linhas de código que comandam os acontecimentos do jogo e estabelecem as regras. O passo inicial foi definir as mecânicas básicas presentes no jogo. Com base nas pesquisas anteriores, foi estipulado que o jogador poderia executar as ações de: deslocamento horizontal, pulo padrão, pulo na parede para maior mobilidade vertical, e *dash*.

Para os inimigos, foram definidos 3 modelos. O primeiro é inimigo básico, que se movimenta horizontalmente e sofre ação da gravidade. Toda vez que este inimigo encontra um obstáculo como uma parede, sua direção de deslocamento é invertida. O segundo tipo de inimigo é um aprimoramento do primeiro, desta vez com o adicional de executar um *dash* semelhante ao do jogador que é acionado quando o

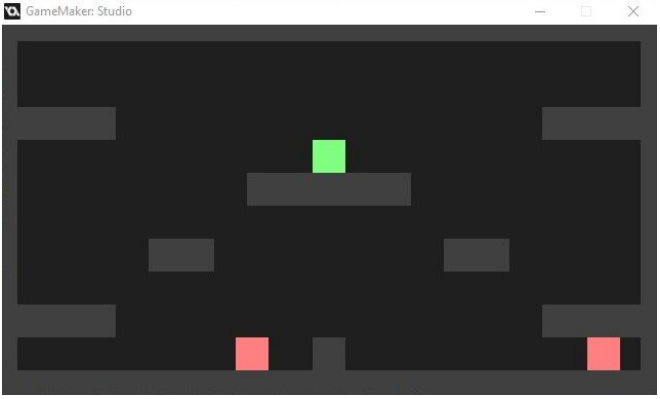
personagem do jogador se encontra dentro de seu campo de visão. O campo de visão é determinado por um semicírculo invisível orientado de acordo com a direção de deslocamento do personagem do inimigo, interpretando como visíveis todos os objetos dentro deste semicírculo. Por fim, o último inimigo não sofre ação da gravidade, desloca-se pelo cenário como se estivesse voando e perseguindo o jogador, além de possuir a capacidade de executar um *dash*.

Como forma de construir os cenários, foram definidos os blocos sólidos como elementos básicos, além de 2 tipos de elementos interativos: uma alavanca que aciona espinhos no cenário, funcionando como uma outra forma de atacar os inimigos; e blocos que podem ser construídos para alterar a rota dos inimigos, semelhante às barricadas vistas no modo *Zombies* do título *Call of Duty*.

O sistema de progressão do jogo ocorre de maneira simples, com o início da primeira onda de inimigos começando assim que se inicia a fase. Cada onda conta com um número de inimigos a serem gerados, e assim que todos são eliminados ocorre um pequeno intervalo de tempo, seguido pelo início da próxima onda. O objetivo do jogador é sobreviver o máximo que puder ao longo das ondas, usando sua mobilidade para posicionamento, *dash* e acionamento das armadilhas de espinhos para eliminação de inimigos, e os blocos acionáveis para alteração do layout da fase e controle de posicionamento dos inimigos.

Inicialmente os pilares de games de plataforma como movimentação lateral e pulo para o personagem do jogador foram programados, evitando trabalhar qualquer aspecto relacionado a gráficos. Para isto, foram usados apenas quadrados de cores distintas para representarem o personagem, blocos e inimigos. Com o estabelecimento de velocidade de movimentação e gravidade no protótipo, foram adicionadas velocidade de aceleração e desaceleração ao jogador para trazer uma transição mais suave ao seu deslocamento.

Figura 17 *- Protótipo inicial do jogo*



Fonte: Screenshot feito pelo autor

Um balanceamento inicial foi feito com relação à intensidade da gravidade, altura do pulo do personagem e sua velocidade de deslocamento. Segundo Scott Rogers,

> Determinar a métrica começa com a altura básica do personagem, a velocidade que o personagem tem e a altura que o personagem pode alcançar. Eu sempre uso o personagem herói como um parâmetro para o resto do mundo. (ROGERS, pg 117, 2014)

O tamanho escolhido para o personagem foi de 32 *pixels* de altura por 32 *pixels* de largura. Este tamanho se mostrou pequeno o suficiente para valorizar o estilo pixel art, ao mesmo tempo que proporciona maior quantidade de detalhes se comparado a *sprites* de Super Mario Bros. por exemplo, que possuem metade desta altura e largura.

A etapa seguinte foi adicionar a mecânica do *dash*, que consiste em fazer com que o personagem se desloque na direção atual com uma velocidade escolhida (maior que seu deslocamento básico) por um tempo determinado. Outro fator adicionado é que o jogador não sofre influência da gravidade enquanto se encontra no estado de *dash*, trazendo uma sensação de velocidade alta o suficiente para planar quando o personagem está no ar. Após o término do período de *dash*, o

personagem do jogador tem que esperar um intervalo de tempo para poder executar o *dash* novamente como forma de evitar o uso contínuo da mecânica e abrindo janela para vulnerabilidade.

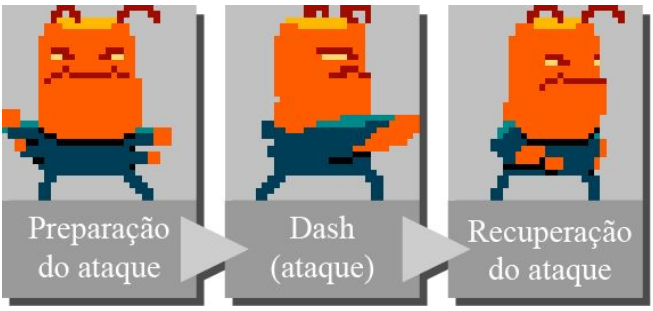
Juntamente com a produção da ação de *dash* no protótipo, foi feita a base do objeto que viria a ser o inimigo mais básico. O mesmo sofre influência da gravidade e movimenta-se como o jogador, com a diferença de trocar de direção quando colide com uma parede a sua frente e também ser destruído ao colidir com o jogador em estado de *dash*.

Foi programado o estado de atordoamento do jogador, que é quando o mesmo, fora do estado de *dash*, colide com qualquer inimigo que não esteja atacando. O estado de atordoamento do jogador impede que ele possa pular ou atacar por um curto período de tempo, penalizando o jogador por colidir com inimigos mas ainda assim oferecendo a opção de se deslocar horizontalmente pelo cenário para uma tentativa de recuperação.

Por fim, o objeto do jogador recebeu a ação de pulo na parede (ou popularmente conhecida como wall jump), permitindo ter uma movimentação vertical mais livre dentro do cenário.

Para a criação da segunda variação de inimigo, foi usado o modelo de inimigo já existente mesclado com a mecânica de *dash* do jogador, desta vez sendo acionada não por um input do usuário e sim de acordo com o posicionamento do inimigo em relação ao jogador. Diferente do jogador, o inimigo executa uma ação de preparo antes de atacar, mostrando uma animação que indique isto e permitindo que o jogador reaja antes do golpe. Quando o jogador se encontra dentro do raio de visão do inimigo e este está apto a atacar, é acionado o estado de carregamento, que após um tempo determinado executa então a ação de *dash*. Após o término do *dash*, o inimigo para sua movimentação por um tempo determinado para representar sua recuperação. Assim, o segundo inimigo possui 3 estados que compõem seu ataque e que virão a ser utilizados na execução das animações desta ação.

Figura 18 *- Estados de ataque de um dos inimigos*



Fonte: Desenvolvido pelo autor

A última variação de inimigo foi planejada para imitar um personagem voador. Sua ação de ataque possui os 3 estados descritos no inimigo anterior, com a diferença acontecendo nos valores de velocidade e tempo de duração de cada segmento. Além disso, quando fora do estado de ataque, o inimigo voador se desloca buscando a direção entre sua localização atual e a localização do jogador, desviando de objetos sólidos no meio do caminho.

Para o controle do andamento da partida, foi criado um objeto invisível que controla o nível (ou wave, como usado no jogo) atual do jogador. Isso implica em controlar quantos inimigos serão gerados, em que momento e local irão surgir.

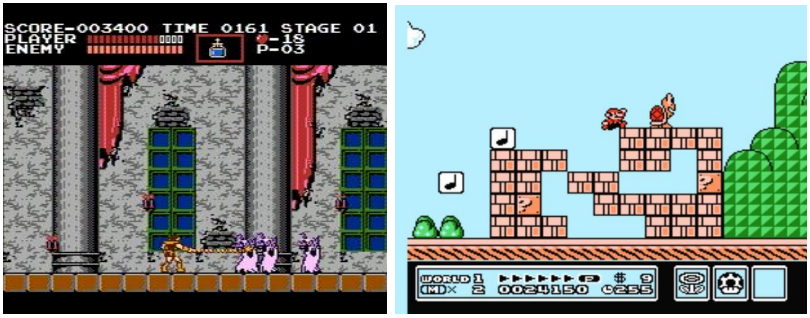
Com a criação de todos os elementos citados, o protótipo apresentou a jogabilidade desejada para o projeto, podendo ser executado o loop de *gameplay* que o jogador viria a experienciar.

4.2 ARTE

4.2.1 Escolha do estilo gráfico e referências

Como escolha de gráficos, foi selecionado o estilo pixel art, que busca renderizar telas em baixa resolução e relembra muitos títulos lançados para plataformas como Super Nintendo. O estilo se mostra presente principalmente no mercado de jogos independentes, muitas vezes remetendo a jogos que buscam um gameplay semelhante aos *arcades*. Uma vez que o projeto a ser desenvolvido busca trazer uma experiência e sensação de jogos retro, títulos mais antigos como Castlevania (1986) e Super Mario Bros. 3 (1988).

***Figura 19** - Jogos Castlevania e Super Mario Bros. 3 para NES*



Fonte: Montagem desenvolvida pelo autor

Além dos títulos lançados originalmente para Nintendo Entertainment System, jogos modernos como Shovel Knight e Super Helmknight compuseram as principais inspirações visuais. Estes títulos foram destacados por trazerem com clareza a sensação e linguagem de jogos considerados retro, com um trabalho em pixel art bem executado.

Figura 20 *- Jogos Shovel Knight e Super Helmknight*

Fonte: Montagem desenvolvida pelo autor

Com a escala de personagens previamente definida, um passo importante seria a escolha de cores do jogo. Os hardwares que executavam jogos nos anos 80 e 90 possuíam uma exibição de cores limitada, principalmente se comparada a plataformas modernas. Como base, foi utilizada a paleta de cores que o hardware do console Nintendo Entertainment System era capaz de renderizar, visto que ela possui cores bem características dos títulos lançados para a plataforma. A paleta de

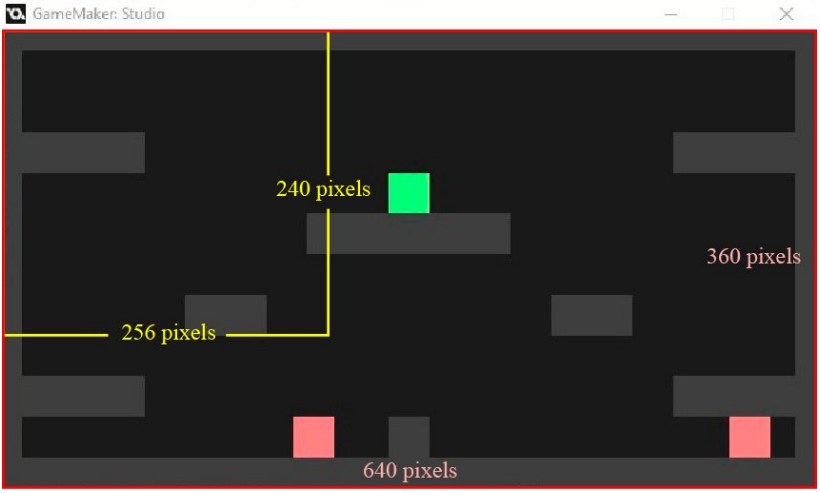
cores base para a criação dos *sprites* do jogo em desenvolvimento é a mesma do console Nintendo Entertainment System.

Figura 21 *- Paleta de cores do console NES*

Fonte:
http://www.thealmightyguru.com/Games/Hacking/Wiki/index.php/NES_Palette

Com a definição da escala dos personagens e paleta de cores a ser usada, foi necessária a definição da resolução de execução do jogo. Monitores e telas tinham, em sua maioria, a proporção 4:3 (4 unidades de largura e 3 unidades de altura), trazendo um aspecto mais quadrado se comparado aos padrões de tela atuais. O console Nintendo Entertainment System possui a capacidade de renderizar uma tela de 256 pixel na horizontal e 240 pixels na vertical (https://en.wikipedia.org/wiki/Nintendo_Entertainment_System). O teste com esta resolução se mostrou inadequado por tornar o personagem muito grande na tela, trazendo um campo de visão reduzido. Outro resultado deste primeiro teste foi a falta de aproveitamento da proporção 16:9 da tela de monitores disponibilizados atualmente, resultando em distorção da imagem ou uso de bordas laterais indesejadas. Com o aumento da resolução escolhida e uso da proporção 16:9 para adequar-se a telas mais populares, a resolução final escolhida foi de 640 pixels de largura por 360 pixels de altura.

Figura 22 *- Comparativo entre resoluções*



Fonte: Desenvolvido pelo autor

4.2.2 Adaptação da anatomia dos personagens

A criação do conteúdo visual dos personagens iniciou com sua adaptação do universo *Rotfather* para melhor adequação a um jogo arcade. Originalmente, os personagens presentes no universo *Rotfather* foram desenvolvidos com proporções de corpo e cabeça próximas a 7 cabeças de altura em um personagem adulto regular. O modelos dos personagens Takeda e Al Kane tem uma silhueta muito mais vertical do que as vistas em jogos como *Super Mario Bros*. e *Shovel Knight* (*Yacht Club Games*, 2014), que buscam um visual mais voltado ao cartoon. Títulos com personagens anatomicamente realistas costumam ter uma jogabilidade que também traga mais realismo. Em títulos dos anos 80 e 90, temos títulos como *Prince of Persia* (Jordan Mechner, 2989), *Flashback* (Delphine, 1992) e *Another World* (Éric Chahi, 1991) que traziam personagens humanoides com uma anatomia mais correspondente à que consideramos biologicamente correta. Seu estilo de *gameplay* era mais lento, com animações baseadas em rotoscopia e velocidade de deslocamento realistas. Dado que o game a ser desenvolvido busca velocidade e até mesmo um ar mais cartunesco, os

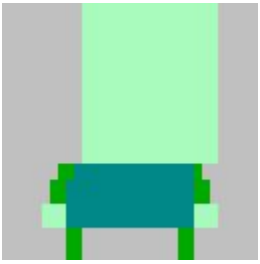
personagens deveriam ser adaptados para uma proporção menor de cabeças em relação ao corpo. O personagem Mario do título Super Mario World, por exemplo, traz uma proporção perto de duas cabeças de altura, que seria justamente o objetivo da primeira adaptação visual a ser feita.

***Figura 23** - Altura dos personagens Prince of Persia e Mario*

Fonte: Montagem desenvolvida pelo autor

A criação do novo modelo de anatomia iniciou-se com a limitação de uma área quadrada onde seria feita a blocagem dos membros dos personagens. Com a ferramenta Piskel, foi possível criar um modelo onde a cabeça dos personagens viria ocupar cerca de metade da sua altura total, enquanto o corpo e membros seriam reduzidos, trazendo um aspecto semelhante a personagens "chibi".

***Figura 24** - Modelo de anatomia para o jogo em desenvolvimento*



Fonte: Desenvolvido pelo autor

4.2.2.1 Takeda

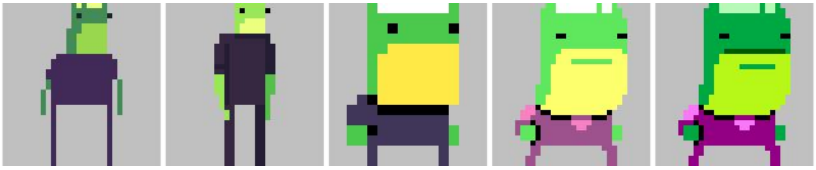
Seguindo a base criada, foram feitos testes do personagem Takeda dentro do novo modelo de anatomia.

Figura 25 *- Personagem Takeda*

Fonte: Desenvolvido pelo autor

Foram usadas cores vindas da paleta do console Nintendo Entertainment System, buscando usar poucas variações de cores e trazer bastante contraste entre cabeça e corpo. Os olhos afastados e a boca larga foram reforçados, protagonizando o espaço do rosto juntamente com a divisória de cor presente no design dos sapos do universo *Rotfather*. As mãos foram aumentadas para facilitar a leitura de seu movimento já que nem sempre farão parte do recorte da silhueta do personagem e por fim, o brilho no topo da cabeça do personagem foi adicionado para representar a textura úmida da pele dos sapos.

Figura 26 *- Adaptação do personagem Takeda para pixel art*



Fonte: Desenvolvido pelo autor

4.2.2.2 Baratas

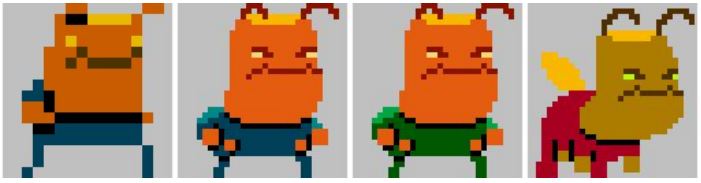
A raça inicial de personagens escolhida para os inimigos presentes nesse projeto são baratas. As adaptações necessárias que se diferenciam em relação ao personagem Takeda são o alargamento do corpo, o posicionamento do par secundário de braços que existe na anatomia das baratas do universo *Rotfather*, além da adição das antenas no topo da cabeça, que exigiu o deslocamento da cabeça para baixo para que o limite de altura do sprite fosse respeitado. A união desses elementos resultou também numa postura mais inclinada para frente para as baratas, fazendo-as parecerem mais fortes e robustas quando comparadas a um personagem mais esguio como Takeda.

Figura 27 *- Baratas do universo The Rotfather*

Fonte: Desenvolvido pelo autor

A diferenciação visual entre os 3 tipos distintos de inimigos desenvolvidos vem com a cor se suas roupas: um tipo de barata vestindo verde, outra azul e outra vermelho. Além desse fator, a barata voadora tem sua postura alterada para parecer que seu peso todo é carregado pelas asas, inclinando ainda mais a cabeça deixando os braços e pernas soltos ao ar.

Figura 28 *- Adaptação das baratas para pixel art*



Fonte: Desenvolvido pelo autor

4.2.3 Desenvolvimento das animações do personagem principal

Definição dos estados do personagem baseado nas mecânicas

Para o desenvolvimento das animações, o primeiro passo foi separar os estados que o personagem viria a ter no jogo. São eles:

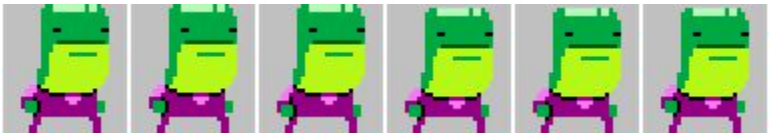
- Repouso, quando o personagem está em contato com o chão e parado, sem agir ou sofrer alguma ação;
- Corrida, quando o personagem ainda se encontra em contato com o chão, desta vez se deslocando horizontalmente;
- Pulo e queda livre;
- Ataque;
- Atordoamento, iniciado quando o jogador colide com inimigos;
- Troca de direção;
- Deslizamento vertical, acionado quando o jogador está em queda livre e colidindo lateralmente com uma parede ao mesmo tempo;

Os estados descritos foram então pensados em duas formas de aplicação: animações que seriam colocadas em *loop* enquanto o personagem se mantivesse em seu estado; e estados que são representados por apenas um quadro de animação estático, mas que é alterado e modificado conforme condições específicas sofridas pelo personagem.

4.2.3.1 Animações em loop

As animações que são executadas em *loop* são a de repouso e a corrida. A animação de repouso é a mais simples, e aqui o objetivo com ela foi passar um movimento sutil de respiração ao personagem. A solução para isto foi dividir a animação em 4 *frames* distintos, sendo 2 *keyframes* e 2 *frames* de *breakdown*. Os *keyframes* tem uma diferença sutil, porém um mostra o personagem no que seria o fim do processo de inspiração do ar, enquanto o outro mostra o fim da expiração. Os *frames* intermediários deslocam os braços e a cabeça do personagem em tempos diferentes, trazendo assim um *follow through* sutil ao movimento. A animação de repouso é executada sempre que o personagem não está executando ou sofrendo alguma ação, além de estar em contato com o chão em seus pés.

Figura 29 *- Animação de repouso do Takeda*



Fonte: Desenvolvido pelo autor

Para a animação de corrida, foi usado sistema descrito por Richard Williams em Manual de Animação como "método da pose baixa" (2016), segundo ele usado por Art Babbitt. Foram feitas primeiramente as duas poses baixas da corrida do personagem e em seguida, são adicionadas a poses de subida.

Figura 30 *- Frames principais da animação de corrida do Takeda*

Fonte: Desenvolvido pelo autor

Por fim são feitos os *frames* intermediários para maior fluidez. O personagem Takeda recebeu também uma adaptação em sua pose, desta vez inclinando o corpo para frente e projetando os braços para trás.

Os braços foram feitos para haver um follow through sutil em seu movimento, impulsionado pelo balanço do torso do personagem. Os membros e partes do corpo buscaram manter o máximo de fidelidade com o modelo de anatomia descrito anteriormente no que diz respeito ao número de pixels dispostos por membro, ao mesmo tempo que se busca um resultado visual agradável. A execução desta animação se dá quando o jogador está em contato com o chão, sem execução nenhuma ação de ataque ou sofrer dano, além de possuir velocidade de deslocamento horizontal maior do que zero. A velocidade que a animação de corrida é executada obedece a velocidade de deslocamento do personagem, trazendo assim um ritmo de passo correspondente ao espaço percorrido.

Figura 31 *- Animação de corrida completa do Takeda*

Fonte: Desenvolvido pelo autor

4.2.3.2 Animações de 1 *frame*

Após a conclusão das animações de repouso e corrida para o personagem Takeda, foi necessário desenvolver uma transição entre os dois estados a fim de atingir maior fluidez em sua passagem. Como mecanicamente o personagem possui uma velocidade de aceleração, gradualmente aumentando seu deslocamento horizontal até que atinja sua velocidade máxima, foi feito um *frame* que representasse o início dessa aceleração. A pose é um intermediário entre as poses de repouso e de corrida, na medida em que descrevem uma silhueta contrastante. O acionamento desta animação se dá quando o personagem está com uma velocidade horizontal abaixo do valor desejado e em contato com o solo, sem executar nenhuma ação de ataque.

Figura 32 *- Frame intermediário entre repouso e corrida do Takeda*

Fonte: Desenvolvido pelo autor

A animação de pulo e queda vertical foi separada em 3 *frames* distintos, que são executados de acordo com condições de velocidade vertical do personagem.

O primeiro *frame* representa a etapa de subida no arco do pulo do jogador, sendo executado quando a velocidade vertical está dentro de um valor escolhido que é atingido nos primeiros momentos da ação de pulo. Sua pose coloca o rosto e cabeça do personagem Takeda projetados para cima, com braços apontando para baixo e uma das pernas dobrada, enquanto a outra se estica. Este *frame* é executado

quando o personagem se encontra em uma velocidade de ascensão vertical, sem executar o *dash* ou sofrer dano.

O segundo *frame* foi feito para representar o topo do arco de pulo, fazendo o personagem parecer que está flutuando. Sua cabeça começa a ser envolvida pelos ombros e os braços sobem, como se continuassem seguindo a velocidade vertical da etapa de subida. Suas pernas se encontram levemente dobradas. O segundo *frame* é executado quando está dentro de um intervalo de velocidade vertical próximo de zero, que seria essa etapa de "flutuação" do pulo.

Finalmente, o terceiro *frame* representa tanto a etapa de descida do pulo quanto a queda livre do personagem. Sua pose agora põe a cabeça mais abaixo do que os *frames* anteriores, elevando os ombros e os braços que sofrem com a velocidade. As pernas estão dispostas de forma contrária ao primeiro *frame*, desta vez com a perna que antes estava esticada ficando dobrada e vice-versa. Assim, quando executados em sequência, os 3 *frames* descritos mostram as pernas alternando de posição, como se o personagem fizesse metade de um ciclo de caminhada no ar. O terceiro *frame* de pulo é executado quando o personagem se encontra com velocidade de queda maior do que um valor numérico escolhido. O jogo lê queda livre e pulo como mesmo estado do personagem, sendo assim a execução dos *frames* se dá baseada em sua velocidade vertical enquanto não colide com o chão.

Figura 33 *- Frames de pulo do Takeda*

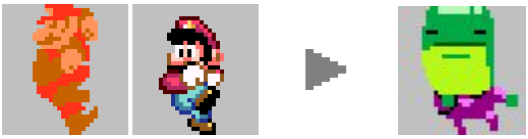
Fonte: Desenvolvido pelo autor

Como forma de polir ainda mais a movimentação do personagem e trazer fluidez, foi planejado e feito um *frame* que representa a mudança de direção do personagem quando ele está correndo.

Esta animação é inspirada no mesmo tipo de solução que é visto desde o primeiro título na série *Super Mario Bros.*, que é executada quando o jogador está, por exemplo, correndo para a direita e repentinamente interrompe o movimento, apertando o comando de

movimentação na direção oposta. Assim, o personagem fica em uma pose como se estivesse "deslizando" no chão, enquanto reduz sua velocidade até mudar a direção para a desejada, fazendo a transição de direções enquanto se movimenta.

Figura 34 *- Frame de troca de direção do Takeda*



Fonte: Desenvolvido pelo autor

Como foi optado por adicionar a mecânica de pulo na parede ao jogo, que consiste em possibilitar que o personagem pule enquanto colide com uma parede à sua frente e não com o chão em seus pés, foi desenvolvida uma pose para representar o personagem escorregando pela parede. Dado que a mecânica também implica em reduzir a velocidade vertical de queda enquanto o personagem desliza na parede, o *frame* traz uma pose que mostra Takeda apoiando uma mão e uma perna na parede, enquanto olha para o lado oposto.

Figura 35 *- Frame de deslizamento vertical do Takeda*

Fonte: Desenvolvido pelo autor

O estado de atordoamento do personagem precisava representar de forma simples e clara que o jogador havia sofrido dano, para que pudesse alterar sua estratégia conforme a situação pede. Assim, o personagem Takeda foi representado com suas mãos na barriga e pernas encolhidas, enquanto apresenta uma expressão facial de dor com olhos fechados e boca aberta. Este estado é o único que traz um contraste grande de alteração de expressão facial, que teve o objetivo justamente

de se tornar um elemento facilitador de leitura para alertar o jogador. Este *frame* é executado quando o jogador colide com inimigos que não estão atacando, dando início ao estado de atordoamento que se encerra automaticamente de acordo com o tempo estipulado no jogo.

Figura 36 *- Frame de atordoamento do Takeda*

Fonte: Desenvolvido pelo autor

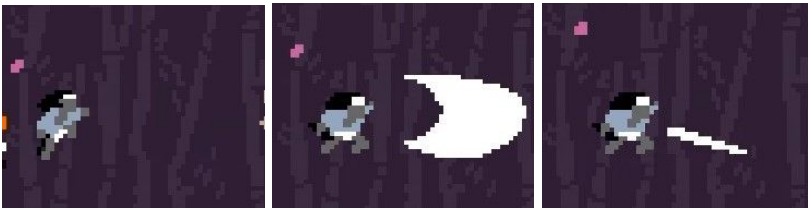
Para a ação de ataque do jogador, ou no caso o *dash*, sendo que a velocidade de deslocamento alta o suficiente para comprometer a leitura de detalhes do *sprite*, a solução para enriquecer a animação de ataque foi apoiar-se no movimento feito pela espada do personagem. Como dito por Mariel Cartwright em sua palestra "*Fluid and Powerful Animation Within Frame Restrictions*" feita na GDC em 2014, "*Smears will also fill in these gaps if you find that your spacing is large*". Assim, enquanto o personagem Takeda fica em pose estática retirada de um dos *frames* de corrida, a animação da sua espada é representada por um grande *smear*, ou borrão de movimento, que finaliza em um *ease out* sutil.

Figura 37 *- Frames de animação de ataque do Takeda*

Fonte: Desenvolvido pelo autor

A mesma solução pode ser vista no título Samurai Gunn, que traz personagens de movimentação ágil que possuem ataque com espadas. Apesar destes não executarem um deslocamento horizontal brusco como Takeda durante o ataque, seus golpes trazem toda a movimentação e fluidez na animação da espada, enquanto o personagem em si se mantém em um *frame* único.

Figura 38 - *Frames de animação de ataque do jogo Samurai Gunn*



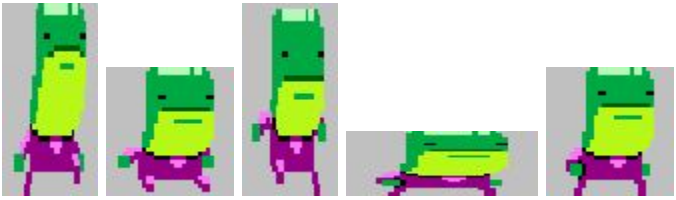
Fonte: Screenshots retirados pelo autor

4.2.3.3 Squash e Stretch executado por código

Como um adicional a todas as animações previamente descritas, foram acrescentados squash e stretch ao personagem, de forma a realçar a sensação de velocidade e elasticidade do personagem. A deformação na forma do personagem é feita de acordo com o estado que ele se encontra e sua velocidade. Por exemplo: durante o pulo, o personagem irá reduzir sua largura e aumentar sua altura conforme sua velocidade se afasta de zero, quando ao entrar em contato com o chão, sua largura é aumentada e sua altura reduzida, simulando um achatamento por conta do impacto ao encostar no chão.

Além dos dois exemplos descritos acima, foi adicionada a deformação ao *sprite* do personagem quando está no estado de *dash*, Esta traz um grande aumento na sua largura, capaz de aproximar o *sprite* de um *smear*, ao mesmo tempo que busca manter uma área total consistente.

Figura 39 - *Deformação dos frames do personagem Takeda*



Fonte: Desenvolvido pelo autor

4.2.4 Desenvolvimento das animações dos personagens secundários

Para personagens que não são controlados pelo jogador, teremos apenas as 3 variações de inimigos para este projeto. Suas animações foram separadas de acordo com suas capacidades mecânicas e com a forma que viriam a interagir com o personagem do jogador. Além das animações em *sprites*, as baratas inimigas também receberam adição de stretch e squash por código como Takeda, com a adição de uma animação extra por código executada para a barata voadora que será explorado mais adiante.

4.2.4.1 Animações em *sprite*

Para as animações feitas em *sprites*, as referentes aos dois inimigos terrestres se assemelham muito, enquanto o maior diferencial fica com a barata voadora. Para os terrestres, foi feito um idle simples, usando a mesma técnica usada em Takeda, porém desta vez com os diferenciais anatômicos das baratas em mente. Sua caminhada também foi feita com base na mesma técnica da "pose baixa" previamente descrita, com a adaptação de menor distanciamento entre os elementos para que a barata passe a impressão de estar caminhando (e não correndo), visto que mecanicamente sua velocidade de deslocamento é menor se compara à do jogador.

Figura 40 *- Frames principais de caminhada das baratas*

Fonte: Desenvolvido pelo autor

Foram adicionados também os *frames* intermediários e também um follow through às antenas das baratas para reforçar o peso de seu caminhar.

Figura 41 *- Frames de animação de caminhada completa das baratas*

Fonte: Desenvolvido pelo autor

A barata voadora, por sua vez, traz o diferencial de não ter uma animação de caminhada e repouso separadas, mas sim uma única animação que funciona tanto para seu deslocamento quanto para repouso. A animação é uma simples solução de 2 *frames* que mostram

suas asas batendo, que vem ser complementada com um deslocamento vertical que representa o movimento de balanço no ar que será descrito adiante.

Figura 42 - Frames da animação de vôo das baratas

Fonte: Desenvolvido pelo autor

Para as ações de ataque das baratas, foram divididos 3 estágios que compõem seu dash: preparo, deslocamento e recuperação. Durante a etapa de preparo as baratas fazem uma antecipação do seu ataque, puxando a mão que vai desferir o golpe para trás, mantendo o último *frame* parado uma vez que é alcançado.

***Figura 43** - Frames da animação de preparo do ataque das baratas*

Fonte: Desenvolvido pelo autor

Em seguida, após a duração do tempo de preparo, é executado o *frame* da animação do ataque em si, que se mantém no mesmo *frame* durante todo o período de deslocamento com um smear, até seu encerramento que para o deslocamento da barata e executa a animação de recuperação. Para a recuperação, a barata descreve uma transição entre o *frame* de seu deslocamento de ataque e seu *frame* inicial de repouso. O tempo de duração das animações descritas acima é determinado conforme a duração e velocidade dos ataques, na medida em que estes alteram a dificuldade de enfrentar os inimigos.

Figura 44 *- Frames da animação de ataque das baratas*

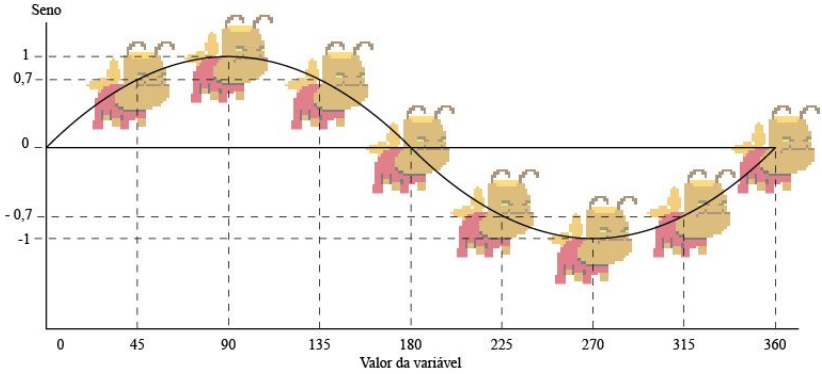
Fonte: Desenvolvido pelo autor

4.2.4.2 Animação por código

Para complementar a animação feita no *sprite* de repouso e deslocamento da barata voadora, buscou-se adicionar um movimento de balanço vertical que trouxesse a sensação do personagem estar flutuando. Como solução, este movimento foi feito adicionando um modificador da posição vertical onde o *sprite* da barata voadora é renderizado.

Foi criada uma variável que é um valor numérico que varia de 0 a 360 e cada instante que se passa no jogo ela tem seu valor atual aumentado constantemente. Quando chega em 360, esta variável retorna a 0 e recomeça o processo de adição. Na etapa de renderização do *sprite*, o objeto da barata voadora renderiza o *sprite* correspondente em sua posição real na tela com a soma do valor do seno da variável descrita anteriormente. Assim, um movimento de deslocamento vertical é descrito com *slow in* e *slow out* por conta da natureza do valor do seno que se modifica de forma não linear em relação ao aumento linear do valor da variável.

Figura 45 *- Representação visual da técnica usada para animar vôo*



Fonte: Desenvolvido pelo autor

Esta técnica é simples e pôde ser adicionada em outros elementos que precisam descrever movimentos com slow in e slow out como será visto adiante.

4.2.5 Implementação de outras animações

Para enriquecer o gameplay e deixar o jogo mais orgânico, foram adicionados elementos que não interagem com o jogador nem alteram o funcionamento das mecânicas. Evitando usar sistemas de partículas vinculados à *engines* para melhor entendimento, foram feitos objetos que simulam partículas e efeitos visuais para que as ideias de suas execuções possam inspirar outras aplicações.

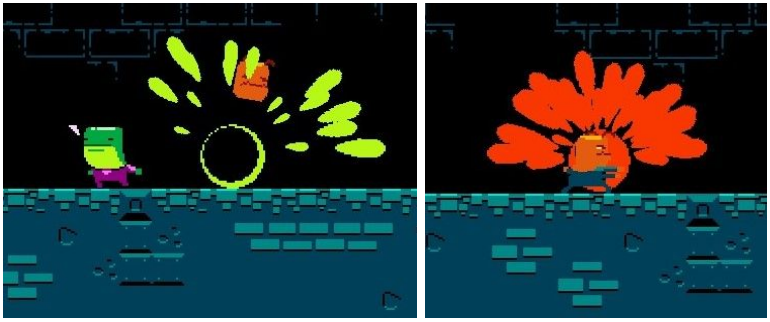
4.2.5.1 Partículas

O primeiro efeito visual a ser implementado foi o sangue dos personagens. O sangue dos personagens foi dividido em 3 tipos de objetos: a explosão principal, as gotas emitidas e a explosão secundária.

A explosão principal é uma animação simples feita a partir de um círculo que se expande, enquanto um círculo menor em seu interior também aumenta, desta vez funcionando como “negativo” da imagem. Após o término do loop da animação, o objeto é excluído da tela.

Em paralelo, as gotas são imagens estáticas que diminuem de tamanho conforme passa seu tempo de duração na tela e recebem uma velocidade e direção iniciais que são escolhidas aleatoriamente dentro de limites pré-determinados quando criados. Assim, as gotas descrevem um movimento em arco até colidirem em algum objeto, gerando a explosão secundária já mencionada. A explosão secundária é uma versão reduzida da animação de explosão principal, buscando descrever a gota de sangue sendo espalhada na colisão. Com o término do loop desta última animação, o objeto se exclui.

***Figura 46** - Explosões de partículas de sangue*



Fonte: Desenvolvido pelo autor

4.2.5.2 Efeitos visuais

Como complemento da ação de pular, foi adicionado um efeito visual semelhante a um flash projetado verticalmente sob os pés do personagem. A mesma animação foi usada para quando o personagem efetua o pulo na parede, trazendo mais impacto à ação.

Figura 47 *- Frames do efeito de pulo do personagem*

Fonte: Desenvolvido pelo autor

Durante o combate entre jogador e inimigo, caso ocorra a situação onde ambos colidem durante seus respectivos golpes, o resultado é o anulamento dos dois ataques, com nenhum dos dois personagens mortos. A decisão foi feita baseada em mecânicas de *parry* que ficaram populares com os títulos da franquia *Dark Souls* (From Software, 2011), onde o jogador pode responder ao ataque do inimigo em um curto período de tempo antes de ser acertado com um comando específico, não só anulando o ataque do adversário mas também abrindo janela para um contra-ataque.

No título em desenvolvimento não foi feito um comando correspondente à ação de *parry*, mas sim utilizado o próprio comando de ataque para tal. Como acréscimo para este tipo de evento foi adicionado um efeito semelhante a um flash, inspirado no mesmo efeito usado em *Super Mario World*, quando o jogador chuta um casco de tartaruga para longe por exemplo. A animação em si é simples, onde no primeiro *frame* é a imagem do flash e no segundo é a mesma imagem, porém espelhada horizontalmente para dar uma sensação de movimento e em seguida o efeito some.

***Figura 48** - Frames que mostram efeito de impacto*

Fonte: Screenshots feitos pelo autor

Como forma de trazer um toque mais orgânico e vivo ao cenário, foram adicionados objetos que seriam montes de grama que balançam como se estivessem fluindo com uma corrente de vento. Para a criação desses objetos, foram feitos diferentes padrões de grama, que somados com a mesma técnica de deslocamento na animação da barata voadora, passam a sensação de estarem se movendo de forma suave de um lado para o outro. A adaptação necessária foi converter o deslocamento vertical original do código de animação da barata para uma distorção horizontal no *sprite*. Ainda por cima, foi adicionada a interação de cortar a grama, que quando em contato com o jogador em estado de *dash*, muda para uma imagem de grama cortada e emite partículas menores de grama.

***Figura 49** - Grama usada no jogo em desenvolvimento*

Fonte: Desenvolvido pelo autor

4.3 ACRÉSCIMOS

Neste segmento, será mostrado o processo de produção de alguns elementos que compõem o projeto e não são diretamente relacionados a animação.

4.3.1 Cenário

A produção da arte relacionada aos cenários começou com uma busca de referência de títulos que usam uma ambientação parecida com a escolhida para o jogo: ambiente interno e escuro, com paredes de tijolos. Foi muito importante entender como os artistas de jogos para NES utilizavam da paleta de cores de forma que o cenário não chamasse mais atenção do que os personagens por conta das poucas cores disponíveis.

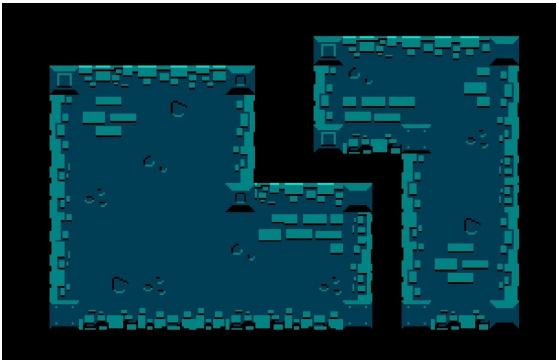
Inicialmente, os jogos Castlevania e Batman para NES se mostraram promissores por conta da sua atmosfera. Batman traz cenários bem escuros, com predominância da cor preta e fundos com paredes sujas e urbanas, que serviram de inspiração para os elementos de cenário a serem feitos. É possível ver que além do padrão de tijolos básico, há também placas maiores que parecem ser de concreto nas paredes do fund. Castlevania traz menos predominância do preto na maior parte do tempo, porém ainda é possível ver como sombras duras moldam o volume de alguns elementos, principalmente as colunas na imagem de exemplo. Por outro lado, é possível ver como a repetição do padrão de tijolos em Castlevania acaba por deixar o cenário um pouco mais artificial. Assim, além de padrões básicos de tijolos, foram feitas placas maiores para quebrar a monotonia do cenário de fundo.

Figura 50 *- Jogos Batman e Castlevania para NES*

Fonte: Montagem feita pelo autor

A ferramenta Game Maker, assim como muitas outras *engines* de jogos, conta com o recurso de montagem de tiles. Tiles são usados como uma espécie de mosaico, que permite que o responsável por montar o background do jogo possa usar as peças individuais deste mosaico para construir um cenário maior com certa variedade. Os tiles que viriam compor as paredes que colidem com os personagens do jogo foram elaborados com o objetivo de deixar a leitura daquilo que é interativo fácil e clara ao jogador. A cor base de preenchimento é o tom de azul mais escuro presente na paleta de cores escolhida, com texturas de tijolos claros nas bordas. Como adicional, foram feitas variações de rachaduras e imperfeições para serem colocadas no meio dos blocos. Para as quinas, foram feitos pequenos blocos individuais que funcionam tanto em quinas internas como externas, diminuindo a quantidade de variações de tiles necessárias, além de trazer variedade visual.

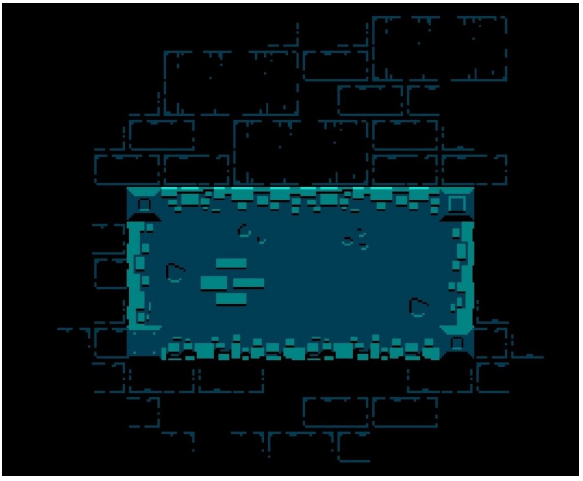
***Figura 51** - Exemplo de bloco montado com tiles desenvolvidos*



Fonte: Desenvolvido pelo autor

Como forma de preencher o fundo preto sem chamar muita atenção, foram feitos padrões de tijolos no tom mais escuro de azul, porém mantendo o preenchimento em preto. Assim, o contorno passa a sensação de ser um fundo de tijolos, ainda mantendo a ambientação escura ao fundo.

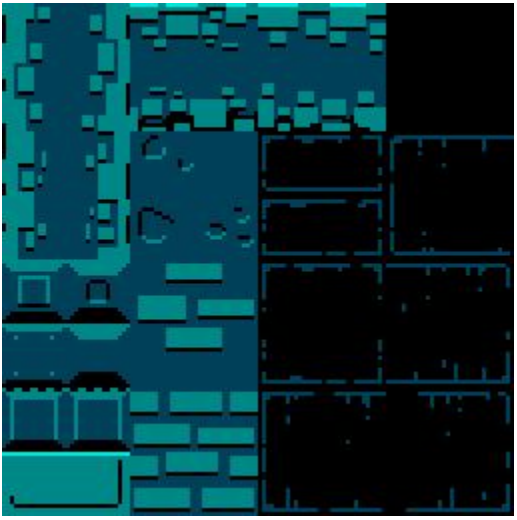
***Figura 52** - Exemplo de tiles para fundo de cenário*



Fonte: Desenvolvido pelo autor

Baseado nos módulos que compõem a arte do *background*, foi feita a composição que contém todos os elementos de cenário a serem utilizados pela ferramenta de uso de tileset do Game Maker.

Figura 53 *- Tileset usado no jogo*



Fonte: Desenvolvido pelo autor

4.3.2 Efeitos finais

Como forma de aumentar o impacto de cada inimigo derrotado, foi adicionado um efeito conhecido como *screenshake* ao jogo. Toda vez que o jogador elimina uma barata com seu *dash*, a tela é inclinada alguns graus no sentido horário ou anti-horário, e rapidamente retorna à sua angulação normal. O resultado desta sequência é uma tremida na tela que reforça o peso do ataque do jogador.

Figura 54 *- Demonstração da inclinação na câmera*

Fonte: Desenvolvido pelo autor

Por fim, quando o jogador morre, o jogo reduz sua velocidade de execução para trazer uma sensação de câmera lenta e dramatizar o momento. Após o evento que mata o personagem do jogador, a velocidade vai gradualmente retornando ao seu valor normal e em seguida surge o menu que permite que o jogador reinicie a partida ou volte ao menu principal.

5 CONCLUSÃO

O desenvolvimento de animações para jogos digitais requer, além do conhecimento de animação para filmes animados, uma noção de como funcionam os gatilhos de animação e as interações relacionadas às mecânicas do jogo.

O entendimento de *game design* auxilia o animador a planejar sua animação dentro das restrições de jogabilidade, enquanto a linguagem de programação pode servir também como ferramenta para montar animações por código. Apesar de cada vez mais ramificadas, as áreas de conhecimento para a produção de um jogo requerem da pessoa uma ideia do funcionamento de todas as outras áreas envolvidas.

Existem estúdios independentes que são compostos por apenas uma pessoa, sendo esta responsável por todas as áreas de produção do game. Nestes casos, como feito no jogo Undertale, o profissional acaba tendo controle de quase todos os parâmetros que se comunicam entre *game design*, animação e programação, podendo estabelecer um fluxo de funcionamento eficiente entre eles. Em estúdios maiores, que são compostos por profissionais especializados em suas respectivas áreas de produção, a qualidade do funcionamento do jogo vai depender não somente da habilidade destas pessoas em seu cargo, mas também da eficiência na comunicação entre os diferentes segmentos que compõem um jogo digital.

Com o desenvolvimento deste projeto, foi possível perceber alguns dos casos comuns onde a animação deve ser adaptada e planejada de acordo com restrições e guias feitos pelo *game designer*. O uso da animação de antecipação, por exemplo, que foi deixada de lado quando nos referimos ao personagem do jogador para favorecer um *gameplay* ágil. Com os inimigos, por outro lado, buscou-se o caminho oposto, tentando evidenciar bem a antecipação de seu ataque para que o jogador possa trazer uma resposta a isso.

Foi possível ver também a aplicação de programação simples para contribuição com animação em tempo real, permitindo uma interação direta entre mecânica e animação. Técnicas simples aplicadas neste projeto, como strech e squash baseados na velocidade de queda do personagem, acabam por enriquecer a resposta visual entregue ao jogador. É possível ver em muitos títulos como a animação de alguns elementos é aplicada baseada em programação, como em movimentos

de cabelos e roupas de personagens. O movimento de tecidos e fios de cabelo em títulos como *The Witcher 3: Wild Hunt* (CD Projekt RED, 2015) permitem mostrar um *follow through* composto por uma naturalidade muito difícil de alcançar se aplicada baseada em animações previamente feitas.

Para produtores de games pequenos, o mercado atual traz muitos títulos independentes que podem servir de inspiração por trazerem um produto de qualidade mesmo com baixo orçamento ou equipe pequena. Títulos como *Thomas Was Alone*, de 2012, que traz um minimalismo muito forte em seu gráfico, é enriquecido por sutilezas nas animações de deformação dos personagens e até mesmo simples movimentos de câmera.

O jogo com a produção descrita aqui busca estender seu desenvolvimento para trazer mais conteúdo aos jogadores e poder ser lançado no mercado. Inicialmente, os planos futuros envolvem desenvolvimento de múltiplos personagens jogáveis, com habilidades e funcionamentos diferenciados entre si, além de maior variedade de níveis e inimigos. O universo *The Rotfather* continuará sendo o embasamento para criação de personagens e ambientes, podendo desenvolver assim um título complementar aos produtos pertencentes ao projeto.

REFERÊNCIAS

BYCER, Josh. **The Critical Core Gameplay Loop**. **Medium**, Estado, abril 2019. Disponível em: <https://medium.com/super-jump/the-critical-core-gameplay-loop-d8a70b3975cc>. Acesso em: 11 jun. 2019.

COOPER, Johathan. **The 12 principles of animation in video games**. **gamasutra**, Estado, 13 maio 2019. Disponível em: <https://www.gamasutra.com/view/news/342457/The_12_principles_of_animation_in_video_games.php>. Acesso em: 11 jun. 2019.

EBERHART, Kaleb. **Development of Difficulty in Games**. **gamasutra**, Estado, 26 fev. 2019. Disponível em: <https://www.gamasutra.com/blogs/KalebEberhart/20190226/337341/Development_of_Difficulty_in_Games.php>. Acesso em: 11 jun. 2019

KING, Alexander. **The Key Design Elements of Roguelikes**. **gamedevelopment**, Estado, 10 abril 2015. Disponível em: <https://gamedevelopment.tutsplus.com/articles/the-key-design-elements-of-roguelikes--cms-23510>. Acesso em: 11 jun. 2019.

ROGERS, Scott. **Level Up:** UM GUIA PARA O DESIGN DE GRANDES JOGOS. São Paulo: Editora Edgard Blücher Ltda., 2013.

ROUSE III, Richard. **Game Design - Theory and Practice:** The Elements of Gameplay. **gamasutra**, Estado, 27 jun. 2001. Disponível em: <https://www.gamasutra.com/view/feature/131472/game_design__theory_and_practice_.php>. Acesso em: 11 jun. 2019.

THOMAS, Frank; JOHNSTON, Ollie. **Disney Animation:** The Illusion of Life. Cidade, Disney Editions, 1995

WAWRO, Alex. **Warren Robinett reflects on his ‘signature’ game: *Adventure***. **gamasutra**, Estado, 5 março 2015. Disponível em: <https://www.gamasutra.com/view/news/238159/Warren_Robinett_reflects_on_his_signature_game_Adventure.php>. Acesso em: 11 jun. 2019.

WILLIAMS, Richard. **Manual de Animação**. São Paulo: Editora Senac São Paulo, 2016.