CENTRO UNIVERSITÁRIO CARIOCA

LEONARDO PEREIRA THURLER

MICHELE DA SILVA ANTONIO

DESENVOLVIMENTO DO JOGO MONSTER CHEF:

DO CONCEITO AO PRODUTO

RIO DE JANEIRO, RJ

2017

LEONARDO PEREIRA THURLER

MICHELE DA SILVA ANTONIO

DESENVOLVIMENTO DO JOGO MONSTER CHEF:

DO CONCEITO AO PRODUTO

Dissertação apresentada ao Programa de Pós-Graduação em Jogos e Animação Digital do Centro Universitário Carioca – UniCarioca como parte dos requisitos necessários para a obtenção do grau de Especialista em Jogos e Animação Digital.

Orientador: Prof. André Cotelli do Espírito Santo

RIO DE JANEIRO, RJ

2017

RESUMO

Apesar do mercado de jogos ser bem consolidado e reconhecido mundialmente, ainda existem muitas dúvidas em relação ao processo de produção de um jogo. Essas dúvidas envolvem tanto questões mais genéricas como: "Como transformar uma ideia em um jogo?", até questões mais específicas como: "Qual ferramenta utilizar ao produzir uma animação de um personagem?". Tais perguntas têm se originado devido ao crescimento no número de pessoas interessadas a produzir jogos e que acabam buscando conteúdos sobre como iniciar ou se aprofundar na área de produção de jogos. Com o aumento dessa busca, foram surgindo cada vez mais conceitos e ferramentas que auxiliam na construção de jogos. Tanto esses conceitos quanto essas ferramentas foram se desenvolvendo, atendendo a demanda de novos tipos de jogos e necessidades encontradas pelos desenvolvedores. Com isso, é possível encontrar ferramentas que auxiliam na parte da gestão do projeto, na programação, na criação dos elementos visuais do jogo, roteiros, som e de vários outros aspectos que o envolvem. Nesse sentido, o presente trabalho tem por objetivo construir um jogo em duas dimensões (2D) para plataforma mobile, apresentando as metodologias, conceitos e ferramentas utilizadas além de descrever o uso destas, de forma a exemplificar os ganhos obtidos ao utilizá-las. Para isto foi desenvolvida a versão de demonstração do jogo Monster Chef, e, através das etapas descritas nesta dissertação, é possível observar como funciona todo o processo de desenvolvimento, que vai desde a documentação da ideia até a parte final da produção do jogo. Além disso, é relatado como as ferramentas foram utilizadas demonstrando o ganho obtido e a importância delas para que seja alcançado um resultado final de qualidade.

Palavras-chave: jogos digitais, jogos 2D, mobile, game design, produção de jogos.

ABSTRACT

Although the game market is well established and worldwide recognized, there are still many questions regarding the process of producing a game. These doubts involve more generic questions such as: "How can I turn an idea into a game?", to more specific issues such as "What tool should I use to produce an animation of a character?". Such questions have originated due to the growth of people numbers interested to produce games and who end up seeking knowledge about how start or how specialize in the game production area. With the increase of this search, were arise increasingly concepts and tools that help in the game building. Both concepts and tools were developing, attending the demand of new game types and the necessity found by developers. Thereby, it's possible found tools that help with project management, programmation, produce visual elements for the game, scripts, sounds and many others aspect that involve the project. In this sense, the present work aims to make a game in two dimensions (2D) for the mobile platform, presenting the methodologies, concepts and tools used in addition to describing the use of these, in order to exemplify the gains made in using them. For this was developed the demonstration version of Monster Chef game, and, through steps described in this dissertation , is possible see how works all development process, ranging from idea documentation to the production final part of the game. Beyond that, is reported how the tools were used demonstrating the gain obtained and their importance in achieving a final quality result.

Keywords: digital games, 2D games, mobile, game design, game development.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO

1. INTRODUÇÃO

A indústria de jogos é uma das mais rentáveis no mundo (G1, 2013; UOL 2015), atualmente há empresas que criam jogos e faturam apenas no dia de lançamento, sem contar os outros dias de vendas, milhões de dólares. Devido ao desenvolvimento e o sucesso dessa indústria o número de pessoas interessadas em produzir jogos tem aumentado de forma considerável a cada dia. Esse aumento aliado ao desenvolvimento da tecnologia tem feito com que os jogos estejam cada vez mais presentes na vida das pessoas. Basta sairmos de nossa casa, para percebermos que não é difícil encontrar pessoas jogando em seus smartphones e tablets.

Devido a essa alta procura, os jogos tem evoluído e se adaptado para atender a todo tipo de pessoas e as diferentes tecnologias. Há jogos que rodam somente em computadores mais robustos que tenham um alto poder de processamento, mas também há aqueles que são capazes de rodar em um smartwatch (relógio inteligente) como pode ser observado na notícia publicada pela (TECHTUDO, 2016). Além dessas diferenças, os objetivos dos jogos também tem se tornado bem variados, enquanto alguns buscam entreter as pessoas, outros tem como objetivos a educação ou efeitos terapêuticos, ajudando no tratamento de pessoas.

Dada toda essa variedade de jogos, é de se esperar que cada um deles tenha o seu próprio processo e forma de desenvolvimento, uma vez que um jogo direcionado ao tratamento de pessoas com câncer terá particularidades que um jogo voltado para ensinar crianças a escreverem por exemplo, não teria. Devido ao crescimento da indústria de jogos, o processo de produção foi sendo refinado com o passar do tempo. Hoje é possível afirmar que existem algumas características que estarão presentes nos projetos jogos e com base nessas características foram definidos os tipos de profissionais necessários para o desenvolvimento de jogos como: artistas, programadores, game designers, músicos, roteiristas e diversos outros.

A maior parte dos jogos mais conhecidos são desenvolvidos por uma equipe com dezenas ou centenas de pessoas e que geralmente trabalham em uma empresa de jogos há muitos anos, esses jogos são chamados de triple A ou AAA. Porém existem jogos de grande sucesso que foram produzidos por uma pessoa ou uma equipe de poucas pessoas. Esses projetos geralmente são produzidos sem investimento e muitas das vezes por pessoas que não trabalham ativamente na indústria de games. Esse jogos são chamados de independentes ou indies. Dado esse cenário a pergunta que fica no ar é “Como um jogo indie pode fazer sucesso em meio a tantos jogos AAA?”. A resposta a essa pergunta nos leva a uma breve análise do mercado de jogos. Geralmente os jogos AAA tem o problema de não inovarem e se tornarem muito parecidos uns com os outros, isso faz com que o mercado acabe ficando saturado de jogos que parecem cópias ou que oferecem a mesma sensação ao jogar. Já os jogos indies, nesse ponto, costumam se destacar, em geral, os jogos indies de sucesso buscam inovar e trazer uma experiência diferente das encontradas nos jogos AAA e dessa maneira eles conseguem um espaço no mercado conforme pode ser constatado pela notícia da (TECHTUDO, 2012).

Apesar de toda a evolução da indústria e do processo, o desenvolvimento de um jogo, mesmo que seja um jogo indie não é algo simples de ser realizado e gerenciado. Ainda há muitas dúvidas em relação ao processo de produção, essas dúvidas são originadas tanto por falta de conhecimento quanto por diversidade de ferramentas. Devido a variedade de tipos de jogos e ferramentas para desenvolvimento não é possível afirmar qual é a melhor forma de produzir um jogo, porém é possível apresentar e analisar alguns conceitos e ferramentas que têm se tornado eficiente para a criação de diversos tipos de jogos.

Neste contexto, é proposto neste trabalho a documentação do processo de produção da versão de demonstração, ou demo, do jogo Monster Chef, de forma a demonstrar os conceitos e ferramentas utilizadas no desenvolvimento desse jogo independente.

1.1. OBJETIVOS

1.1.1. OBJETIVO GERAL

O objetivo desta dissertação é construir um jogo em duas dimensões (2D), apresentando as metodologias, conceitos, ferramentas utilizadas e descrevendo o uso destas de forma a exemplificar os ganhos obtidos ao utilizá-las.

1.1.2. OBJETIVOS ESPECÍFICOS

- Produzir a versão demo deste jogo 2D para a plataforma mobile de forma independente;
- Validar a proposta do jogo e verificar a viabilidade de produzi-lo por completo;
- Gerenciar o processo de produção para que se torne mais fácil de administrar as tarefas e ter uma visão geral do jogo; e
- Descrever o uso das ferramentas exemplificando os ganhos obtidos através da utilização delas.

CAPÍTULO

2. FERRAMENTAS E CONCEITOS

2.1. CONCEITOS

2.1.1. METODOLOGIA ÁGIL

De acordo com (BERNADO, 2015), a metodologia ágil surgiu para ser uma alternativa às gestões tradicionais de projetos. Essa metodologia surgiu inicialmente na área de desenvolvimento de software e devido a sua eficiência. Atualmente ela já é utilizada em diversas outras áreas.

Essa metodologia busca gerenciar os projetos de uma forma que incentiva a validação e mudanças frequentes à medida que o projeto avança. Isso funciona como uma filosofia que tem como foco o melhorar o trabalho em equipe, a comunicação frequente, o foco no cliente e a entrega de valor. Sendo assim, essa metodologia possui 4 valores principais que são:

- Indivíduos e interação entre eles mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

A equipe precisa ter esses valores em mente e buscar organizar o seu trabalho de uma forma que priorize a utilização deles. A utilização na produção de projetos tem se demonstrado muito eficiente em diversas áreas e por isso, ela tem se tornado cada vez mais conhecida e mais utilizada por profissionais.

2.1.2. ANIMAÇÃO 2D ATRAVÉS DE BONES

Durante muito tempo as animações de imagens 2D eram realizadas somente por técnicas que consistem em refazer o desenho em várias poses diferentes e

realizar a troca dessas imagens a cada frame do vídeo. Com o passar do tempo foram surgindo outras formas de animar esse tipo de imagem e uma delas é a animação através de bones.

Nesse tipo de animação não é necessário refazer o desenho a cada frame, ao invés disso os elementos do desenho são rotacionados, movimentados, esticados e achatados para realizar a animação. Diferente da animação frame a frame, onde todos os elementos ficam na mesma imagem, nessa cada elemento da arte precisa ser separado em arquivos diferentes e esses elementos devem estar desenhados completamente, mesmo que na cena exista algo que fique em frente a uma parte desse elemento.

2.1.3. SISTEMA DE CONTROLE DE VERSÃO

Conforme a publicação do site (GIT, 2017), um sistema de controle de versão serve para registrar todas as mudanças realizadas em um ou mais arquivos, de maneira que as informações possam ser recuperadas posteriormente. Esse tipo de sistema permite reverter arquivos e projetos inteiros para um estado anterior caso tenha ocorrido algum problema no estado atual, comparar as mudanças feitas ao decorrer do tempo em cada arquivo, verificar qual foi o último integrante a editar um determinado arquivo, controlar versões do projeto que estejam estáveis e muitas outras coisas. Assim sendo, utilizar esse tipo de sistema fornece uma maior segurança e facilidade no gerenciamento dos arquivos do projeto.

2.1.4. GAME DESIGN DOCUMENT (GDD)

A produção de um jogo envolve profissionais de muitas áreas de conhecimento, como artistas, músicos, programadores, game designers, level designers e diversas outras especialidades. Com base nessa diversidade de áreas envolvidas, é comum que os jogos sejam produzidos por uma equipe e não somente por uma pessoa. É claro que sempre irá existir a exceção e poderemos encontrar jogos que foram produzidos apenas por uma pessoa. Porém, mesmo sendo desenvolvido por uma equipe, um grupo pequeno ou apenas uma pessoa, o projeto

pode ser demorado e fazer com que alguns detalhes se percam com o passar do tempo ou ele pode ser tão grande a ponto de fazer com que as pessoas que o idealizaram não tenham todos os detalhes em mente no primeiro momento. Além disso, é comum que parte da equipe não seja envolvida em todo o processo criativo do jogo. Sendo assim, surgiu a necessidade da criação de um documento que representasse uma visão do jogo e de todos os seus elementos em que toda a equipe possa utilizá-lo durante o processo de produção, definindo uma mesma visão do projeto como um todo.

De acordo com (SCOTT, 2012 p. 96), “Um GDD esboça tudo o que estará no jogo. É um documento muito importante ao qual toda a equipe se referenciará durante a produção de seu jogo.”. Desse modo, podemos ter uma ideia da importância e do conteúdo deste documento. É muito importante ressaltar que apesar do GDD precisar esboçar tudo o que estará no jogo, deve-se tomar cuidado para não colocar informações desnecessárias ou vagas demais. Esse documento precisa ser direto e útil para que as pessoas leiam e entendam o seu conteúdo.

Um dos grandes empecilhos que faz com que muitas equipes pequenas optem por não utilizar um GDD em seus projetos é o esforço e o tempo necessário para redigir um documento com tantos detalhes e páginas e, que possivelmente, não será muito utilizado visto que a equipe está trabalhando sempre no mesmo ambiente. Apesar do esforço e tempo citados, o GDD é um documento importante mesmo em jogos pequenos, que serve como um guia visando não permitir que os desenvolvedores se afastem ou esqueçam o objetivo e características principais do jogo. Caso os desenvolvedores não tenham esse cuidado, os projetos podem nunca serem terminados ou se tornarem um misto de coisas que foram agrupadas e que não fazem muito sentido. Para minimizar esse problema, a não utilização de um documento no processo de desenvolvimento, os GDDs foram se adaptando de forma a ter modelos de GDD de apenas uma página, que segundo (SCOTT, 2012 p. 84) “O página-única é uma visão global do seu jogo.”; o de dez páginas que ainda segundo (SCOTT, 2012 p. 86) “A intenção é fazer com que os leitores entendam o básico do produto final, sem entrar em detalhes excruciantes” e modelos maiores

que englobam diferentes tipos de características do seu jogo. Assim, as equipes podem produzir um documento que melhor se adeque a sua necessidade.

Outro ponto importante a ser levado em conta é que o GDD é um documento mutável, ele pode ser atualizado e ajustado à medida que a equipe for desenvolvendo e conhecendo mais o seu próprio jogo.

2.1.5. WIREFRAME

De acordo com (PEREIRA, 2008), o wireframe é utilizado em diversas áreas para representar um guia visual que visa mostrar de maneira direta, como será um determinado objeto de acordo com suas especificações. Esse objeto pode ser um site, um aplicativo, um modelo de carro, entre outros.

Para a produção do jogo desta dissertação o wireframe será utilizado do mesmo modo que é utilizado pela área de design interfaces. Nessa área o wireframe serve para exemplificar o fluxo de interações do usuário com a interface que está sendo desenvolvida e algumas vezes representar um exemplo da disposição dos elementos da interface na tela. O intuito do wireframe é validar se a ideia do designer está coerente com o que é necessário e verificar se há algum problema na usabilidade ou disposição dos elementos.

Um wireframe não deve ser tratado como o layout final do projeto, ele deve ser produzido de forma rápida visando concentrar os esforços na validação do fluxo e não na aparência do layout. Atualmente essa é uma das maneiras mais eficazes e baratas de validar o funcionamento do design de interface de sua aplicação.

2.1.6. TILESET

Tileset é uma técnica que consiste na criação de uma imagem que contém um mapa com diversos elementos, que servem para criar o visual de um cenário ou lugar específico de um jogo. No início do desenvolvimento de jogos, os computadores e videogames possuem hardwares muito limitados comparados aos atuais, devido a isso, os primeiros jogos que foram criados não possuíam muitos

elementos gráficos.

Apesar dessa limitação, os desenvolvedores sentiram a necessidade da criação de jogos cada vez mais elaborados e com mais elementos gráficos para representar melhor os cenários. Mas devido essa limitação de hardware, eles acabaram desenvolvendo uma técnica para fazer com que os aparelhos pudessem ter um arquivo de imagem menor e a partir desse arquivo, gerar um cenário muito maior do que seria possível caso utilizassem uma imagem única contendo esse cenário.

Para isso, eles criaram o que hoje é conhecido como tileset. Cada elemento que compõe o cenário é separado e colocado dentro de uma imagem única. Sobre essa imagem é aplicada uma matriz que basicamente “recorta” a imagem em vários pedaços e cada pedaço desse pode ser acessado individualmente através dessa matriz. Dessa forma, eles criavam todo o cenário através dos valores da matriz.

Essa técnica se mostrou tão eficiente que mesmo com toda a evolução que teve durante todos os anos da indústria de jogos, ainda é muito utilizada na criação de jogos 2D por facilitar a criação de mapas e detecção de colisões.

2.2. FERRAMENTAS UTILIZADAS

2.2.1. GAME ENGINE

As game engines são programas construídos com o objetivo de facilitar e agilizar o desenvolvimento de jogos eletrônicos. Assim sendo, esses programas buscam disponibilizar, para os desenvolvedores que os utilizarem, diversos tipos de bibliotecas e funções que são comuns a esses tipos de projetos. Geralmente os recursos mais comuns disponibilizados por esses softwares são métodos para a detecção de colisões entre os elementos do jogo, um sistema para facilitar a simulação de física e principalmente uma parte que permite a renderização de gráficos em tempo real de um modo bem otimizado e simples de ser utilizado. Além

desses recursos principais, as game engines costumam oferecer diversos outros recursos adicionais.

Com o passar do tempo e o feedback dos desenvolvedores as game engines mais utilizadas e famosas foram se tornando um ambiente completo e ágil para o desenvolvimento de jogos. Elas passaram a possuir ferramentas para edição de cenários, interpretação de scripts de programação, reprodução de áudio, suporte a animações, integração com redes sociais e muitas outras funções que se tornaram comuns na produção de jogos. Além de facilitar o desenvolvimento do jogo em si as game engines passaram a fornecer outras facilidades aos desenvolvedores, elas adotaram algumas linguagens de programação e formas de desenvolvimento próprios da engine visando a automatização de recursos de gerenciamento de memória e permitindo os desenvolvedores a trabalharem com linguagens de alto nível como C# e Javascript. Além disso, ao utilizar essas linguagens, as próprias engines permitem que o projeto possa ser exportados para diversas plataformas sem a necessidade de alteração de código e ajustes, sendo assim elas também ampliam as possibilidades de alcance dos jogos produzidos, pois o jogo pode rodar em um computador e em um celular sem a necessidade da equipe modificar o projeto.

Atualmente existem diversos tipos de game engines disponíveis no mercado, algumas são focadas em jogos 3D, outras em jogos 2D e outras fornecem suporte a jogos textuais. As diferenças das game engines não param só no tipo de jogos que elas dão suporte mas se estendem para os preços de utilização dos serviços, algumas são gratuitas e outras pagas. Não é possível selecionar e dizer que existe uma engine melhor do que a outra, isso varia de acordo com a necessidade do projeto e da equipe que esteja desenvolvendo o jogo.

2.2.1.1. UNITY 3D

A Unity 3D é uma das game engines mais famosas e utilizadas atualmente pelos desenvolvedores de jogos do mundo inteiro, ela oferece suporte a criação de jogos 2D e 3D fornecendo bibliotecas com funções para simulação de física nos

ambientes e personagens, iluminação, áudio interativo, inputs de comandos através de teclado, mouse e telas de touch screen, suporte a exportação do jogo para computadores, videogames e celulares e diversas outras funções. Devido a grande atividade da comunidade de desenvolvedores e a rápida curva de aprendizagem para a criação de projetos, esta ferramenta foi escolhida para a produção do jogo proposto nesta dissertação.

Essa game engine permite a exportação do jogo para as principais plataformas disponíveis no mercado atualmente como IOS, Android, Windows, Mac, Linux, Facebook Gameroom, Playstation, Xbox e outras. Devido a essa característica o desenvolvedor não precisa se preocupar com as características específicas de cada plataforma ao desenvolver o jogo, a engine também fornece um suporte para identificação e compilação de parte do código que será executado somente na plataforma desejada dando assim a flexibilidade necessária para o desenvolvedor utilizar os recursos específicos das plataformas se assim desejar. O Unity ainda oferece suporte a duas linguagens de programação para a criação de seus scripts, essas linguagens são: JavaScript e C#. O desenvolvedor pode escolher a linguagem que mais domina para produzir o seu projeto.

A Unity 3D atualmente possui duas formas de licenciamento, uma versão gratuita e outra por assinatura mensal. A versão gratuita pode ser utilizada tanto por desenvolvedores quanto por empresas, desde que esses ainda não tenham atingido um rendimento anual de 100 mil dólares caso a equipe atinja esse rendimento ela deverá pagar uma assinatura da Unity. A versão de assinatura possui mais recursos e menos limitações que a versão gratuita e além disso ela possui um suporte da própria empresa que produz a Unity, o valor dessa versão varia de 32 a 115 euros mensais dependendo do rendimento anual da empresa que está contratando.

2.2.2. PHOTOSHOP

O Adobe Photoshop é o software de edição de imagens bidimensionais, em pixels mais conhecido do mercado, desenvolvido pela Adobe Systems. Líder no mercado dos editores de imagem profissionais, é utilizado também para a

manipulação de imagens digitais e pré-impressão. Permite a construção e criação de imagens a partir de layers, podendo também ser a base de pintura de ilustrações, sendo utilizado juntamente com uma mesa digitalizadora, que é indicada para obter o mesmo resultado do uso de um lápis.

Tem como principal característica o uso de camadas (layers), o que não foi um recurso inventado pelos desenvolvedores do Photoshop e sim, por ilustradores. Essa característica foi uma grande contribuição para a utilização do software na animação tradicional, pois por meio das camadas de edição, pode-se recriar a técnica de desenho sobre folhas de acetato, permitindo criar o movimento do personagem através da modificação do desenho base, usando a camada em transparência.

O Adobe Photoshop possui uma versão de avaliação gratuita que funciona por trinta dias disponível na versão CS6 ou pode ser adquirido por meio de assinatura no site da Adobe, no valor de 105 reais ao mês, ou anual, no valor de 852 reais.

2.2.3. FLASH

Originalmente desenvolvido pela Macromedia, hoje pertence à Adobe. O Flash Professional é um software de gráfico vetorial utilizado para a criação de animações interativas, que podem funcionar em um navegador web, em desktops, tablets e televisores.

As ilustrações criadas no Flash são feitas de forma vetorial, ou seja, utilizando primitivas geométricas como pontos, linhas, curvas, formas ou polígonos. O vetor não perde a resolução caso seja aumentado de tamanho.

As animações são feitas por meio de uma linha de tempo, ou seja, são criadas a partir de um ponto e ao serem mostradas em sucessão, dá-se a impressão de movimento.

Assim como o Adobe Photoshop, o Flash Professional também possui uma versão de avaliação com duração de trinta dias e pode ser adquirido na loja online da Adobe no valor de 105 reais ao mês ou anual, no valor de 852 reais.

2.2.4. ILLUSTRATOR

O Adobe Illustrator é o software da Adobe Systems voltado para a criação de ilustrações, gráficos digitais e tipografia em diversos tipos de mídias, como web, vídeo e móvel. As ilustrações feitas no Illustrator usam o mesmo sistema de camada. Assim como no Photoshop, pode-se criar cenários separados dos objetos e personagens.

As ilustrações e gráficos são criados de forma vetorial, ou seja, independem da resolução e podem ser aumentadas infinitamente, sem perda de qualidade.

Assim como o Photoshop e o Flash, o Illustrator também possui uma versão de teste gratuita ou pode ser adquirido na loja online da Adobe, com os mesmos valores dos citados anteriormente.

2.2.5. SPRITER

O Spriter é um programa que permite a construção de animações 2D baseado em bones. Esse programa possui uma linha de tempo para construir suas animações e permite a criação de diversos bones, construindo assim um “esqueleto”, que serve para representar as ligações entre cada elemento da imagem. Após criar esse esqueleto é possível associar uma ou mais imagens a cada bone que o compõe, e após essa associação, as imagens irão acompanhar as transformações que forem realizadas em cada bone. Então utilizando a linha de tempo e realizando as transformações nos bones é possível criar uma animação de forma bem prática e rápida.

As animações geradas a partir deste programa podem ser exportadas para gif, ou para diversas imagens a representar cada frame da animação. Além disso este software possui uma comunidade muito grande e ativa, então ele conta com

diversos plugins não oficiais que permitem a integração da animação gerada com diversos outros programas como o Unity 3D, e isso traz uma diferença significante por otimizar as animações e o processo de desenvolvimento das mesma.

O Spriter possui uma versão gratuita e outra paga, a versão gratuita pode ser utilizada em qualquer tipo de projeto a diferença é que ela possui menos recursos que a versão paga que custa em torno de 60 dólares.

2.2.6. HACKNPLAN

O Hacknplan é um serviço de gerenciamento de projeto baseado em nuvem e que tem como foco fornecer recursos otimizados para projetos de jogos. Esses recursos permitem que a equipe possa planejar, documentar, agendar, rastrear e avaliar todos os detalhes de um projeto de jogo. No mercado já há serviços similares como o Jira e o Trello, porém não há muitos que sejam focados em projetos de jogos e que tenham os recursos que ele possui.

Uma das principais características e que o diferencia bastante dos concorrentes é o conceito de Game Design Model, que permite basicamente a criação de uma estrutura de GDD dentro da ferramenta e faz com que as tarefas que precisam ser desenvolvidas fiquem atrelados a alguma parte dessa estrutura. Assim, fica mais claro para a equipe em qual parte do GDD está o conteúdo referente aquela tarefa, e ainda permite um controle sobre quais mecânicas e recursos já foram produzidos ou não para o jogo final.

O Hacknplan possui tanto uma versão gratuita quanto uma versão paga. A versão paga possui diversas ferramentas a mais como integração com ferramentas de controles de versão, integração com o calendário do google entre outras. A versão gratuita, mesmo limitada, disponibiliza a maior parte das ferramentas para acompanhamento do projeto e também disponibiliza o Game Design Model, fazendo com que ela ainda seja uma boa opção.

2.2.7. ASSEMBLA

O Assembla é um site que disponibiliza repositórios para se utilizar sistemas de controles de versões. Através dele é possível criar uma conta e criar um repositório para hospedar os arquivos do seu projeto e se beneficiar de todos os benefícios de um sistema de controle de versão. Além disso, os arquivos do seu projeto são privados e somente quem você convidar terá acesso a esses códigos, garantindo assim uma segurança ao hospedar o seu código no repositório disponibilizado.

Ele possui tanto uma versão paga quanto uma versão gratuita. A versão gratuita tem limite de armazenamento de 500MB e caso seja necessário armazenar mais do que isso, será preciso assinar a versão paga. A versão paga custa em torno de 75 dólares por mês, com o limite de 10 usuários no projeto. Existem versões com preços que variam de acordo com o número de usuários no projeto.

2.2.7. TILED

Tiled é um software que facilita a criação de mapas utilizando um ou mais tilesets para criar o mapa final. Através dele, é possível criar um arquivo que representa o mapa que será produzido, e, além disso, ele possui uma interface gráfica que permite selecionar os elementos do tileset e posicionar esses elementos sobre o mapa que está sendo criado.

Ele permite exportar o arquivo para diversos tipos de formatos diferentes, esses formatos variam desde arquivos de texto contendo o mapeamento da matriz com os elementos posicionados, até imagens contendo a aparência do cenário. A maior parte das game engines atuais já possuem algum recurso para reconhecer um ou mais desses tipos que são exportados e dessa forma elas conseguem importar o cenário para ser usado no jogo.

O Tiled é um software totalmente gratuito e desenvolvido por uma grupo aberto de pessoas que buscam aprimorar a ferramenta de acordo com os pedidos e necessidades da comunidade que o utiliza.

2.2.8. NINJAMOCK

Ninjamock é um software acessado através do navegador que permite a produção de wireframes e mockup de telas para diversas plataformas. Este tipo de ferramenta costuma ser muito utilizada na produção de aplicativos e sites e ajuda a validar o fluxo das navegação do usuário no aplicativo sem precisar fazer as telas de fato.

A ferramenta possui diversos componentes que são comuns aos aplicativos de celulares como: botões, imagens, ícones, janelas de diálogo, lista de opções e outras coisas. Para utilizar esses componentes basta arrastar e soltá-los para a tela, posicionando e editando os texto conforme necessário. Além dos componentes padrões a ferramenta permite que o usuário faça upload das próprias imagens criando novos componentes conforme sentir necessidade, isso a torna uma ferramenta bem versátil e útil.

O Ninjamock possui tanto uma versão gratuita quanto uma versão paga, a diferença está no fato de que os projetos da versão gratuita sempre serão públicos enquanto a versão paga oferece a possibilidade de criar projetos privados e o valor de assinatura depende do número de projetos privados que o usuário deseja utilizar.

CAPÍTULO

3. DESENVOLVIMENTO DO PROJETO

3.1. GESTÃO DO PROJETO

3.1.1 ORGANIZAÇÃO DAS TAREFAS

Para realizar o projeto de maneira mais organizada foi utilizado os princípios da metodologia ágil para desenvolvimento de softwares em conjunto com o Hacknplan. Primeiro, foi criada uma base para o GDD e essa base foi colocada no Game Design Model do Hacknplan como pode ser visto na figura a seguir.

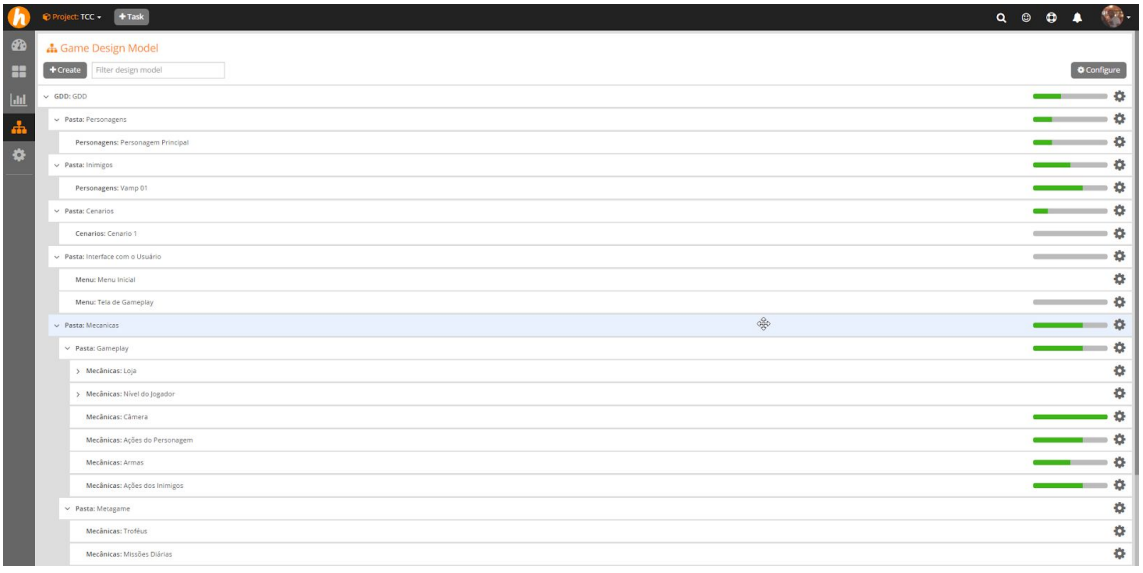


Figura 1 - Game Design Model no Hacknplan

Após colocar essa estrutura no Hacknplan foram realizadas algumas reuniões periódicas para verificar quais tarefas já haviam sido realizadas, quais estavam impedidas por conta de algum problema e quais seriam as próximas tarefas que

teriam que ser realizadas. Durante essas reuniões eram criadas as tasks no Hacknplan. Elas eram separadas por tipo para que fosse mais fácil de identificar quais tinham relação com programação, arte, game design e outras. Além disso elas eram categorizadas em alguma parte do Game Design Model para saber com o que aquela tarefa tinha relação.

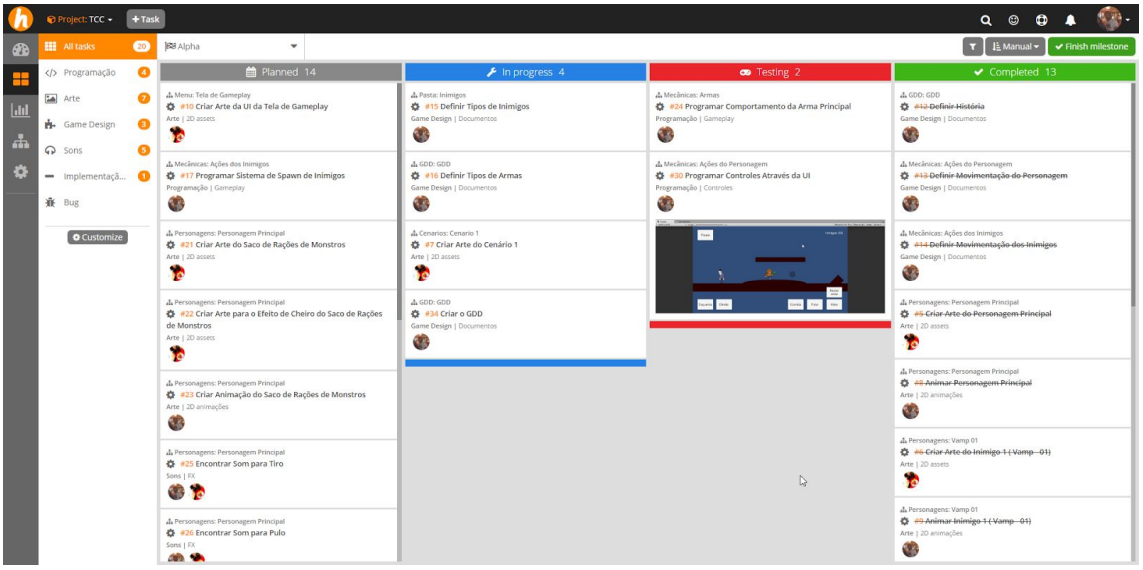


Figura 2 - Kanban do Hacknplan com as tasks representando as tarefas

Conforme pode ser visto na figura 2 as tasks serviram para representar cada tarefa e o estado em que ela se encontrava naquele momento, isso permitia um controle maior da situação atual do projeto.

3.1.2 GERENCIAMENTO DOS ARQUIVOS

Para realizar o gerenciamento do armazenamento e edição dos arquivos foi utilizado o Assembla. Nele criamos um repositório utilizando o SVN, que é um sistema de controle de versão e, com isso, conseguimos controlar todas as edições dos arquivos no projeto. A figura abaixo apresenta a tela do Assembla com algumas das pastas que foram versionadas no projeto.

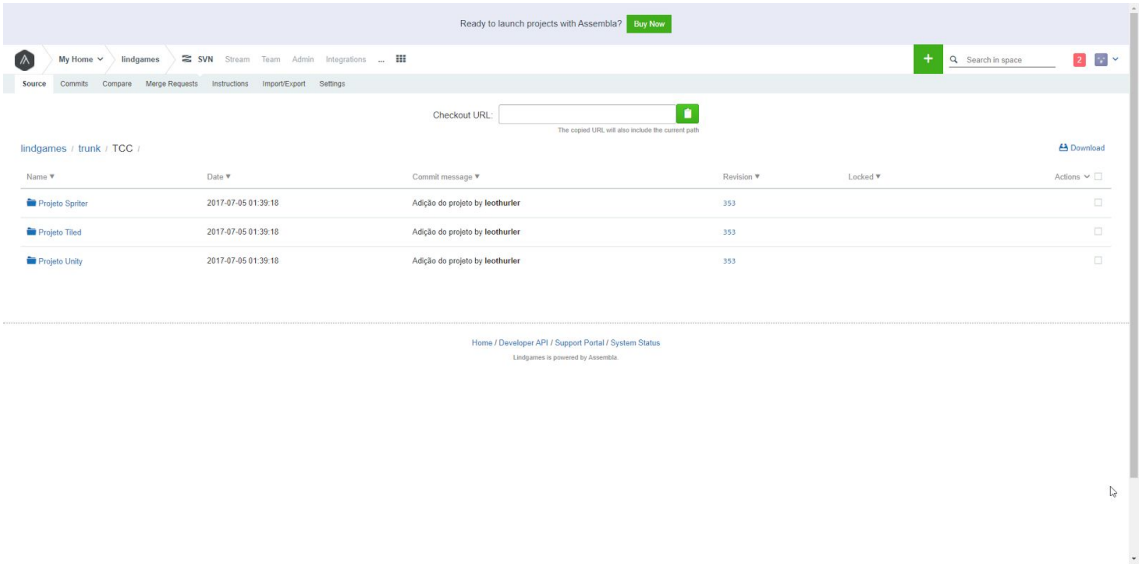


Figura 3 - Tela do Assembla com as pastas versionadas

3.2. GAME DESIGN

3.2.1. GDD

Para documentar a ideia do jogo foi criado um GDD, contendo as informações de cada parte que precisa ser construída, para assim produzir o jogo.

3.2.1.1. INTRODUÇÃO

O Monster Chef é um jogo de aventura e simulação que será desenvolvido para as plataformas Android e IOS. Nele o jogador irá assumir o papel de Beowolf um lobisomem que está montando uma hamburgueria gourmet.

3.2.1.2. HISTÓRIA

O Jogo se passa em um mundo fantasioso habitado por vários tipos de monstros. Nesse mundo o jogador assume o papel de Beowolf um Lobisomem que está montando uma hamburgueria gourmet, que terá especiarias como o delicioso X-Vamp Burguer ou o MC-Vamp Feliz, sanduíche que tem o dente do vampiro morto

como brinde para o seus lobinhos. Mas para que a hamburgueria tenha essas especiarias, é preciso coragem e determinação para ir atrás dos diversos monstros, atrás de ingredientes para os seus produtos e, como se isso não fosse complicado o suficiente Beowolf precisará mostrar que entende bem de vendas e de como agradar seus clientes para atingir o objetivo de ter a hamburgueria mais famosa do mundo.

3.2.1.3. GÊNERO

Monster Chef possui uma mistura de jogos de ação, aventura e simulação 2D. Uma vez que o jogador terá que evoluir o seu personagem permitindo assim que ele vá aos mais diversos cenários para capturar os monstros, com o intuito de conseguir ingredientes para os produtos da sua loja e com esses ingredientes, ele conseguirá produzir e vender os produtos e dessa forma conseguindo dinheiro para melhorar seus equipamentos e atingir o seu objetivo de ter a hamburgueria mais famosa do mundo.

3.2.1.4. PLATAFORMAS

O jogo será desenvolvido para dispositivos móveis como Android e IOS.

3.2.1.5. CARACTERÍSTICAS PRINCIPAIS

- Mobile;
- Single-player;
- Gráficos 2D (duas dimensões);
- Jogabilidade através de toque na tela;
- Cenários e monstros fantasiosos;
- Sistemas de rank, conquista e missões;
- Sistemas de skins e equipamentos; e
- Propaganda e vendas no jogo.

3.2.1.6. PERSONAGENS

3.2.1.6.1. JOGÁVEIS

- Beowolf.

3.2.1.6.2. PERSONAGENS SECUNDÁRIOS

- Frank; e
- Personagens clientes da loja (lobisomens).

3.2.1.6.3. INIMIGOS

O jogo possuirá diversos tipos de monstros, abaixo será apresentada uma lista com os tipos de monstros e um resumo das características de cada um deles.

- Vampiros;
    - Vampiro-01 - normal, 1hp, foge.
    - Vampiro-02 - lento, 2hp, foge.
    - Vampiro-03 - normal, 1hp, ataca e foge.
    - Vampiro-04 - normal, 2hp, ataca e foge.
    - Vampiro-05 - rápido, 1hp, foge.
    - Boss.
- Zumbis;
    - Zumbi-01 - normal, 1hp, foge.
    - Zumbi-02 - lento, 2hp, foge.
    - Zumbi-03 - normal, 1hp, ataca e foge.
    - Zumbi-04 - normal, 2hp, ataca e foge.
    - Zumbi-05 - rápido, 1hp, foge.

    - Boss.
- Fantasmas;
    - Fantasma-01 - normal, 1hp, foge.
    - Fantasma-02 - lento, 2hp, foge.
    - Fantasma-03 - normal, 1hp, ataca e foge.
    - Fantasma-04 - normal, 2hp, ataca e foge.
    - Fantasma-05 - rápido, 1hp, foge.
    - Boss.
- Bruxas; e
    - Bruxa-01 - normal, 1hp, foge.
    - Bruxa-02 - lento, 2hp, foge.
    - Bruxa-03 - normal, 1hp, ataca e foge.
    - Bruxa-04 - normal, 2hp, ataca e foge.
    - Bruxa-05 - rápido, 1hp, foge.
    - Boss.
- Mûmias.
    - Múmia-01 - normal, 1hp, foge.
    - Múmia-02 - lento, 2hp, foge.
    - Múmia-03 - normal, 1hp, ataca e foge.
    - Múmia-04 - normal, 2hp, ataca e foge.
    - Múmia-05 - rápido, 1hp, foge.
    - Boss .

3.2.1.7. CENÁRIOS

O jogador deverá caçar monstros para coletar os ingredientes do seus produtos. Essa parte de caçadas será dividida em diversos cenários que são habitados por diferentes tipos de monstros, a lista abaixo representa os cenários e os monstros que serão encontrados em cada cenário do jogo.

- Cemitério (vampiros);
- Pântano (zumbis);
- Casa Abandonada (fantasmas);
- Floresta (bruxas); e
- Pirâmide (mûmias).

3.2.1.8. INTERFACE COM O USUÁRIO

3.2.1.8.1 FLUXO DE TELAS

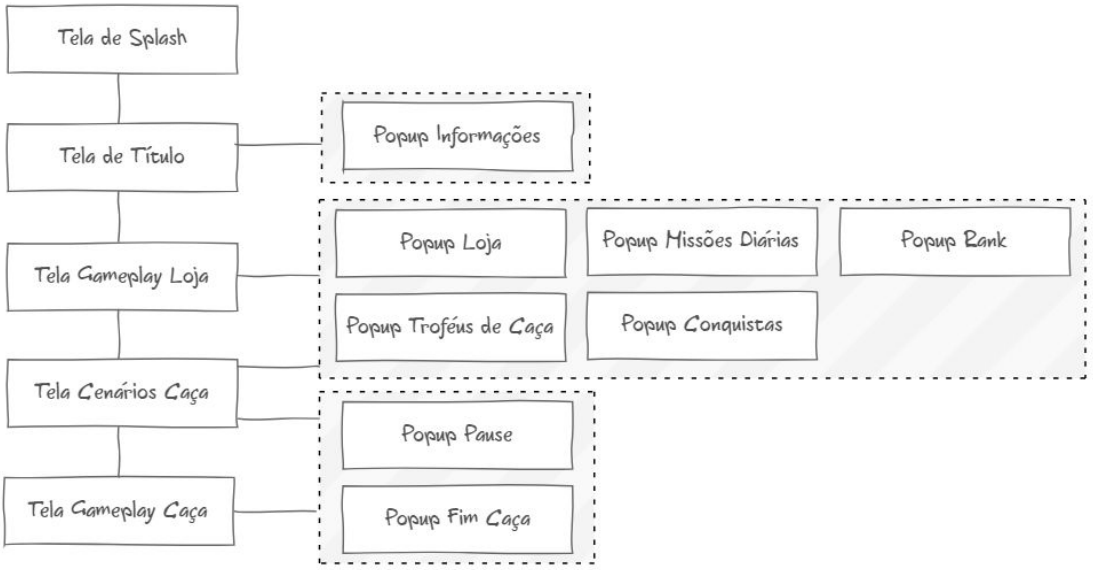


Figura 4 - Fluxo de telas do jogo

A figura 4 apresenta o fluxo de telas do jogo. Podemos observar que o fluxo se inicia pela “Tela de Splash” e vai até a “Tela Gameplay Caça”, esse fluxo serve

como um mapa que demonstra a partir de quais telas o jogador terá acesso a cada função do jogo.

3.2.1.8.2 WIREFRAME DAS TELAS

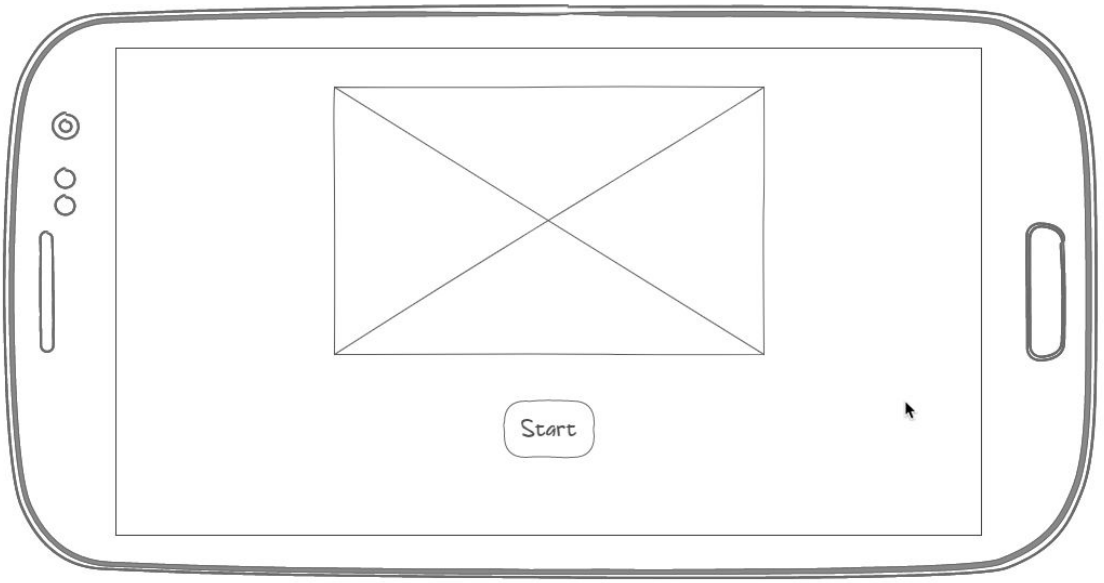


Figura 5 - Wireframe da tela de título

A figura 5 apresenta o wireframe da tela de título. Nela podemos observar o botão “Start” que na demo do jogo, fará com que o jogador entre na tela de gameplay de caça.

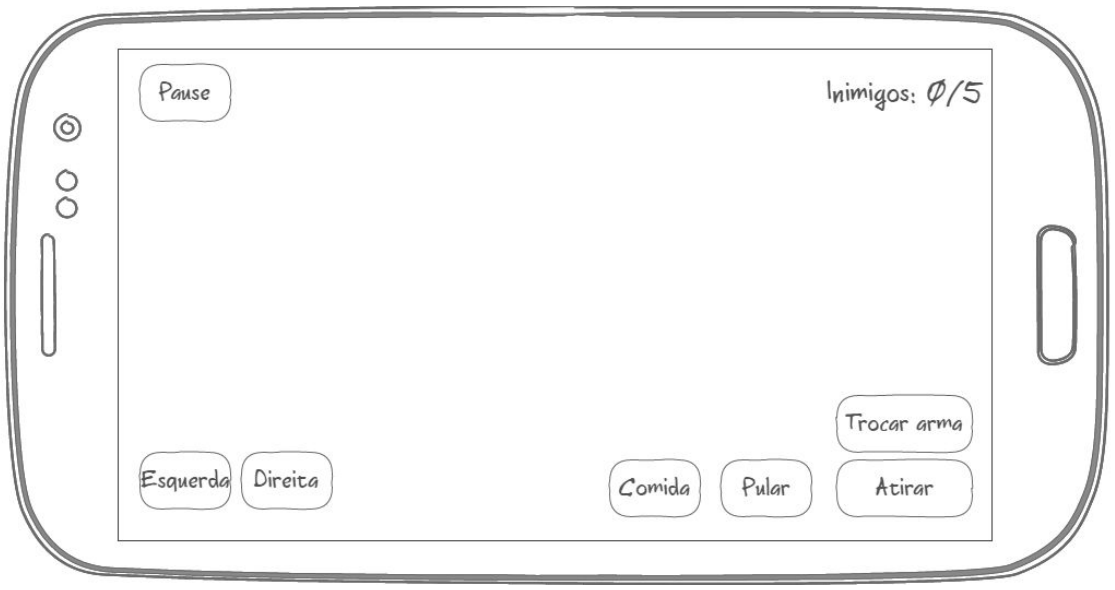


Figura 6 - Wireframe da tela gameplay caça

Na figura 6 podemos observar alguns botões que devem executar comandos ao serem pressionados. O botão “Esquerda” deverá mover o personagem para a esquerda, o botão “Direita” deverá mover o personagem para direita. O botão “Comida” servirá para o jogador arremessar a comida no chão, o botão “Pular” servirá para fazer o personagem pular, o botão “Atirar” fará com o que o jogador atire caso a arma já esteja carregada ou em condições de atirar; e o botão “Trocar arma” servirá para o jogador mudar a arma. O botão “Pause” irá pausar o jogo e apresentar a tela da figura 7, já o texto “Inimigos: 0/5” representará a contagem de inimigos disponíveis no cenário (no exemplo representado pelo 5) e quantos inimigos o jogador conseguiu capturar (no exemplo representado pelo 0).

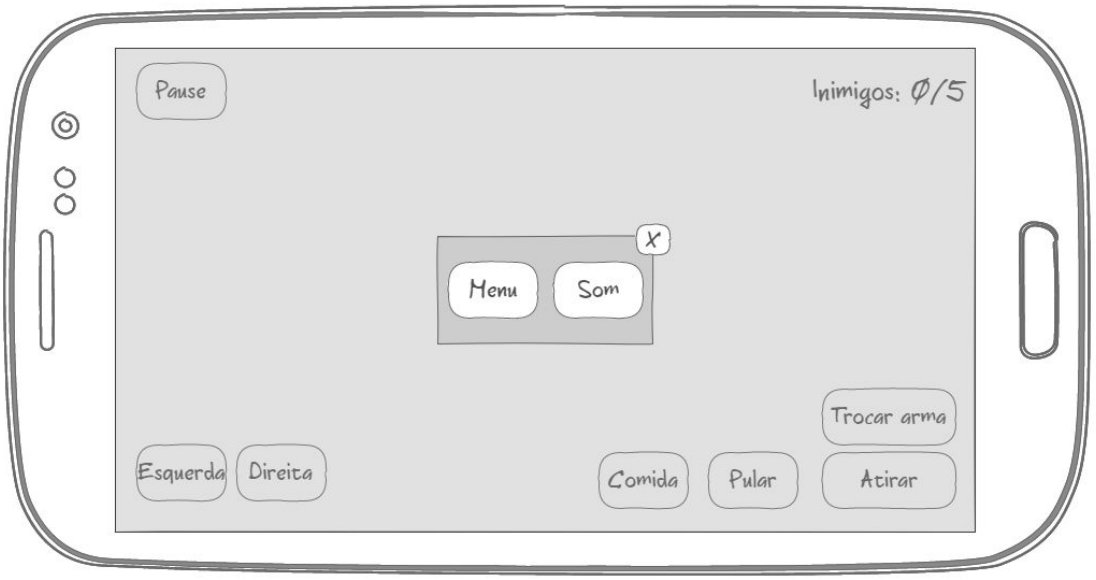


Figura 7 - Wireframe da popup pause

Na figura 7 temos a janela de pause e os botões “Menu”, “Som” e “X”. O botão “Menu” fará com que o jogador volte a tela de gameplay da loja, o botão “Som” servirá para o jogador habilitar e desabilitar o som do jogo e o botão “X” servirá para ele fechar a janela de pausa e voltar para o jogo.

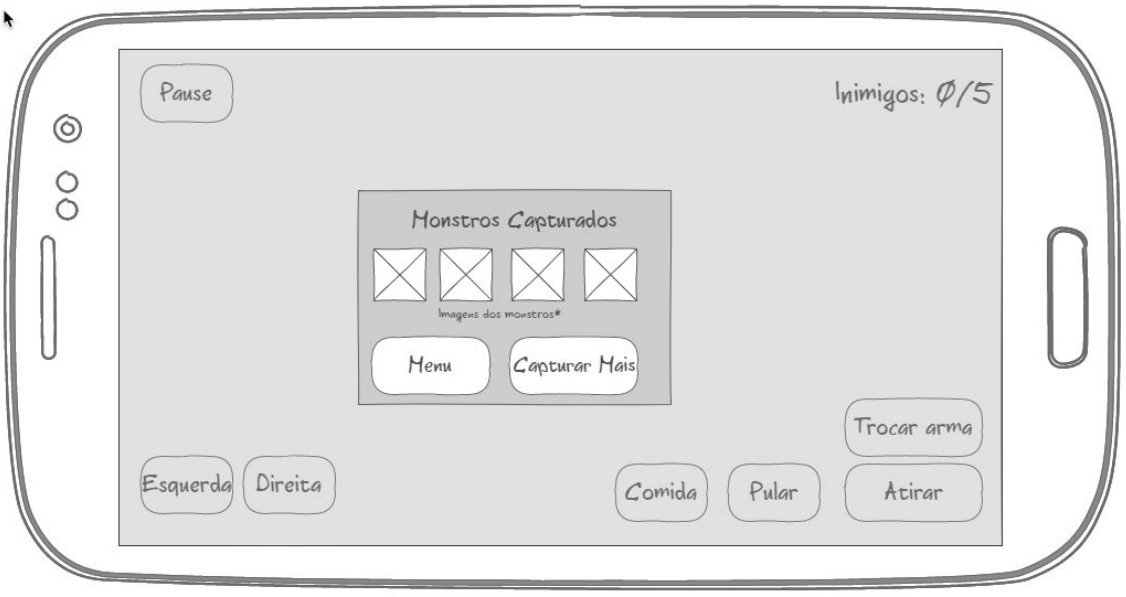


Figura 8 - Wireframe da popup fim caça

Na figura 8 temos a janela de fim de caça, essa tela é apresentada quando o último monstro do cenário é capturado ou foge. Possui os botões “Menu” e “Capturar Mais”. O botão “Menu” fará com que o jogador volte a tela de gameplay da loja, já o botão “Capturar Mais” fará o jogador ir para a tela de cenários de caça.

3.2.1.9. MECÂNICAS DO JOGO

3.2.1.9.1. GAMEPLAY

3.2.1.9.1.1. VEÍCULO

O jogador terá um veículo para vender os seus produtos e se locomover até os lugares para caçar os monstros. Esse veículo será uma Van Gourmet similar aos food trucks da realidade, e no fundo dela terá um compartimento secreto que contém a cozinha com as chapas de hambúrguer, onde os monstros são colocados para produzir os produtos.

Para produzir cada produto será necessária uma quantidade variadas de monstros, que devem ser colocados na chapa e esperar o tempo de produção que varia de chapa para chapa, cada produto produzido irá render uma quantia variada de “Ouro”.

Seguem abaixo algumas características da van e das chapas:

- Van Gourmet
    - Terá melhorias de aparência como rodas, pintura e adesivo. Essas melhorias serão liberadas à medida que o usuário for passando de nível.

- Chapas de Hambúrguer
    - Cada produto terá sua chapa para ser preparado, a chapa irá precisar de uma quantidade variada de um ou mais monstros para fazer o produto.
    - A chapa pode ser melhorada e produzir mais unidades do produto e dessa forma aumentar o lucro da loja com a mesma quantidade de monstros.

3.2.1.9.1.2. RAÇÃO PARA MONSTROS

Para que o jogador possa caçar os monstros precisará de ração para monstros. O jogador possuirá um armazém com capacidade limitada de rações, quando ele for para uma caçada gastará uma ração desse estoque. Caso o estoque não esteja cheio, o fornecedor trará uma ração a cada 10 minutos. Sendo assim, o jogador que gastar toda a ração para monstro precisará esperar 10 minutos para realizar uma nova caçada.

3.2.1.9.1.3. NÍVEIS DO JOGADOR

O jogador precisará passar de nível para desbloquear novos produtos, armas, cenários e chapas de hambúrguer.

3.2.1.9.1.4. MOEDAS

O jogo terá dois tipos de moedas o Ouro e o Osso. O ouro será a moeda mais comum do jogo e servirá para melhorar os equipamentos, recursos da loja e adquirir itens e roupas. Já o “Osso”será uma moeda mais rara e será obtida através de dinheiro real, conquistas e alguns eventos no jogo. Essa moeda servirá para comprar roupas e trocar por uma quantia grande de Ouro.

Segue abaixo características dessas moedas:

- Ouro
    - Adquirida ao vender os produtos na loja;
    - Adquirida ao passar por níveis;
    - Adquirida ao coletar troféus de caçador; e
    - Alta chance de dropar ao capturar monstros.
- Ossos
    - Adquirida com dinheiro real;
    - Adquirida ao passar por níveis;
    - Adquirida ao coletar troféus de caçador; e
    - Baixa chance de dropar ao capturar monstros.

3.2.1.9.1.5. LOJA

O jogo terá um sistema de loja onde o jogador poderá comprar e melhorar equipamentos, roupas, itens e recursos da sua hamburgueria. Alguns itens serão vendidos somente com Ossos.

Segue abaixo alguns exemplos de itens da loja:

- Roupas
    - As roupas servirão para personalizar o personagem e para fornecer bônus de moeda ao capturar os monstros e chance de aparecer mais monstros em cada nível;
    - Por mais que não seja possível equipar mais de uma roupa ao mesmo tempo, os bônus serão acumulados ao comprar mais de uma roupa.
- Novos Ajudantes / Shows / Decoração
    - Será possível contratar novos ajudantes, ou até mesmo uma banda para tocar ou decorar o local visando agradar mais os clientes e dessa maneira, aumentar a probabilidade dos clientes darem gorjetas.
- Fornecedor de ração para monstros
    - Será possível pagar, com Ossos, para o fornecedor trazer mais rações a cada 10 minutos.

3.2.1.9.1.6. MISSÕES DE NÍVEL

Cada nível conterá missões específicas que o jogador deve realizar para adquirir experiência e passar de nível.

Segue abaixo alguns exemplos de missões de nível:

- Vender produtos;
- Adquirir/Melhorar armas;
- Melhorar veículo;
- Capturar X monstros; e
- Adquirir alguma roupa.

3.2.1.9.1.7. CÂMERA

A câmera terá uma perspectiva de jogo plataforma 2D, e seguirá o jogador permitindo uma área de visão maior para a direção que o jogador estiver se movendo. Desse modo, deve ser possível parametrizar essa distância para que ao realizar os testes no jogo, seja possível encontrar uma distância satisfatória. Além disso a câmera deve possuir propriedades para representar os limites até o ponto em que ela poderá se movimentar, isso servirá para representar o início e o final de cada cenário.

3.2.1.9.1.8. AÇÕES DO JOGADOR

O jogador terá os movimentos de correr, pular, atirar, mudar de arma e jogar comida. Esses movimentos serão comandados através de controles na tela que serão representados através de botões.

O jogador só poderá realizar a ação de correr na horizontal e deve ser possível parametrizar a velocidade de movimento do personagem.

A ação de pulo deve ser executada somente quando o jogador estiver pisando em alguma plataforma, desse modo não é possível realizar um pulo duplo no jogo. Além disso é necessário que a força de pulo seja parametrizável.

A ação de atirar deve ser validada de acordo com a arma que o jogador estiver utilizando, cada arma terá características próprias e sendo assim cada arma terá um comportamento completamente diferente de outra arma.

A ação de mudar de arma servirá para o jogador trocar entre a arma principal e a arma secundária que escolheu levar na caçada com ele.

A ação de jogar comida servirá para atrair os inimigos que estarão escondidos no cenário, é necessário que tenha um delay para que o jogador não fique repetindo essa ação infinitamente de forma rápida.

3.2.1.9.1.9. ARMAS

O jogador poderá adquirir e utilizar vários tipos de armas que serão desbloqueadas à medida que o jogador for progredindo no jogo. Essas armas poderão ser melhoradas através da loja do jogo e a melhoria varia de acordo com a arma.

Segue abaixo a lista de armas do jogo:

- Arma que atira corda boleadeira (principal)
    - Tem delay entre os tiros e tem um alcance relativamente curto;
    - Terá melhoria de alcance para atirar mais longe.

- Arma que atira goma de mascar (será como um lança granadas)
    - A arma possui um alcance do tiro;
    - A trajetória do tiro será exibida quando o inimigo estiver dentro do alcance da arma e o tiro irá em direção ao inimigo mais próximo;
    - O tiro explode e prende os monstros no raio da explosão; e
    - Terá melhoria de alcance da explosão.

- Arma que atira um raio que “empacota” o monstro
    - A arma tem um tiro que sempre vai reto e não tem limite de alcance;
    - Quando o tiro encostar no monstro, ele será empacotado dentro de uma caixa ou grade; e
    - Terá melhoria de atravessar um monstro podendo atingir mais monstros ao mesmo tempo.

- Armadilha congelante
    - O jogador joga uma armadilha no chão que congela o monstro quando ele toca na armadilha;
    - Terá melhoria de congelar em área.

3.2.1.9.1.10. AÇÕES DOS INIMIGOS

Todos os inimigos irão se comportar praticamente da mesma forma. Eles irão se movimentar através de alguns estágios e o objetivo de todos os inimigos será sempre fugir do cenário sem que sejam capturados.

Quando o jogador entrar no cenário, todos os inimigos estarão escondidos. Os esconderijos serão indicados por um ícone no cenário e eles só sairão de lá caso o jogador arremesse uma comida próximo ao esconderijo. Caso o jogador esteja fora do campo de visão dele, essas duas condições devem ser satisfeitas juntas para que o inimigo comece a olhar para a comida e decida sair do esconderijo.

Após sair, o inimigo andará até a comida mais próxima e ficará se alimentando. Caso não tenha nenhuma comida dentro do campo de visão o inimigo ficará simplesmente andando e parando aleatoriamente.

Quando o inimigo estiver fora do esconderijo e o jogador ficar dentro do seu campo de visão durante um determinado tempo, o inimigo avistará o jogador e tentará correr na direção oposta para fugir. O inimigo pode também arremessar algum item com o intuito de ferir e deixar o jogador atordoado durante algum tempo facilitando a sua fuga. Além disso, caso o jogador encoste no inimigo, ele tomará dano e ficará atordoado.

3.2.1.9.1.11. BOSSES

Cada cenário terá um boss que irá aparecer de acordo com uma variação de tempo e objetivos que o jogador deverá cumprir. Caso queria, o jogador terá a opção de capturar o boss e caso consiga capturá-lo, ganhará uma quantia de Ossos e poderá usar o boss para produzir qualquer tipo de produto daquele tipo de monstro.

3.2.1.9.2. METAGAME

3.2.1.9.2.1. MISSÕES DIÁRIAS

As missões diárias irão beneficiar o jogador com itens e moedas.

3.2.1.9.2.2. TROFÉUS DE CAÇADOR

Os Troféus de caçador são obtidos após completar um número variado de fases capturando todos os monstros dela. Ao obter esses troféus, o jogador irá ser recompensado com itens e moedas.

3.2.1.9.2.3. CONQUISTAS

O jogo terá sistema de conquistas visando um desafio maior e a longo prazo para o jogador que quiser finalizar o jogo com 100% de aproveitamento.

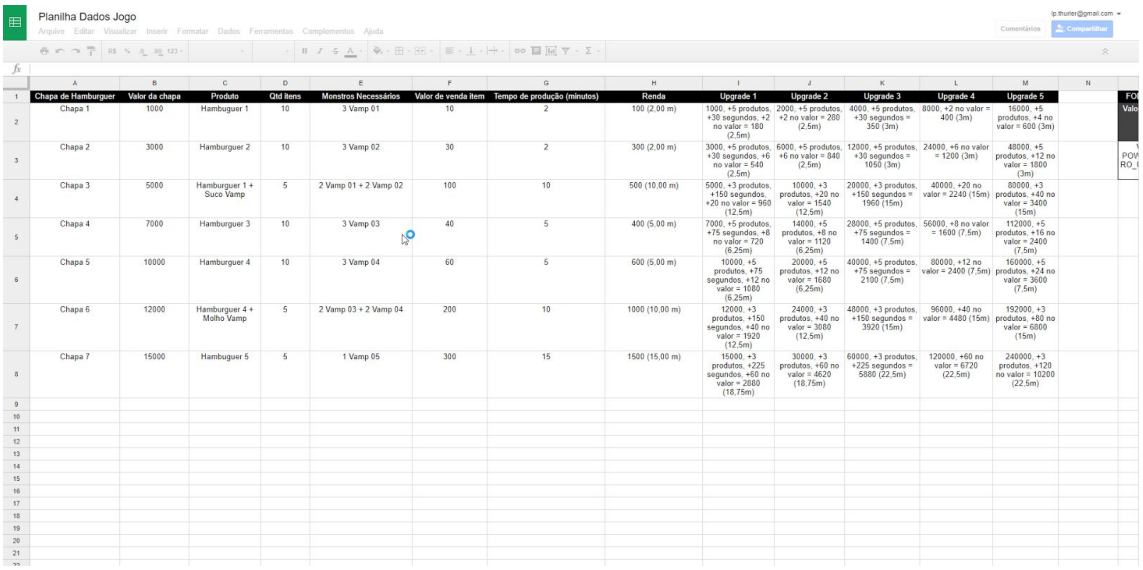
3.2.1.9.2.4. RANKS

O jogo terá rank social comparando os maiores caçadores de todos os tipos de monstros. Esse rank irá proporcionar uma forma dos jogadores disputarem com os seus amigos quem é o maior caçador. Os ranks serão divididos em um rank de um tipo específico de monstro e outro rank com o somatório de todos os monstros caçados.

3.2.2. DOCUMENTOS COMPLEMENTARES

Algumas informações não precisam ser detalhadas no GDD. Informações como o custo e bonûs obtidos ao realizar a melhoria de alguma arma no jogo ou em que níveis os inimigos são liberados, ficam melhor detalhados em planilhas ou em outros tipos de documentos. Sendo assim, é comum que sejam criados outros

documentos para detalhar essas informações que não precisaram entrar no GDD, mas que são essenciais para a que a equipe possa desenvolver o jogo.



Planilha Dados Jogo

Chapa de Hamburguer	Valor da chapa	Produto	Qtd itens	Monstros Necessários	Valor de venda item	Tempo de produção (minutos)	Renda	Upgrade 1	Upgrade 2	Upgrade 3	Upgrade 4	Upgrade 5
Chapa 1	1000	Hamburguer 1	10	3 Vamp 01	10	2	100 (2,00 m)	1000, +5 produtos, +30 segundos, +2 no valor = 180 (2,5m)	2000, +5 produtos, +2 no valor = 280 (2,5m)	4000, +5 produtos, +30 segundos = 350 (3m)	8000, +2 no valor = 400 (3m)	16000, +5 produtos, +4 no valor = 600 (3m)
Chapa 2	3000	Hamburguer 2	10	3 Vamp 02	30	2	300 (2,00 m)	3000, +5 produtos, +30 segundos, +6 no valor = 540 (2,5m)	6000, +5 produtos, +6 no valor = 840 (2,5m)	12000, +5 produtos, +30 segundos = 1050 (3m)	24000, +6 no valor = 1200 (3m)	48000, +5 produtos, +12 no valor = 1800 (3m)
Chapa 3	5000	Hamburguer 1 + Suco Vamp	5	2 Vamp 01 + 2 Vamp 02	100	10	500 (10,00 m)	5000, +3 produtos, +150 segundos, +20 no valor = 960 (12,5m)	10000, +3 produtos, +20 no valor = 1540 (12,5m)	20000, +3 produtos, +150 segundos = 1960 (15m)	40000, +20 no valor = 2240 (15m)	80000, +3 produtos, +40 no valor = 3400 (15m)
Chapa 4	7000	Hamburguer 3	10	3 Vamp 03	40	5	400 (5,00 m)	7000, +5 produtos, +75 segundos, +8 no valor = 720 (6,25m)	14000, +5 produtos, +8 no valor = 1120 (6,25m)	28000, +5 produtos, +75 segundos = 1400 (7,5m)	56000, +8 no valor = 1600 (7,5m)	112000, +5 produtos, +16 no valor = 2400 (7,5m)
Chapa 5	10000	Hamburguer 4	10	3 Vamp 04	60	5	600 (5,00 m)	10000, +5 produtos, +75 segundos, +12 no valor = 1080 (6,25m)	20000, +5 produtos, +12 no valor = 1680 (6,25m)	40000, +5 produtos, +75 segundos = 2100 (7,5m)	80000, +12 no valor = 2400 (7,5m)	160000, +5 produtos, +24 no valor = 3600 (7,5m)
Chapa 6	12000	Hamburguer 4 + Molho Vamp	5	2 Vamp 03 + 2 Vamp 04	200	10	1000 (10,00 m)	12000, +3 produtos, +150 segundos, +40 no valor = 1920 (12,5m)	24000, +3 produtos, +40 no valor = 3080 (12,5m)	48000, +3 produtos, +150 segundos = 3920 (15m)	96000, +40 no valor = 4480 (15m)	192000, +3 produtos, +80 no valor = 6800 (15m)
Chapa 7	15000	Hamburguer 5	5	1 Vamp 05	300	15	1500 (15,00 m)	15000, +3 produtos, +225 segundos, +60 no valor = 2880 (18,75m)	30000, +3 produtos, +60 no valor = 4620 (18,75m)	60000, +3 produtos, +225 segundos = 5880 (22,5m)	120000, +60 no valor = 6720 (22,5m)	240000, +3 produtos, +120 no valor = 10200 (22,5m)

Figura 9 - Planilha com as informações das chapas de hambúrguer

A figura 9 apresenta uma planilha contendo as informações de custo, bônus e renda das chapas de hambúrguer do jogo. Essas informações servem para balancear o jogo de uma forma que seja possível definir a velocidade de progressão.

3.3. ARTE

3.3.1. CRIAÇÃO DOS PERSONAGENS

Para a criação dos personagens no jogo foram utilizados os softwares Adobe Photoshop, Flash e Adobe Illustrator.

3.3.1.1. PERSONAGEM PRINCIPAL

Figura 10 - Esboço do personagem principal no photoshop

Conforme pode ser observado na figura 10, o photoshop foi utilizado para a criação do esboço do personagem, através da ferramenta de camadas. É possível separar os elementos que constituem o personagem. Essa separação em camadas permite a edição de cada parte da ilustração de forma independente e isso agiliza muito a criação do personagem.

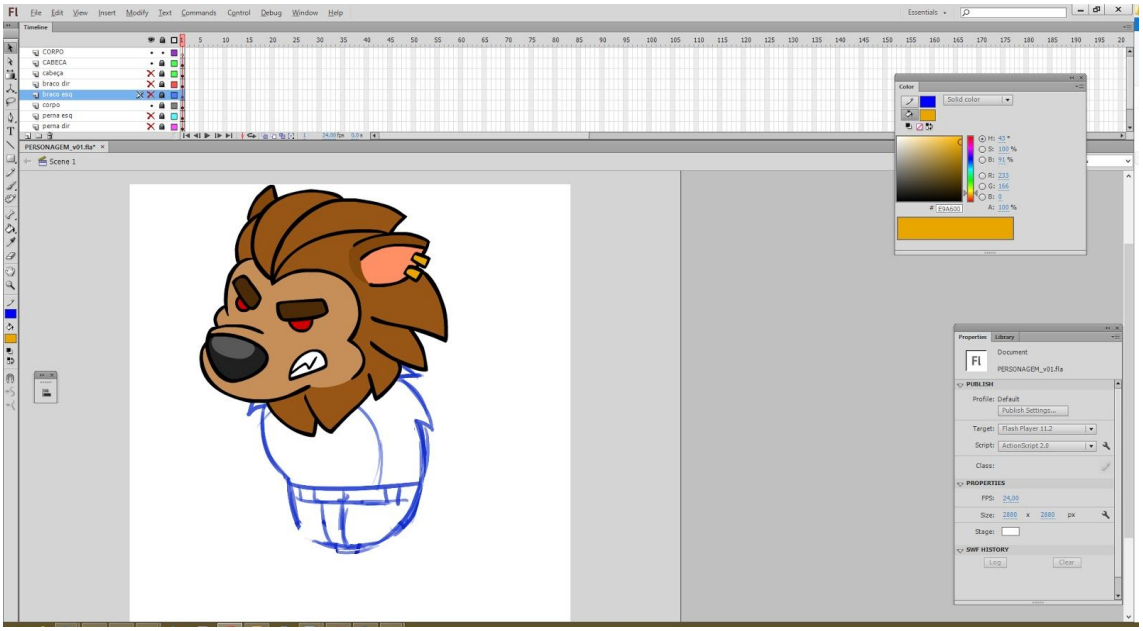


Figura 11 - Vetorização do personagem principal em andamento no flash

Uma vez finalizado o esboço do personagem no photoshop, o mesmo é importado para dentro do Flash para que seja iniciado o processo de vetorização a partir desse esboço. A figura 11 apresenta o processo de vetorização em andamento. Nessa figura é possível observar que assim como no photoshop, o Flash também trabalha com camadas para separar os elementos da ilustração. Então, a medida que um elemento do personagem é finalizado, inicia-se o processo de vetorização do outro elemento em uma camada diferente, para ser mais fácil a edição do personagem como um todo. Este processo é realizado até finalizar completamente o personagem como pode ser visto na figura 12.

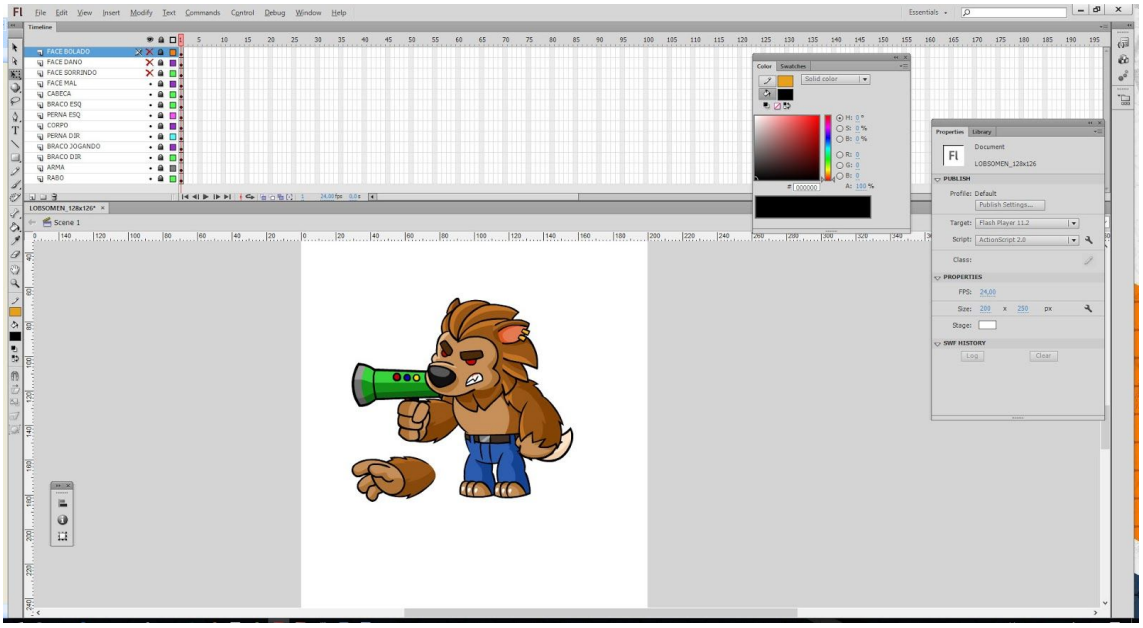


Figura 12 - Vetorização do personagem principal finalizada no flash

Após finalizar todo o processo de vetorização do personagem, foi preciso exportar cada elemento do personagem em imagens diferentes conforme pode ser visto na figura 13. Isso foi necessário para que seja possível animar o personagem através do programa Spriter.

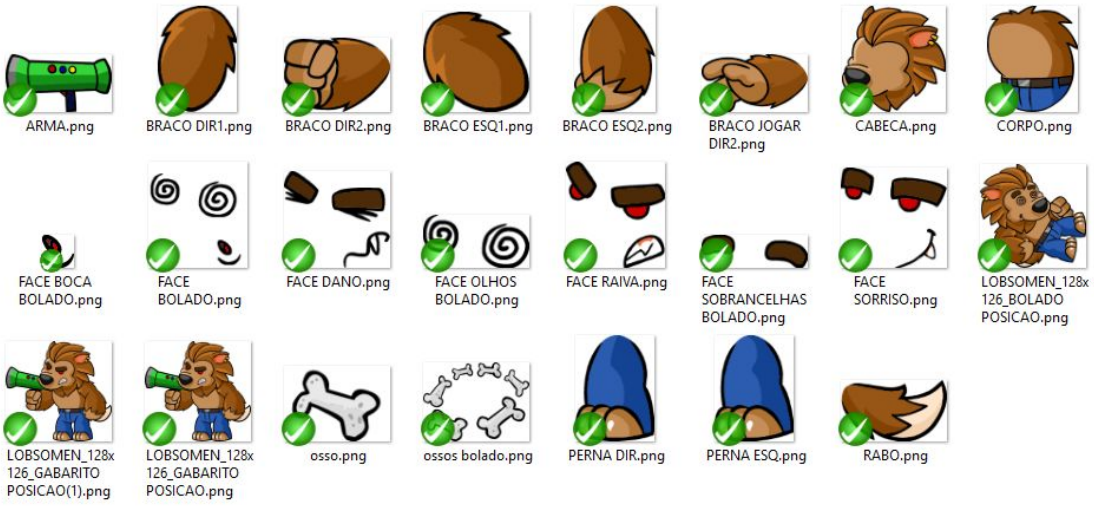


Figura 13 - Imagens do personagem principal exportada pelo flash

3.3.1.2. INIMIGOS

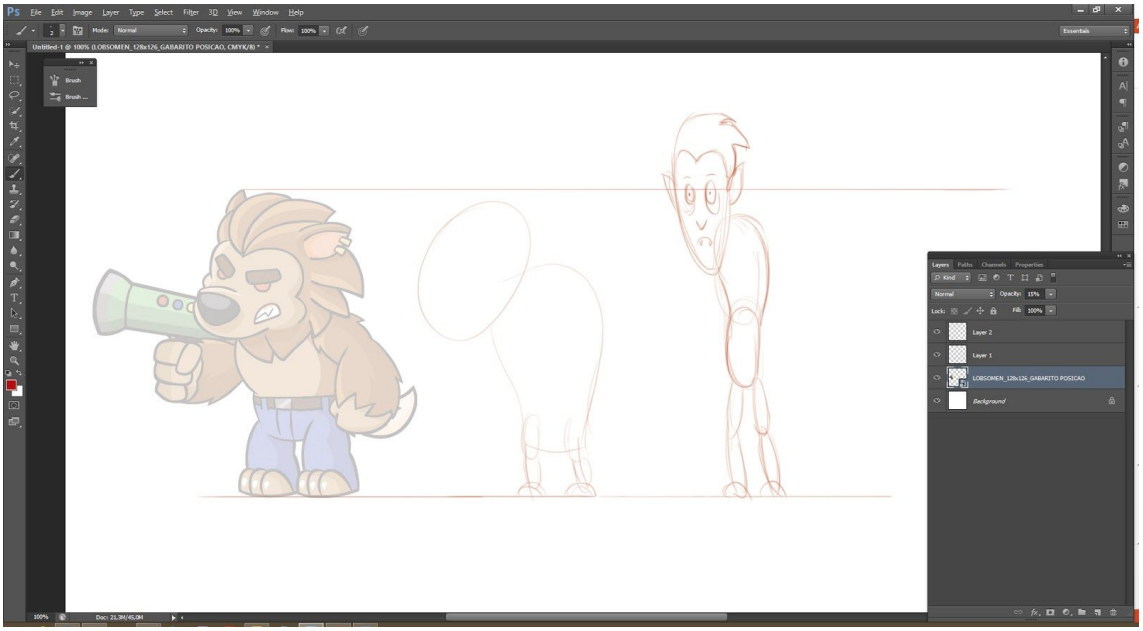


Figura 14 - Esboço do inimigo Vamp-01 no photoshop

Na figura 14 podemos observar que assim como o personagem principal, o processo de criação dos inimigos se iniciou na construção do esboço do inimigo no photoshop. A diferença é que dessa vez, foi utilizado o personagem principal como referência de tamanho para a criação do inimigo, e, assim, através dessa referência, foi possível definir a estatura que os inimigos teriam.

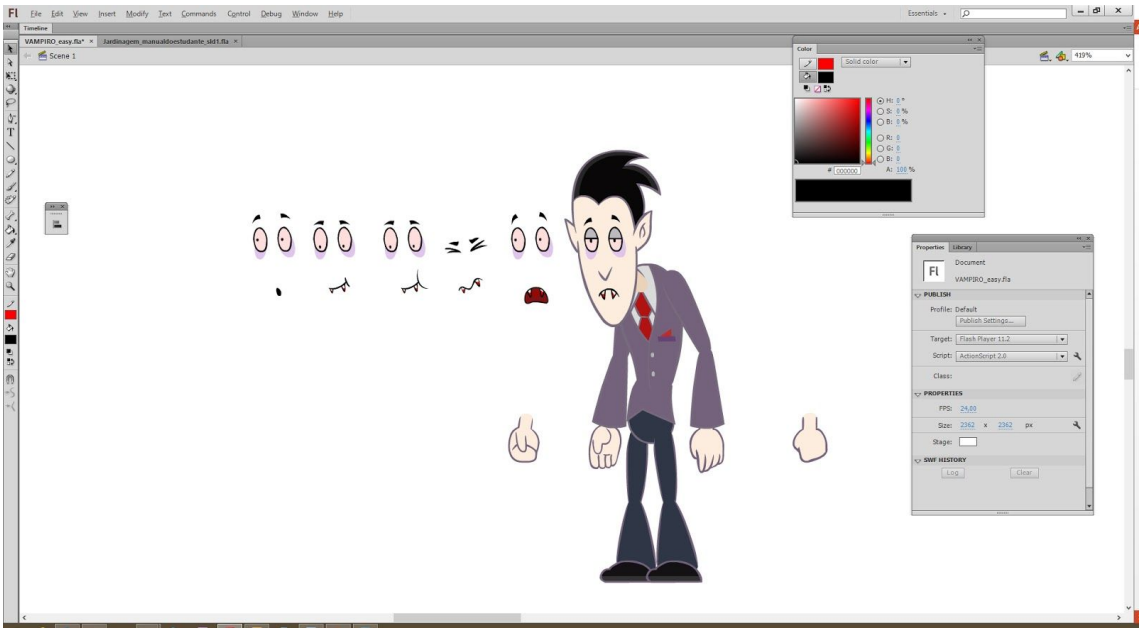


Figura 15 - Inimigo Vamp-01 finalizado no flash

A figura 15 apresenta o inimigo Vamp-01 totalmente vetorizado no flash. O processo de vetorização dele foi idêntico ao do personagem principal e também foi preciso exportar as imagens separadas para futuramente criar a animação através do Spriter.

3.3.2. CRIAÇÃO DO TILESET DO CENÁRIO

Para a criação do cenário e dos seus elementos no jogo, foram utilizados os softwares Photoshop e o Illustrator.

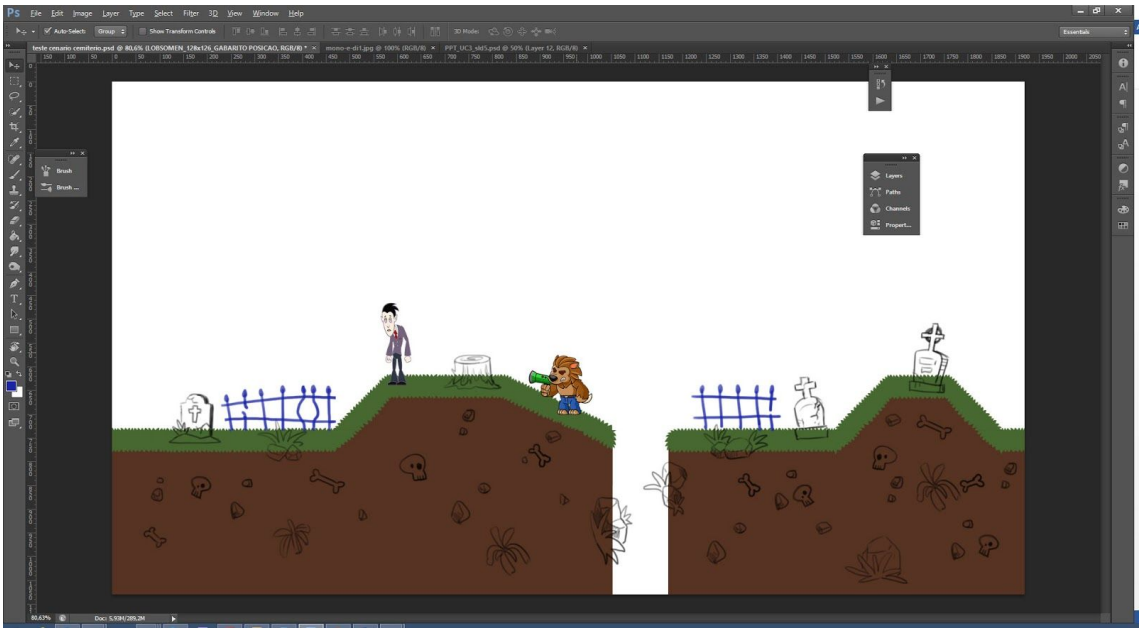


Figura 16 - Esboço do cenário e dos elementos no photoshop

Assim como na criação dos personagens, também foi necessário criar um esboço para o cenário e seus elementos. Conforme pode ser visto na figura 16, para a criação desse esboço foi novamente utilizado o photoshop, e agora, tanto o personagem principal e o inimigo foram utilizados como referência para definir o tamanho dos elementos que iriam compor o cenário.

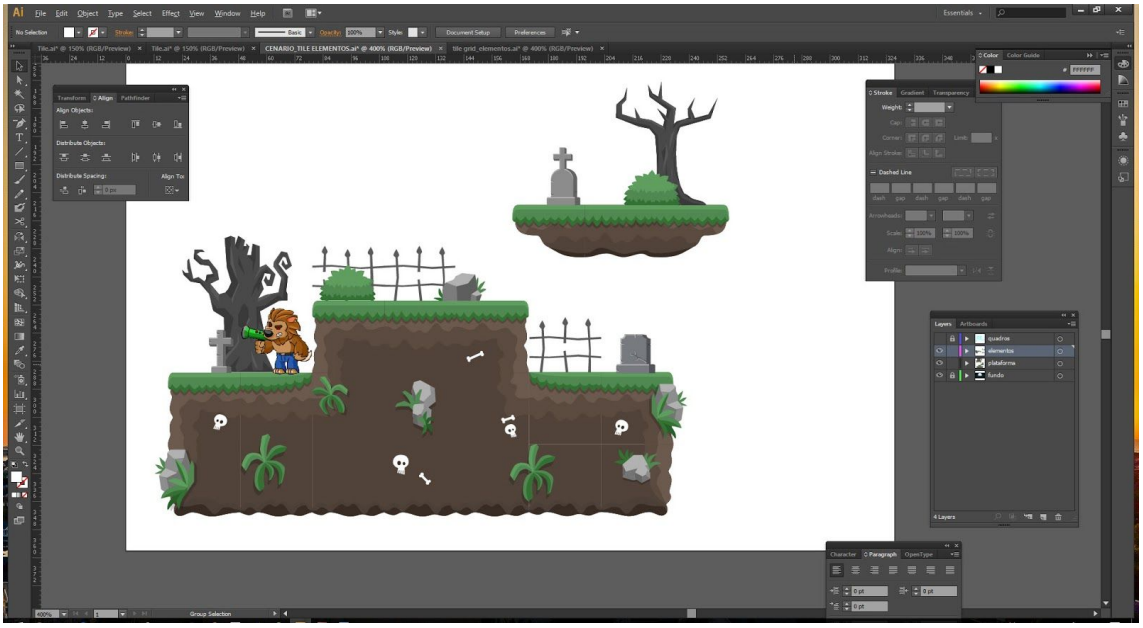


Figura 17 - Cenário finalizado no Illustrator

Após a criação do esboço do cenário no photoshop, o mesmo foi importado para o illustrator e assim como foi realizado para os personagens no flash, foi iniciado o processo de vetorização dos elementos do cenário. A figura 17 apresenta o cenário e seus elementos completamente vetorizados.

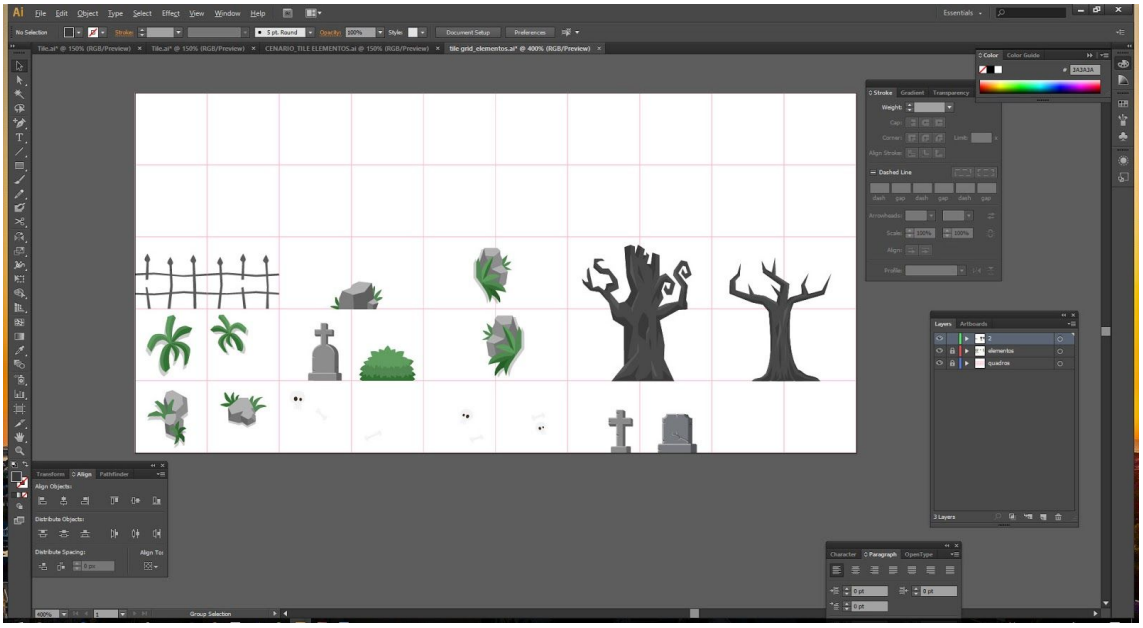


Figura 18 - Elementos do cenário separados dentro do tileset

Para que fosse possível utilizar o cenário e seus elementos do cenário dentro do jogo, foi criado um tileset para o cenário. A figura 18 demonstra como foram separados os elementos dentro da marcação do tileset. Após isso, foram exportadas as imagens da figura 19 que seriam utilizadas pelo Tiled para gerar o cenário.

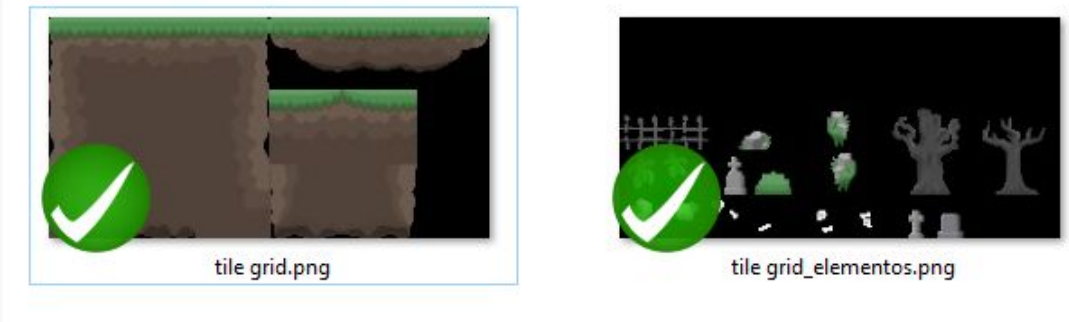


Figura 19 - Tilesets do cenário

3.3.3. CRIAÇÃO DOS ELEMENTOS DA INTERFACE

Para a criação dos elementos da interface foram utilizados os softwares Ninjamock, Photoshop e Illustrator. O Ninjamock foi utilizado para gerar os wireframes que foram apresentados na seção de Game Design.

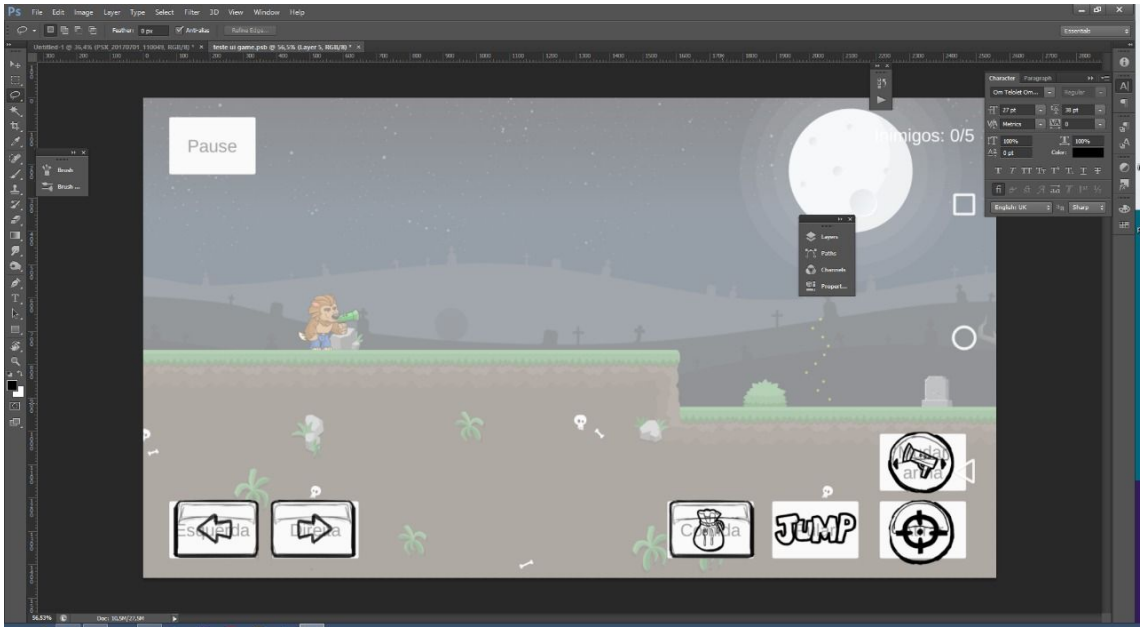


Figura 20 - Esboço dos elementos da interface no photoshop

Conforme pode ser visto na figura 20, uma vez que os wireframes já estavam criados eles foram utilizados como base para a criação do esboço de cada elemento da interface através do photoshop.

Figura 21 - Construção dos elementos da interface no Illustrator

Na figura 21 pode ser visto que o esboço que foi criado para cada elemento da interface foi importado para o illustrator. E a partir destes esboços, foram criados os elementos vetorizados que seriam utilizados no jogo. A figura abaixo demonstra alguns dos elementos que foram criados para a interface.

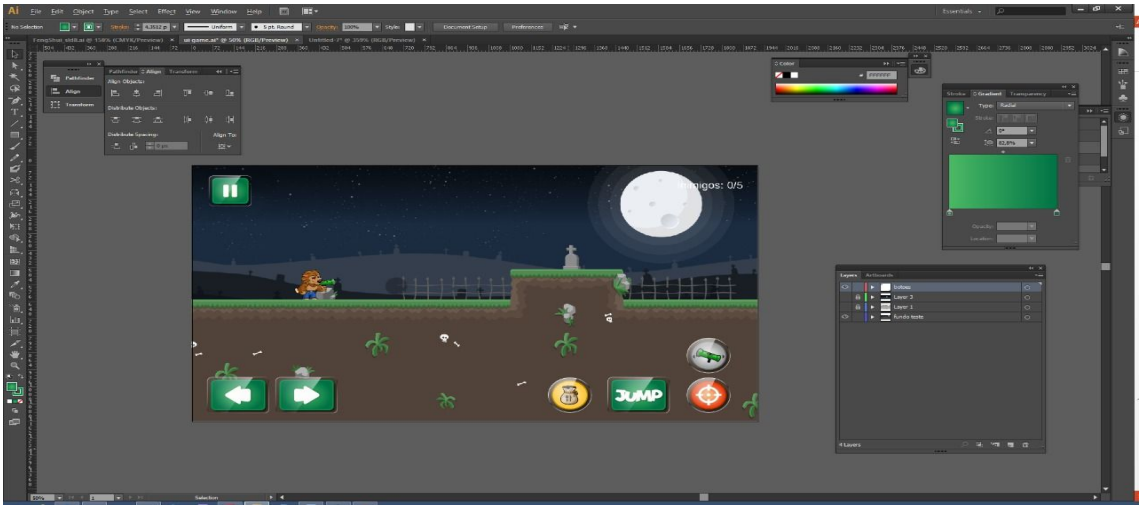


Figura 22 - Elementos da interface no Illustrator

3.3.4. MONTAGEM DO MAPA NO TILED

Após a criação dos tilesets do cenário, foi utilizado o Tiled para a realizar a montagem dos blocos que posteriormente foram importados pelo Unity 3D para fazer parte jogo.

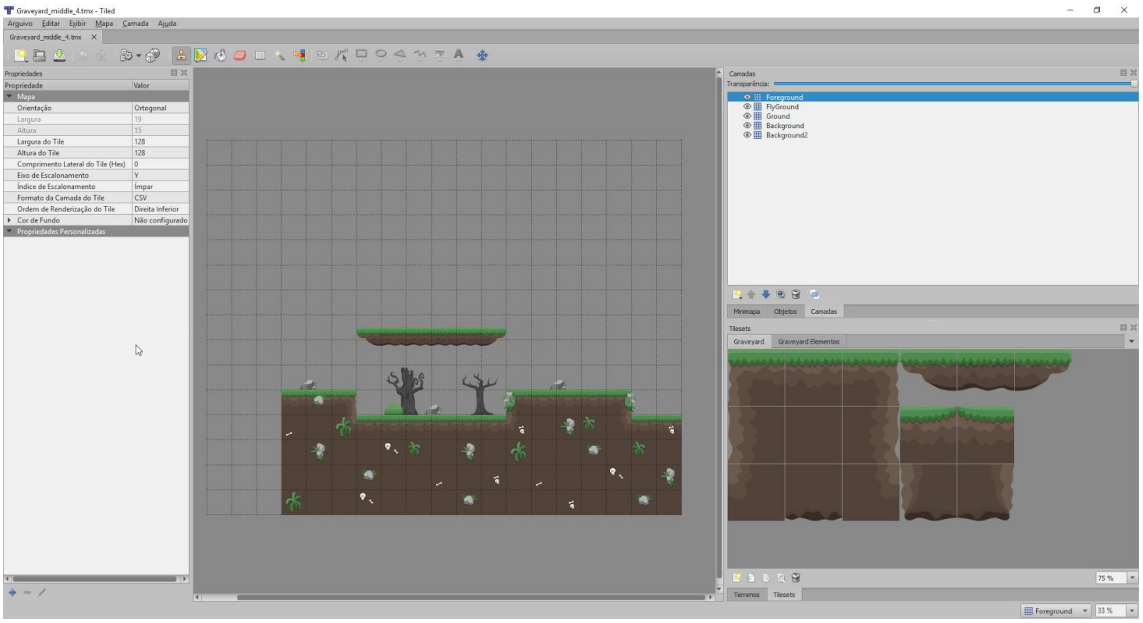


Figura 23 - Tela do Tiled contendo uma parte do cenário do cemitério

Na figura 23 pode ser visto uma das partes que foi criada para o cenário do cemitério. Nela é possível observar que o mapa possui 5 camadas. Essas camadas servem para posicionar os elementos a frente de outros elementos e dessa forma, foi possível colocar as pedras e gramas acima do chão, por exemplo.

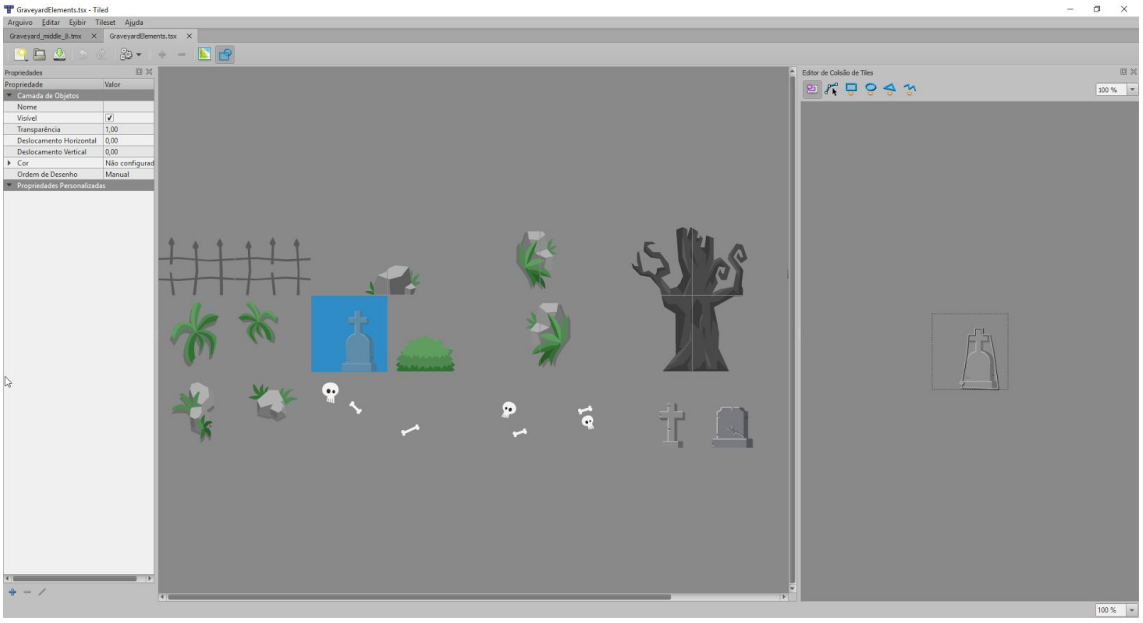


Figura 24 - Tela do Tiled com o editor de colisões

Além de ser possível a criação do mapa contendo toda a aparência do cenário, o Tiled também permite definir os pontos de colisões dos elementos do tileset, como pode ser visto na figura 24. Dessa forma, quando esse mapa é importado na Unity 3D, além de possuir a aparência do cenário ele também possui as informações dos colisores facilitando assim a montagem do jogo.

3.3.5. ANIMAÇÕES NO SPRITER

Para criar as animações dos personagens, foi utilizado o software Spriter. Sendo assim, após a criação dos elementos que compõem os personagens, foram importados para o Spriter para que fosse possível criar as animações.

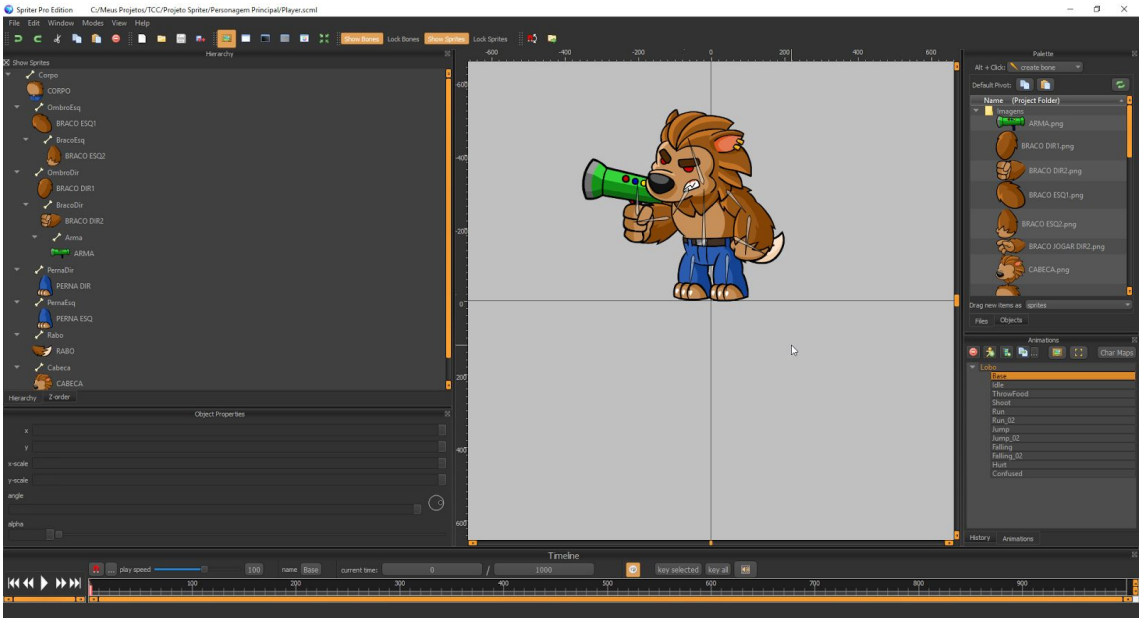


Figura 25 - Personagem principal no Spriter

O primeiro passo após importar as imagens do personagem no Spriter, foi criar o esqueleto do personagem e atribuir as imagens à esse esqueleto. Conforme pode ser visto na parte esquerda figura 25, o esqueleto funciona como uma hierarquia onde é possível atribuir imagens e pedaços de esqueletos um dentro do outro. Através dessa hierarquia, o Spriter faz as ligações desses elementos. Sendo assim, ao rotacionar um pedaço do esqueleto, todos os pedaços que estiverem dentro dele também serão rotacionados, facilitando a criação das animações.

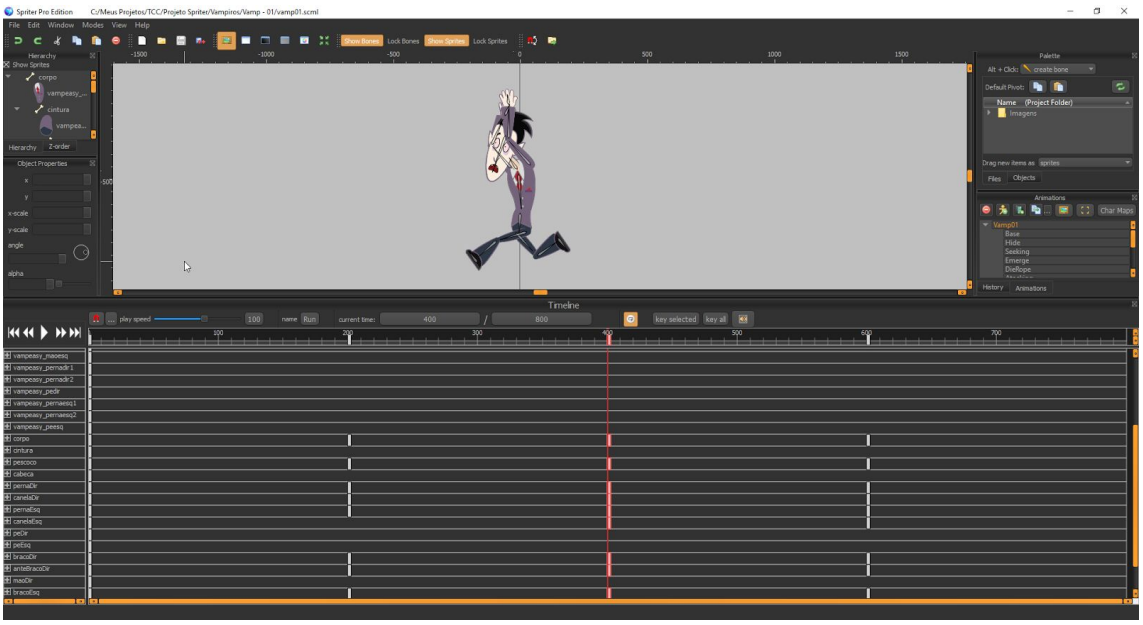


Figura 26 - Timeline da animação de corrida do Vamp-01 no Spriter

A figura 26 apresenta a timeline de uma das animações do personagem Vamp-01 do jogo. É através dessa timeline que é possível criar as animações no Spriter. Após criar o esqueleto do personagem, é preciso utilizar a timeline para realizar as transformações nos elementos. Pode-se fazer todo tipo de transformação na imagem como rotacionar, modificar transparência, aumentar, diminuir e diversas outras ações. A animação é criada realizando essas transformações em tempos distintos na timeline. Essas transformações ficam representadas pelos retângulos que é possível ver na timeline. Após aplicar essas transformações, o Spriter faz com que sejam aplicadas proporcionalmente no intervalo de tempo que foi deixado entre elas, criando assim, as animações.

3.4. PROGRAMAÇÃO

3.4.1. AÇÕES DO JOGADOR

Para permitir as ações do personagem no jogo foram criados vários scripts com base nas especificações de movimentos e interações que foram definidas no GDD. Essa seção irá demonstrar algumas funções que foram criadas especificamente para as ações do personagem.

3.4.1.1. CORRER

A tabela 1 demonstra a função que executa a ação de correr do personagem, essa função funciona basicamente checando os valores do inputs do jogador para definir em qual direção o personagem irá se movimentar. Ele também possui uma propriedade para indicar a velocidade de movimento do personagem e, além disso, aplica ao cálculo de velocidade a multiplicação ao valor “Time.deltaTime”. Essa multiplicação faz com que a velocidade de movimentação não sofra alterações de acordo com o hardware que está sendo processado. Além do cálculo de velocidade, o script apresentado na tabela 1 verifica se é necessário virar o sprite do personagem de acordo com o movimento e caso precise ele faz uma chamada a função responsável por realizar essa ação.

```
  void Move()
  {
      if ((_isFacingRight && _xInputValue < 0) || (!_isFacingRight && _xInputValue
> 0))
        Flip();

          Vector2 newVelocity = Vector2.right * _xInputValue * _moveSpeed *
Time.deltaTime;
     newVelocity.y = _rb2D.velocity.y;
```

```
    _rb2D.velocity = newVelocity;
}
```

Tabela 1 - Script da ação correr do personagem

3.4.1.2. PULAR

A tabela 2 contém a função da ação de pular do personagem e utiliza uma propriedade para representar a força do pulo do jogador que é publicada pelo valor “Vector2.up” representando a direção em que a força deve ser aplicada. Ademais, ela possui uma propriedade que verifica se o jogador está posicionado acima do chão e só permite realizar o movimento caso essa condição seja verdadeira.

```
void Jump()
{
    if (_isGrounded)
    {
        _rb2D.velocity = new Vector2(_rb2D.velocity.x, 0);
        _rb2D.AddForce(Vector2.up * _jumpForce);
        _jump = false;
    }
}
```

Tabela 2 - Script da ação pular do personagem

3.4.1.3. JOGAR COMIDA

A tabela 3 mostra a função que permite o jogador arremessar a comida. Essa função utiliza uma propriedade para verificar se o jogador pode realizar o movimento de arremessar comida. Caso o jogador possa realizar essa ação, ele criará um novo objeto na cena que representará a comida e aplicará uma força de arremesso de

comida de acordo com a propriedade que foi designada para isso, considerando também a velocidade do jogador para definir a direção e força do arremesso final. Além disso, ele informa ao controlador de animações que o jogador realizou a ação de jogar a comida, também realizará o controle de tempo que o jogador precisa esperar para jogar a comida novamente.

```
    void ThrowFood()
    {
                                                    if (_canThrowFood &&
CrossPlatformInputManager.GetButtonDown("ThrowFood"))
        {
            _canThrowFood = false;
            Vector3 foodSpawnPosition = transform.position;
            foodSpawnPosition.x += _isFacingRight ? 0.5f : -0.5f;

                    Rigidbody2D foodRb = Instantiate(_foodPrefab, foodSpawnPosition,
Quaternion.identity).GetComponent<Rigidbody2D>();
            foodRb.velocity = new Vector2(_rb2D.velocity.x, foodRb.velocity.y);
                     foodRb.AddForce(((_isFacingRight ? Vector2.right : Vector2.left) +
Vector2.up) * _throwFoodForce);

            _playerAC.SetTrigger("throwFood");

            Destroy(foodRb.gameObject, 5);
        }
        else if (!_canThrowFood)
        {
            _throwFoodTime += Time.deltaTime;

            if (_throwFoodTime >= _delayThrownFood)
            {
```

```
            _canThrowFood = true;
            _throwFoodTime = 0;
        }
    }
}
```

Tabela 3 - Script da ação jogar comida do personagem

3.4.1.5. TOMAR DANO

A tabela 4 demonstra a ação tomar dano do personagem, essa função basicamente ativa uma propriedade que serve para identificar que o jogador está ferido e parar a sua movimentação. Faz com que o jogador seja empurrado na direção oposta ao ponto em que houve a colisão, o personagem é modificado de camada para que não sejam detectadas colisões com outros inimigos enquanto ele estiver ferido. Também informa ao controlador de animações que o personagem está feriado e no final ele utiliza uma propriedade para esperar a quantidade de tempo que o jogador deverá ficar ferido e após esse tempo o jogador voltará para a camada correta novamente, liberando as ações e colisões do jogador.

```
public void Hurt( Vector3 hitNormal )
{
    if (!_isHurt)
    {
        _isHurt = true;
        _rb2D.velocity = Vector2.zero;
        _rb2D.AddForce((-hitNormal + ( Vector3.up * 5 )) * _throwHurtForce);

        gameObject.layer = LayerMask.NameToLayer("PlayerNoCollision");
        _playerAC.SetBool("hurt", _isHurt);
        _playerAC.SetTrigger("initHurt");
```

```
        StartCoroutine(WaitHurtTime());
    }
}

IEnumerator WaitHurtTime()
{
    yield return new WaitForSeconds(_hurtTime);

    gameObject.layer = LayerMask.NameToLayer("Player");
    _isHurt = false;
    _playerAC.SetBool("hurt", _isHurt);
}
```

Tabela 4 - Script da ação tomar dano do personagem

3.4.2. CONTROLE DE ANIMAÇÕES DOS PERSONAGENS

Tanto o personagem principal quanto os inimigos possuem diversas animações que devem ser controladas para serem executadas nos momentos corretos, essas animações estão atreladas as diversas ações que esses personagens podem executar como: atirar, pular, cair, andar, correr e outras.

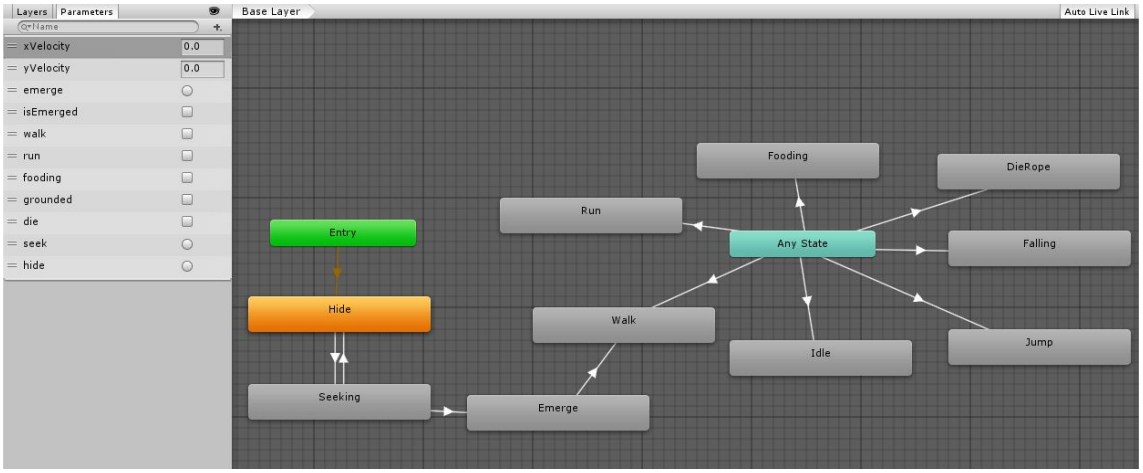


Figura 27 - Representação visual do controle de animações de um dos inimigos

Para que seja possível controlar essas animações foi utilizado um controlador que o Unity 3D disponibiliza. Este controlador pode ser visto na figura 27, nele devemos criar propriedades e criar um fluxo que indica que quando as animações podem ser alteradas. Dessa forma para cada personagem no jogo foi preciso criar um controlador com propriedades e condições variadas de acordo com o comportamento do personagem.

3.4.3. MOVIMENTAÇÃO DA CÂMERA

De acordo com a necessidade de movimentação da câmera que foi descrita no GDD, foi necessário criar um script para suprir essa necessidade.

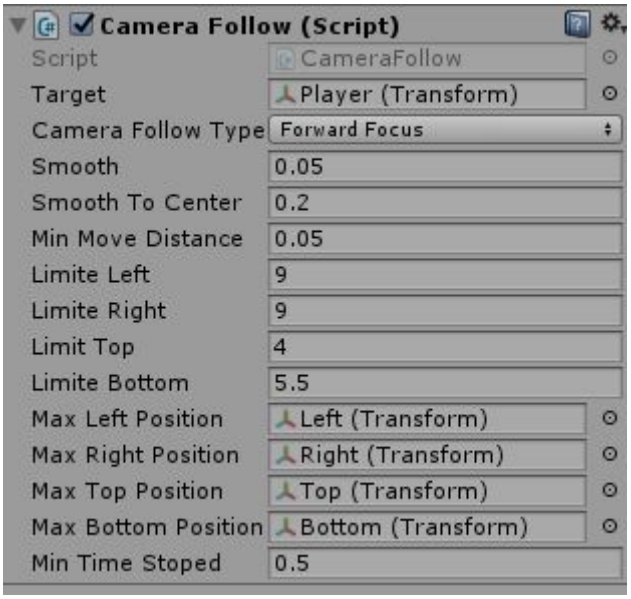


Figura 28 - Propriedades configuráveis no sistema de movimentação da câmera

Como pode ser visto na figura 28 o script disponibiliza um sistema que permite a configuração das propriedades necessárias para realizar a movimentação da câmera. Nesse sistema é possível definir qual será o alvo que a câmera deverá seguir e uma velocidade de suavização enquanto estiver seguindo o alvo. Ademais, disso é possível definir limites que a câmera poderá se mover tanto verticalmente quanto horizontalmente, há também possibilidade de especificar limites dentro da visão da câmera que servem para representar a distância que o alvo pode ficar do centro da câmera. Também é possível configurar o tempo que a câmera deverá esperar até centralizar o alvo caso este pare de se movimentar.

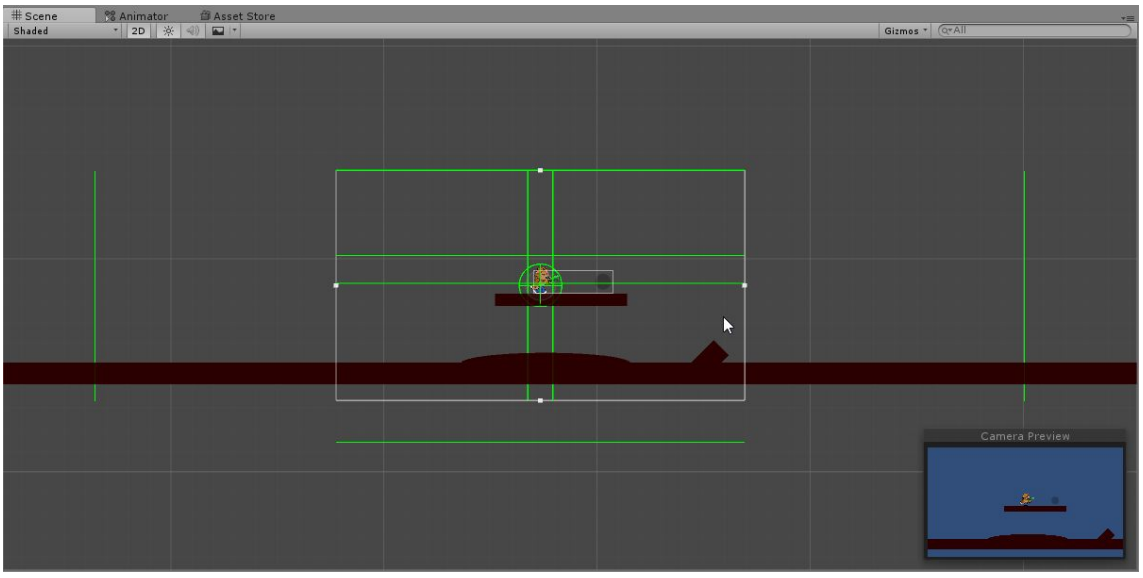


Figura 29 - Representação visual do sistema de movimentação da câmera

As propriedades relacionadas às configurações de limite são representadas pelas linhas verdes conforme pode ser visto na figura 29. O sistema foi idealizado dessa maneira para facilitar o ajuste dos limites e visualização dos mesmos, dando assim mais agilidade na implementação desse recurso no jogo.

3.4.4. GERAÇÃO DE CENÁRIO

Como o jogador deverá realizar diversas caçada do mesmo tipo de monstro durante todo o jogo, foi implementado o comportamento de gerar cenários aleatórios a cada a caçada. Dessa forma, mesmo que o jogador repita várias vezes uma caçada em um cemitério, a aparência desse cemitério poderá ser diferente a cada jogada.

Para que isso fosse possível, foi programado um sistema que funciona através do reconhecimento de pedaços do cenário. O sistema sorteia esses pedaços para poder gerar o cenário final.

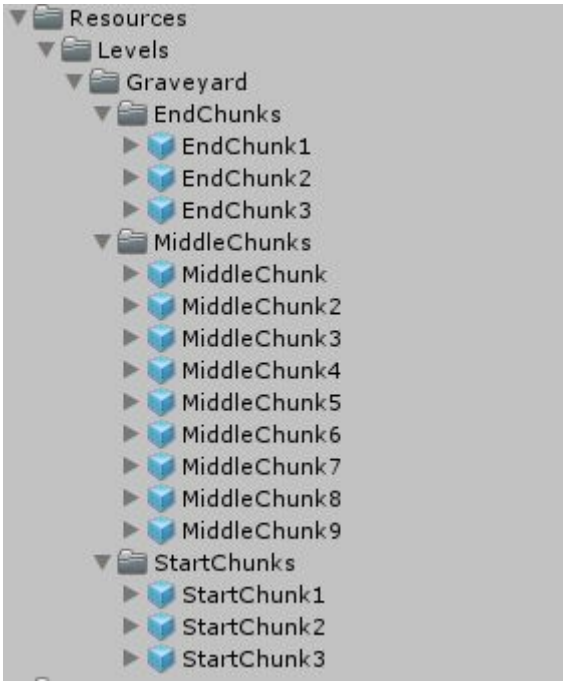


Figura 30 - Lista de objetos que representam os pedaços de um cenário

Na Figura 30 pode ser observado uma estrutura de pastas e objetos que representam os pedaços de um dos cenários do jogo. Essa estrutura possui 3 pastas principais: a “StartChunks” que representa os pedaços que podem iniciar o cenário, a “MiddleChunks” que representa os pedaços que são utilizados para preencher todo o meio do cenário e, por último, a “EndChunks” que representa as opções para terminar o cenário.

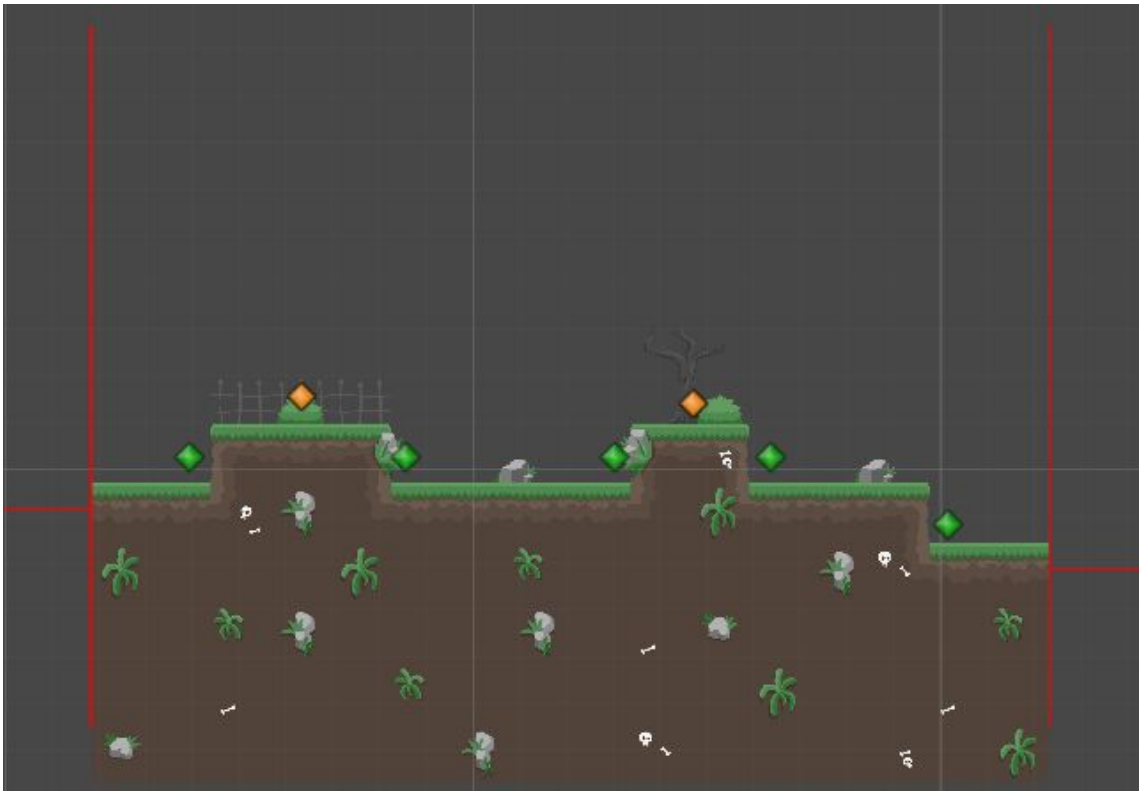


Figura 31 - Representação visual de um pedaço do cenário

Conforme pode ser visto na figura 31, além desses possuírem a imagem de um pedaço do cenário, eles também possuem alguns objetos que não aparecem no cenário mas servem para que o sistema consiga gerar o cenário corretamente. As linhas vermelhas servem para representar os limites e a altura desse pedaço, isso serve para que seja possível identificar onde o outro pedaço deve ser inserido sem que tenha problema no visual do jogo. Os losangos de cor laranja servem como opções em que os inimigos poderão aparecer. Já os de cor verde servem para disparar eventos quando algum inimigo encostar nele. Esses eventos indicam quando um inimigo deve pular ou que ele deve executar alguma outra ação específica.

3.4.5. PARALLAX - EFEITO DE PROFUNDIDADE DO CENÁRIO

É comum que mesmo em um jogo 2D o cenário busque passar uma sensação de profundidade dos seus elementos, isso é conhecido como efeito

Parallax. Caso o jogo tivesse uma câmera fixa e as imagens do fundo ficassem sempre paradas, esses efeitos poderiam ser feito utilizando apenas a parte artística para criar os elementos parecendo ter essa profundidade. Porém, quando um jogo possuí uma câmera ou cenário que se movimenta durante o jogo, é preciso que os elementos da tela se movam em velocidade diferentes para que a sensação de profundidade não se perca devido a esse movimento.

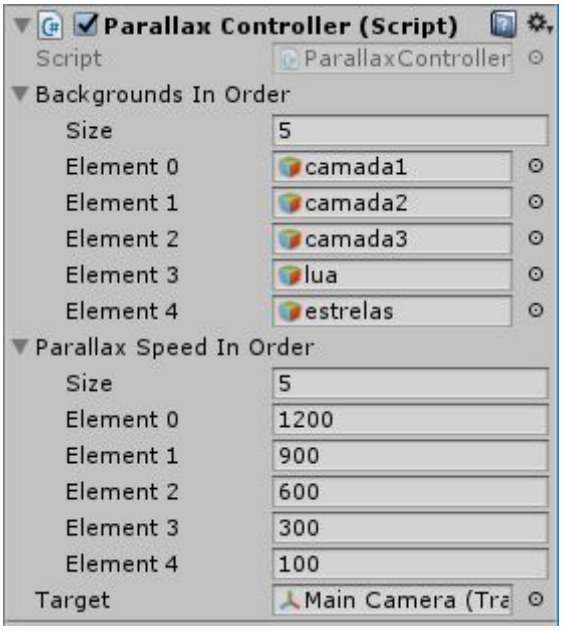


Figura 32 - Propriedades do sistema de controle de profundidade do cenário

Dessa forma, foi programado o sistema para controlar a profundidade do cenário e que pode ser visto na figura 32. Nas suas propriedades devemos indicar elementos do fundo do cenário que precisam se movimentar em velocidades diferentes. Além disso, é preciso especificar uma velocidade que será aplicada a cada objeto que foi identificado anteriormente e, por último, é identificado o alvo que serve para mostrar a direção em que a tela está se movendo e, assim, identificar se é preciso ou não movimentar os elementos, caso seja preciso aplicar a velocidade indicada na direção correta.

CAPÍTULO

4. RESULTADOS

4.1. ORGANIZAÇÃO

A produção e utilização do GDD se mostrou muito útil para termos uma visão geral do jogo e conseguirmos analisar, detalhar e modificar diversas mecânicas antes mesmo de começar a produzi-las. Além disso, facilitou a identificação e planejamento de todas as tarefas para a produção do jogo.

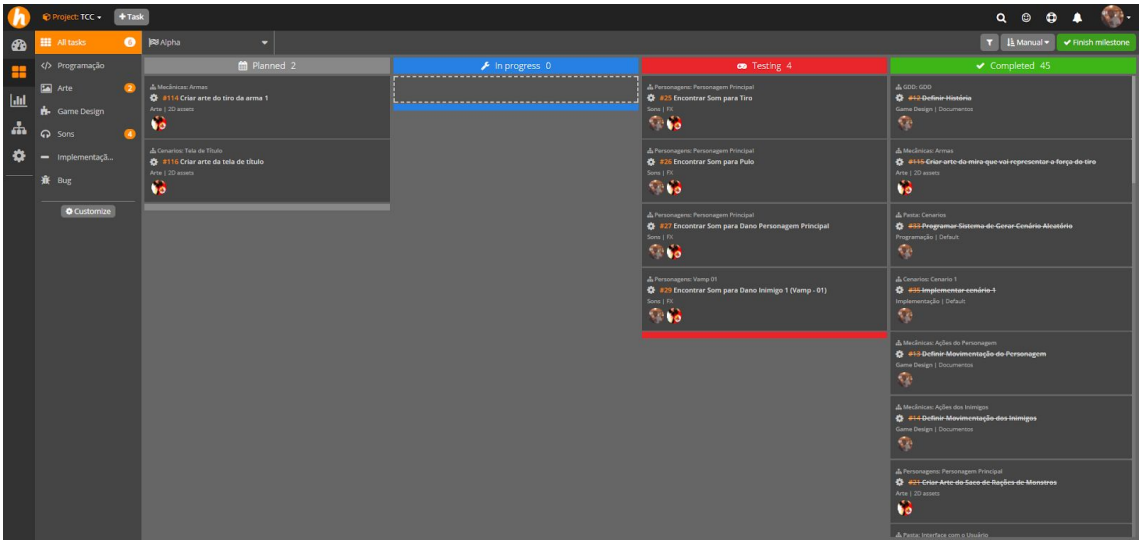


Figura 33 - Tela do Hacknplan com as tarefas planejadas

O Hacknplan se mostrou uma ferramenta muito eficiente para a organização e gestão das tarefas do jogo, conforme pode ser visto na figura 33. Através dele, foi possível organizar as tarefas que foram identificadas no GDD, de forma que pudéssemos gerenciar as etapas de produção. Também foi possível indicar se as tarefas estavam ou não em execução e se tinha algum problema impedindo o andamento da mesma. Dessa forma, foi possível prever e resolver problemas de forma rápida e eficiente, agilizando a produção do jogo.

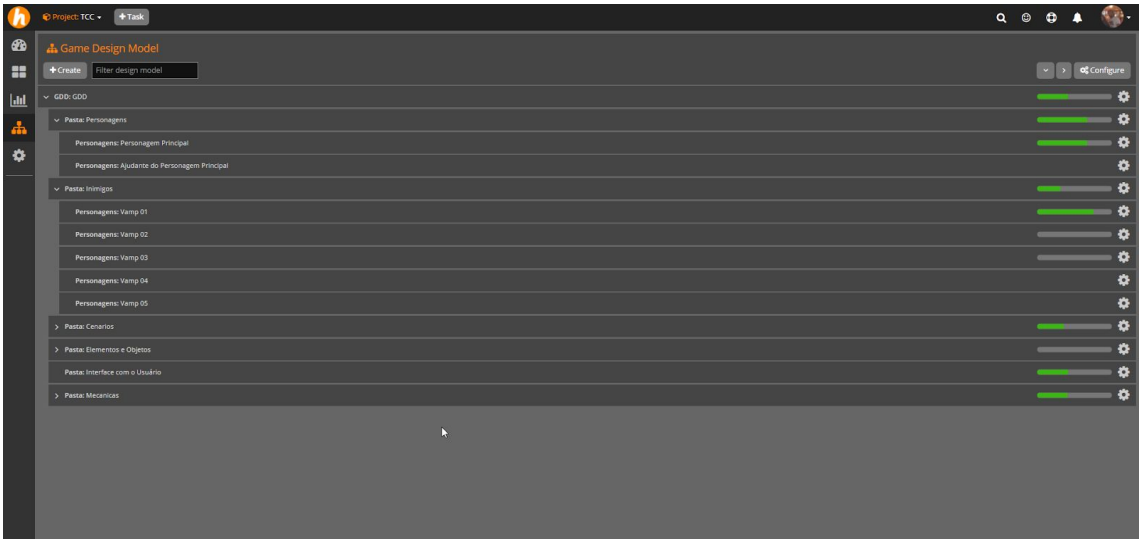


Figura 34 - Visão do Game Design Model do Hacknplan

Além de servir para visualizar as tarefas, o Hacknplan possui a ferramenta Game Design Model, que pode ser observada na figura 34. Essa ferramenta permite uma visão das tarefas agrupadas pelos elementos do GDD, sendo assim possível ter uma visualização geral do andamento de todas as características do jogo.

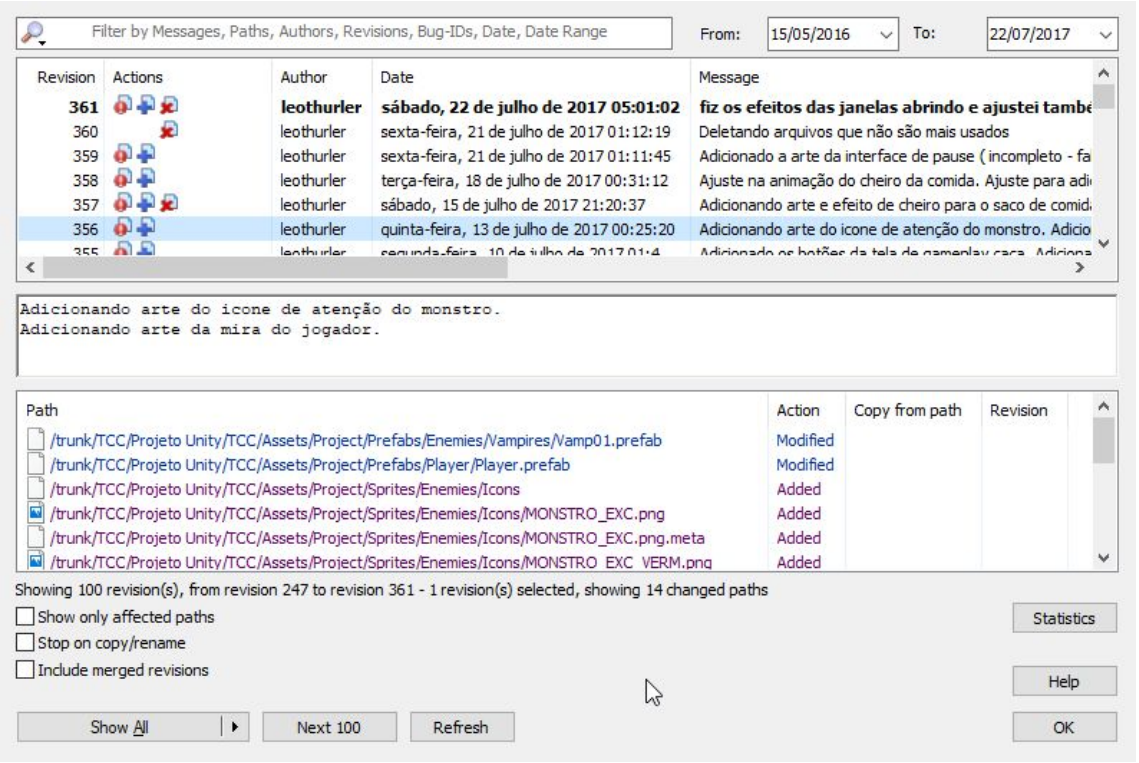


Figura 35 - Tela de log de alterações do SVN.

A utilização do repositório SVN através do site Assembla permitiu um controle eficiente das alterações que foram realizadas à medida que o projeto avançava, como pode ser visto na figura 35. Dessa forma, foi possível identificar quando uma alteração quebrava o funcionamento de alguma parte do jogo e também foi simplificado a atualização dos arquivos do projeto em outras máquinas, facilitando o trabalho em equipe.

4.2. PRODUTO FINAL

No desenvolvimento do jogo foram utilizadas diversas ferramentas e recursos mencionados anteriormente nesta dissertação. Dessa forma, a seguir será apenas apresentado e analisado o produto final que foi obtido.

4.2.1. INTERFACE

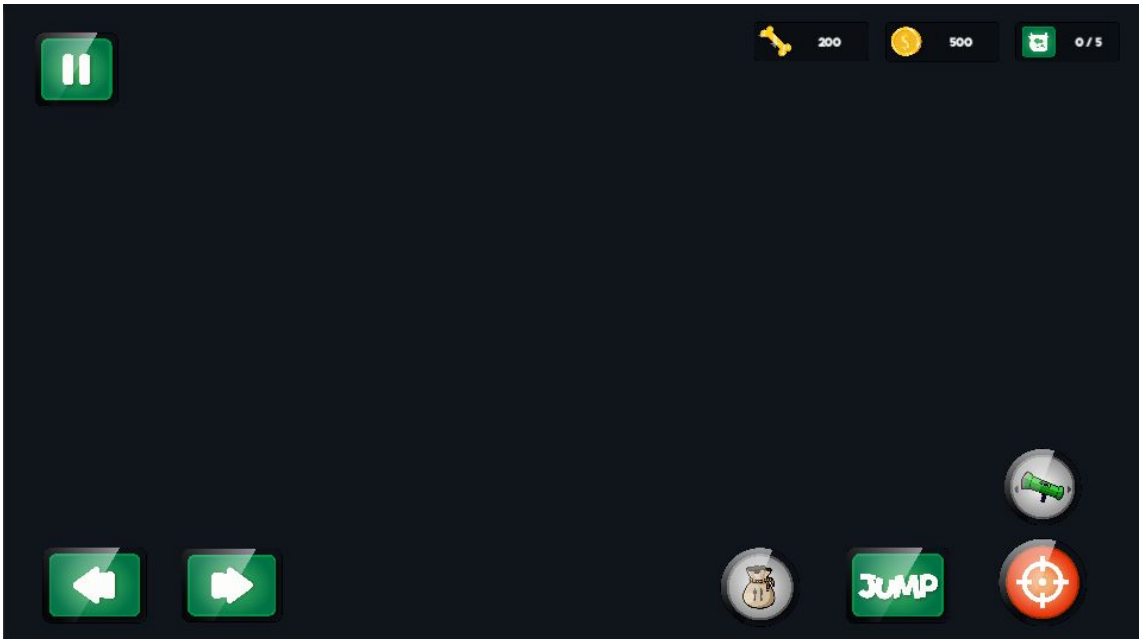


Figura 36 - Resultado interface da tela gameplay caça.

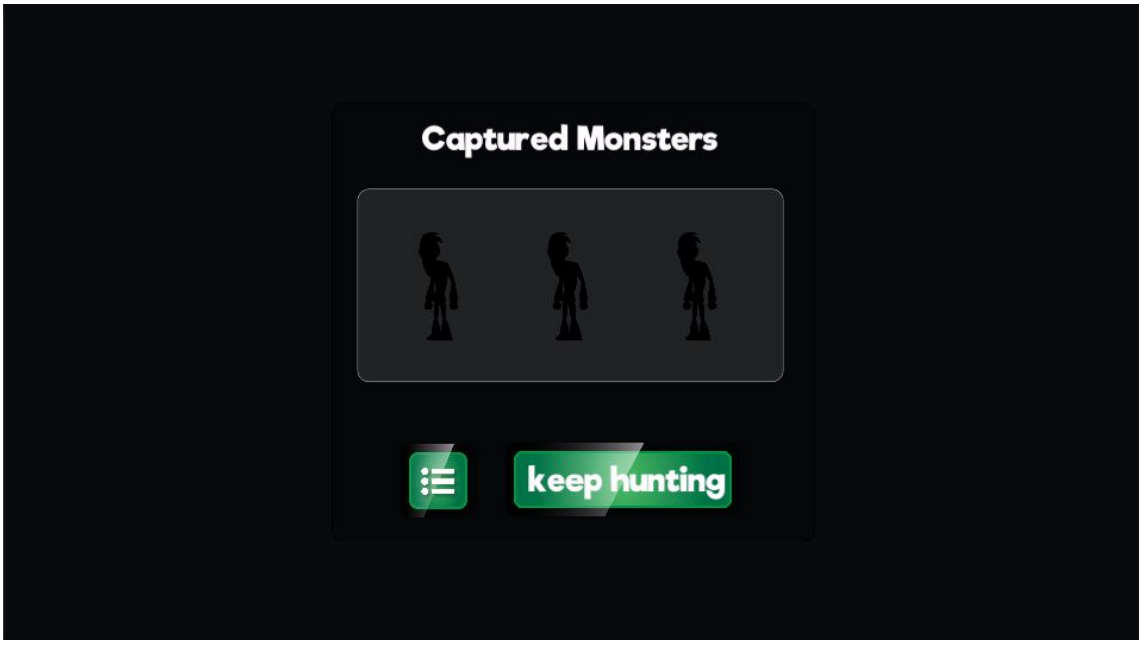


Figura 37 - Resultado interface fim de jogo da tela gameplay caça.

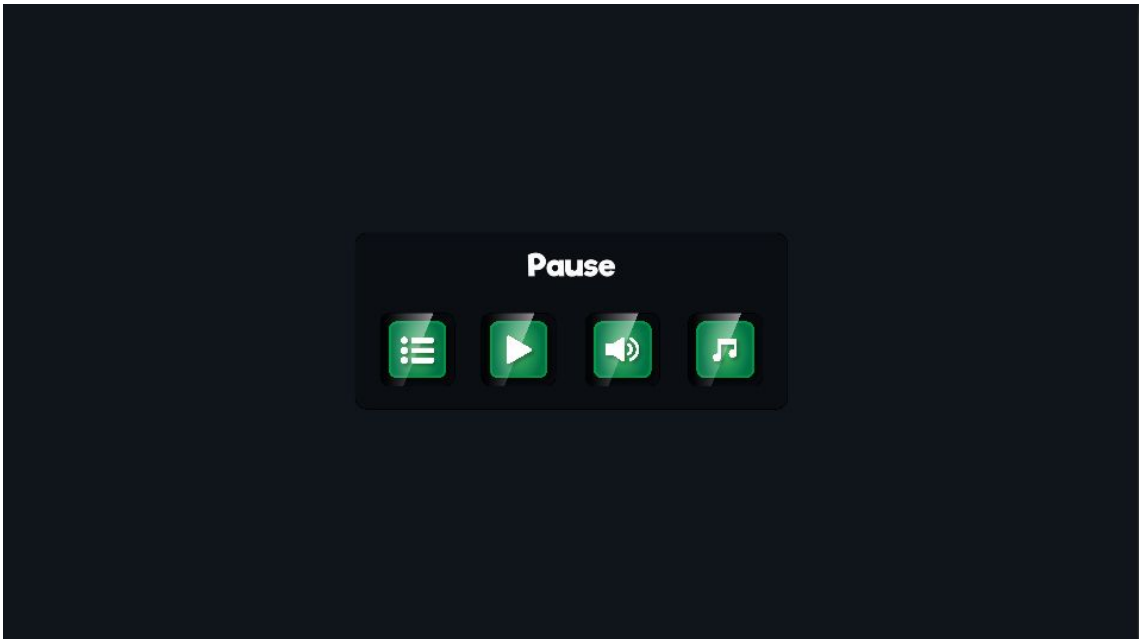


Figura 38 - Resultado interface pause game da tela gameplay caça.

As figuras acima apresentam os resultados obtidos através da criação dos elementos para a interface do jogo.

4.2.2. PERSONAGENS

Figura 39 - Resultado personagem principal.

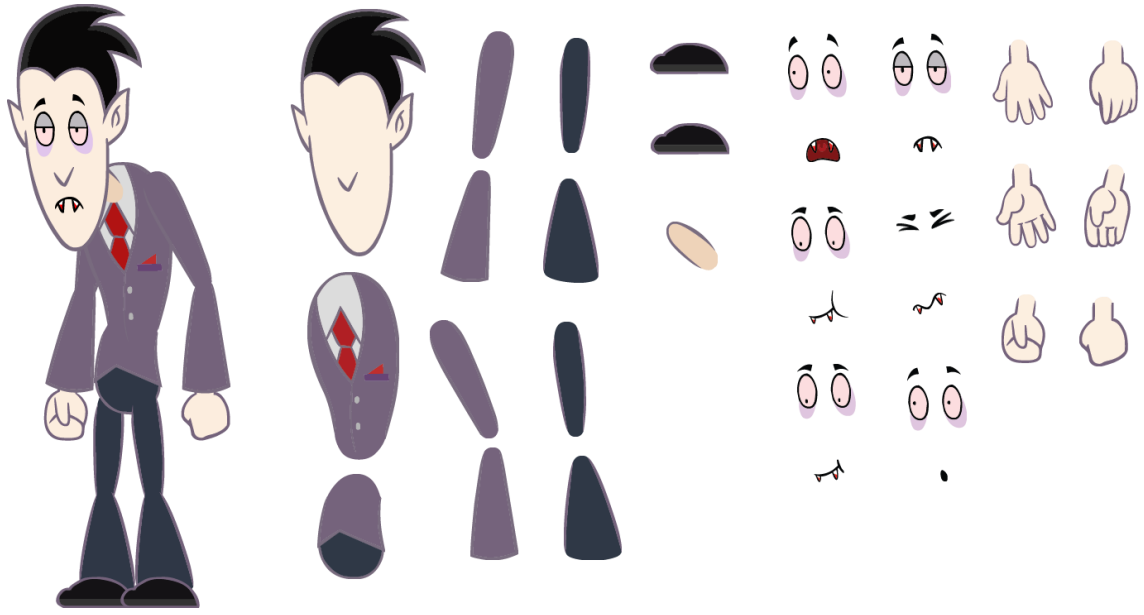


Figura 40 - Resultado inimigo vamp-01.

As figuras acima apresentam os resultados obtidos através da criação dos elementos dos personagens do jogo.

4.2.3. CENÁRIOS

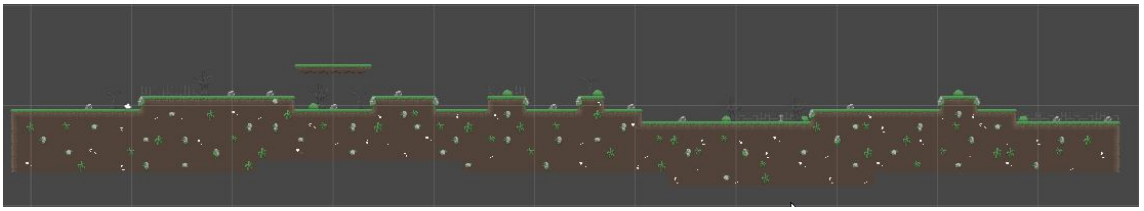


Figura 41 - Exemplo 1 de cenário gerado.

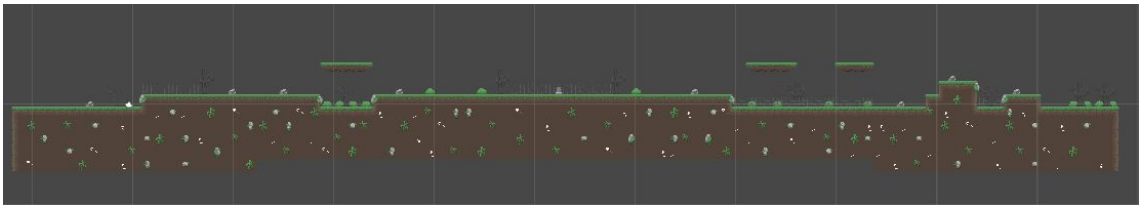


Figura 42 - Exemplo 2 de cenário gerado.

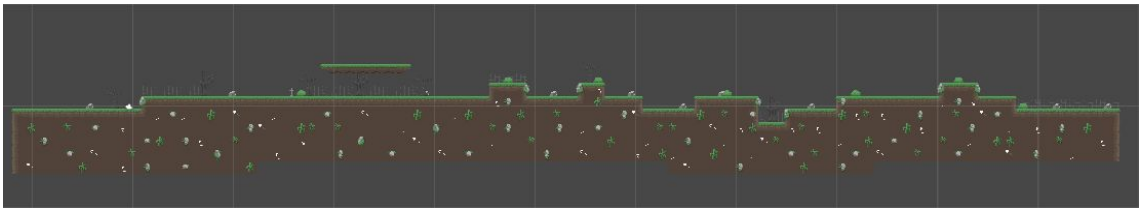


Figura 43 - Exemplo 3 de cenário gerado.

Figura 44 - Fundo do cenário.

As figuras acima apresentam os resultados obtidos através da criação dos elementos dos cenários do jogo. Elas demonstram exemplos de cenário devido ao

mecanismo de geração aleatória e como pode ser visto, existe uma variação em cada exemplo demonstrado.

4.2.3. FOTOS DO JOGO

Figura 45 - Foto 1 do jogo.

Figura 46 - Foto 2 do jogo.

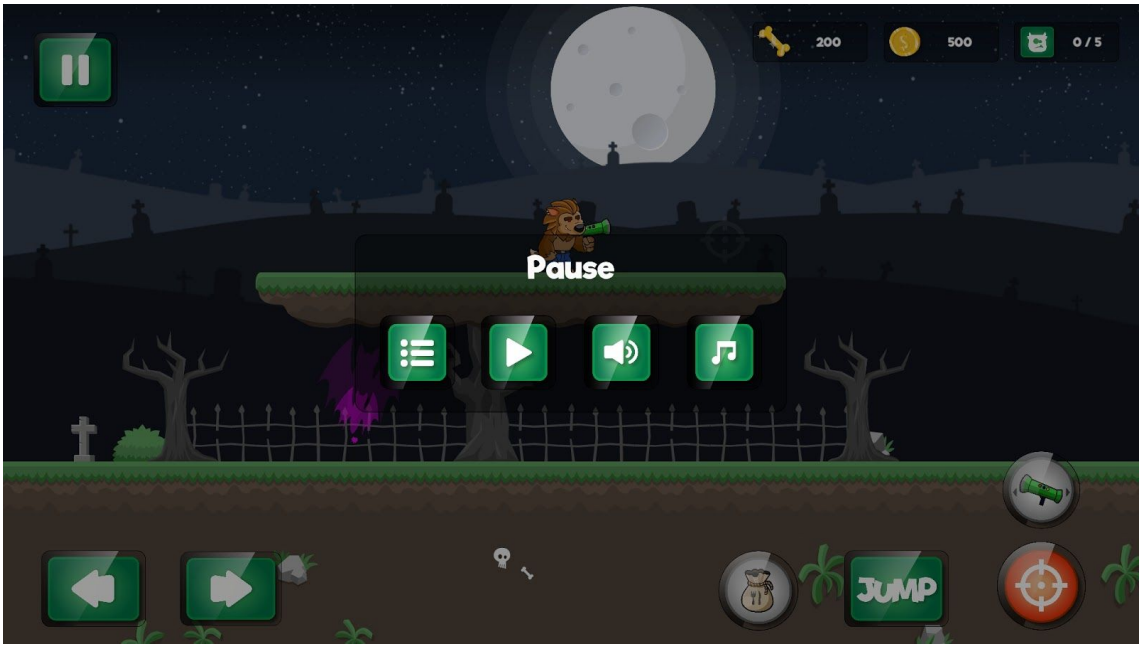


Figura 47 - Foto 3 do jogo.

Figura 48 - Foto 4 do jogo.

As figuras acima apresentam as fotos do jogo sendo executado através de um celular com o sistema Android. Através delas pode ser visto a qualidade do resultado obtido ao final do desenvolvimento desta dissertação.

CAPÍTULO

5. CONCLUSÃO

No presente trabalho, optou-se por construir um jogo 2D devido ao conhecimento e tempo disponível dos autores desta dissertação que também foram os responsáveis pela produção de todo o conteúdo do jogo. Uma vez que ambos já possuíam experiência com produção de jogos e artes em 2D, foi possível atingir um resultado de qualidade ao final da produção.

Sendo assim, o primeiro passo deste trabalho foi definir o escopo do jogo que seria produzido e, de acordo com as características do jogo, definir as ferramentas que seriam utilizadas no gerenciamento do projeto e na produção. Por último, analisar o resultado obtido no jogo através do uso das ferramentas escolhidas.

A escolha e o uso das ferramentas para auxiliar a produção se mostraram de suma importância para atingir o resultado desejado. As ferramentas para a gestão fizeram com que a produção do jogo ocorresse de forma mais organizada e eficiente e, dessa forma, foi possível otimizar a organização e ter mais tempo para a produção em si. Já as ferramentas de desenvolvimento se mostraram muito eficiente agilizando tanto a criação dos elementos visuais, como ilustrações e animações dos elementos do jogo, quanto aos elementos que atuam em uma camada que não é visível pelo consumidor final, como controlar a movimentação dos personagens e guardar os dados de uma partida do jogo.

Além disso, o uso de ferramentas mais populares veio com o benefício de ter uma comunidade ativa de desenvolvedores e artistas que as utilizam. Isso proporcionou uma forma bem ágil e eficiente de resolução de problemas que foram encontrados no decorrer do projeto, uma vez que muitas desses problemas já haviam sido resolvidos por outras pessoas da comunidade.

Devido ao tamanho do projeto e ao tempo disponível para a produção desta dissertação, a versão do jogo que foi produzida é uma demonstração do jogo contendo alguns dos principais aspectos que o jogo final deve possuir.

Finalmente, pelos resultados apresentados de produzir a versão de demonstração do jogo Monster Chef, analisando e gerenciando toda a produção dele de forma independente, foi possível constatar a viabilidade da produção do jogo completo definir métricas para medir o tempo de produção.

5.1. TRABALHOS FUTUROS

Para trabalhos futuros, recomenda-se:

1. Criação da arte e mecânicas do sistema de lojas e equipamentos.
2. Criação da arte e mecânicas dos novos inimigos.
3. Criação da arte e implementação dos novos cenários.
4. Desenvolvimento dos sistemas de metagame.
5. Construção de um trailer promocional.

REFERÊNCIAS BIBLIOGRÁFICAS

ROGERS, Scott. Level Up: Um guia para o design de grandes jogos. 1. ed. São Paulo: Blucher, 2012, 494p.

SALEN, Katie.; ZIMMERMAN, Eric. Regras do jogo: Fundamentos do design de jogos: principais conceitos: volume 1. 1. ed. São Paulo: Blucher, 2012, 167p.

SALEN, Katie.; ZIMMERMAN, Eric. Regras do jogo: Fundamentos do design de jogos: regras: volume 2. 1. ed. São Paulo: Blucher, 2012, 229p.

SALEN, Katie.; ZIMMERMAN, Eric. Regras do jogo: Fundamentos do design de jogos: interação lúdica: volume 3. 1. ed. São Paulo: Blucher, 2012, 258p.

SALEN, Katie.; ZIMMERMAN, Eric. Regras do jogo: Fundamentos do design de jogos: cultura: volume 4. 1. ed. São Paulo: Blucher, 2012, 153p.

SCHELL, Jesse. The Art of Game Design: a Book of Lenses. 1. ed. Burlington: Elsevier, 2008, 489p.

UNITY. Perguntas frequentes (FAQ). Disponível em: <https://unity3d.com/pt/unity/faq >. Acesso em 17 de junho de 2017.

UNITY. Unity Manual. Disponível em: <https://docs.unity3d.com/Manual/index.html>. Acesso em 18 de junho de 2017.

GIT. Primeiros passos - Sobre Controle de Versão. Disponível em: <https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o>. Acesso em 18 de junho de 2017.

RUTE, ANA. Wireframe, protótipo e mockup – Qual a diferença?. Disponível em: <https://anarute.com/wireframe-prototipo-e-mockup-qual-a-diferenca/>. Acesso em 18 de junho de 2017.

PEREIRA, ANA. O que é Wireframe?. Disponível em: <https://www.tecmundo.com.br/programacao/976-o-que-e-wireframe-.htm/>. Acesso em 18 de junho de 2017.

COUTINHO, CAMILO. Afinal, o que é um wireframe?. Disponível em: <http://digitaisdomarketing.com.br/afinal-o-que-e-um-wireframe/>. Acesso em 18 de junho de 2017.

MOZILLA DEVELOPER NETWORK. Tiles and tilemaps overview. Disponível em: <https://developer.mozilla.org/en-US/docs/Games/Techniques/Tilemaps/>. Acesso em 04 de julho de 2017.

UNITY TUTORIAIS. Aprimorando o conhecimento: Jogos em Tiles. Disponível em: <http://tutoriaisunity.blogspot.com.br/2014/03/aprimorando-o-conhecimento-jogos-em.html/>. Acesso em 04 de julho de 2017.

BERNADO, KLEBER. O que são métodos ágeis?. Disponível em: <https://www.culturaagil.com.br/o-que-sao-metodos-ageis//>. Acesso em 05 de julho de 2017.

KNOWLEDGE 21. O Manifesto Ágil. Disponível em: <http://www.knowledge21.com.br/sobreagilidade/agilidade/o-manifesto-agil/>. Acesso em 05 de julho de 2017.

HACKNPLAN. (FAQ). Disponível em: <http://hacknplan.com/faq/>. Acesso em 15 de julho de 2017.

HACKNPLAN. Pricing. Disponível em: <http://hacknplan.com/#pricing>. Acesso em 15 de julho de 2017.

MAP EDITOR. Tiled Documentation. Disponível em: <http://doc.mapeditor.org/#tiled-documentation>. Acesso em 15 de julho de 2017.

BRASH MONKEY. Spriter Features. Disponível em: <https://brashmonkey.com/spriter-features/>. Acesso em 15 de julho de 2017.

NINJAMOCK. Frequently asked questions. Disponível em: <https://ninjamock.com/shared/purchasingFAQ/>. Acesso em 15 de julho de 2017.

ASSEMBLA. Princing. Disponível em: <https://www.assembla.com/pricing/>. Acesso em 15 de julho de 2017.

ADOBE. Preços e planos de associação à Creative Cloud. Disponível em: <https://www.adobe.com/br/creativecloud/plans.html> . Acesso em 15 de julho de 2017.

UOL. Em duas semanas, "The Witcher 3: Wild Hunt" vende 4 milhões de cópias. Disponível em: <https://jogos.uol.com.br/ultimas-noticias/2015/06/09/em-duas-semanas-the-witcher-3-wild-hunt-vende-4-milhoes-de-copias.htm> . Acesso em 22 de julho de 2017

G1. Com US$ 1 bilhão em 3 dias, 'GTA V' quebra novo recorde de faturamento. Disponível em: <http://g1.globo.com/tecnologia/games/noticia/2013/09/com-us-1-bilhao-em-3-dias-gta-v-quebra-novo-recorde-de-faturamento.html> . Acesso em 22 de julho de 2017.

TECHTUDO. Pokémon Go Plus: Apple Watch ganha versão exclusiva do jogo. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2016/09/pokemon-go-ganhara-uma-nova-versao-para-apple-watch.html> . Acesso em 22 de julho de 2017.

TECHTUDO. Entenda o que são jogos indies e confira os principais títulos já lançados. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2012/10/entenda-o-que-sao-jogos-indies-e-confira-os-principais-titulos-ja-lancados.html> . Acesso em 22 de julho de 2017.