**RAFAEL HIDEAKI TERUYA**
**TIAGO SUEDA LIMONE**

# GERAÇÃO PROCEDURAL DE CONTEÚDO APLICADA EM UM JOGO INCREMENTAL

São Paulo
2018

**RAFAEL HIDEAKI TERUYA**
**TIAGO SUEDA LIMONE**

# GERAÇÃO PROCEDURAL DE CONTEÚDO APLICADA EM UM JOGO INCREMENTAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

São Paulo
2018

**RAFAEL HIDEAKI TERUYA**
**TIAGO SUEDA LIMONE**

# GERAÇÃO PROCEDURAL DE CONTEÚDO APLICADA EM UM JOGO INCREMENTAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

Área de Concentração:

Tecnologias Interativas

Orientador:

Prof. Dr. Ricardo Nakamura

São Paulo
2018

Prof. Dr. Ricardo Nakamura

Dedicamos este projeto a todos os amigos e familiares da dupla.

# AGRADECIMENTOS

Agradeço aos meus pais, Edson e Mika, e às minhas irmãs, Yuri e Ákina, por tudo o que fizeram por mim para que eu chegasse onde estou hoje. Além disso, gostaria de agradecer a todos os meus amigos pelo apoio e pela paciência este ano. Por fim, gostaria de agradecer ao professor Nakamura, nosso orientador, pelos conselhos e disposição de tempo para nos atender.

- Rafael Hideaki Teruya

Agradeço aos meus pais Marta e Antonio, que durante toda a minha vida providenciaram suporte para que pudesse chegar onde cheguei - formando-me em uma das escolas de engenharia mais desejadas do país. Agradeço à minha namorada Bryna, que durante toda a graduação me apoiou em minhas decisões e me forneceu suporte nos momentos de desespero e frustrações ao longo do curso, além de comemorar todas as conquistas. Agradeço também ao nosso orientador, Ricardo Nakamura, pela paciência e disponibilidade durante a orientação. Por fim, agradeço ao pessoal da VR Monkey, por me introduzir ao mundo de desenvolvimento de jogos pelo qual me apaixonei.

- Tiago Sueda Limone

# RESUMO

As mudanças no estilo de vida das pessoas, cada vez mais ocupadas e realizando múltiplas tarefas ao mesmo tempo, vem popularizando o gênero de jogos incrementais, que exigem menos foco e atenção dos jogadores, principalmente em plataformas móveis e web. Apesar de existirem diversos títulos do gênero, alguns muito bem sucedidos comercialmente, há sempre uma preocupação constante com a variedade limitada de conteúdo oferecida para o jogador, exigindo atualizações constantes para manter o jogo vivo. Nesse contexto, para reduzir custos de manutenção e até mesmo proporcionar maior rejogabilidade, uma ferramenta bastante explorada em outros gêneros, porém pouco explorada em jogos incrementais, é a geração procedural de conteúdo. O objetivo desse projeto é unir esses dois conceitos, buscando uma experiência agradável, sem fim e menos repetitiva para o jogador. Como forma prática de representação da união dos conceitos, foi desenvolvido um jogo em Unreal Engine 4, com foco em plataformas mobile (porém não restrito a elas), composto por dois núcleos distintos e interligados, alimentados por conteúdo gerado proceduralmente. O primeiro núcleo consiste em um *manager*, no qual o jogador deve obter e gerenciar recursos sobre seu terreno, construindo e recrutando personagens para auxiliá-lo. O segundo núcleo é uma etapa de combates no estilo RPG de ação em *dungeons* geradas proceduralmente.

**Palavras-Chave** – Incremental, Geração procedural, Jogos, Mobile, Multiplataforma.

# ABSTRACT

Changes on people's lifestyle, making them always busier and multitasking, have been popularizing the incremental games genre, which demand less focus and attention from its players, mainly on mobile and web platforms. Even though there are many different titles of the genre, some of them even very successful, there is always this constant worry with the limited range of content that can be offered to the player, requiring frequent maintenance and updates to keep the game alive. In this context, to reduce maintenance costs and even promote bigger replayability, a tool largely explored in other genres, but forgotten on incremental games, is the procedural content generation. The objective of this project is to combine both these concepts, seeking a pleasant, endless and less repetitive experience for the players. As a form of practical representation of both concepts' union, a game was developed using Unreal Engine 4, focusing on mobile platforms (though not restricted to them), composed of two different and interconnected cores which consume procedural generated content. The first one consists of a manager, in which the players must obtain and manage their resources over a terrain, building and recruiting characters to aid in their goals. The second one is the combat stage in an action RPG style through procedural generated dungeons.

**Keywords** – Incremental, Procedural generation, Games, Mobile, Multiplatform.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# GLOSSÁRIO

***Balanceamento*** - Conceito associado a estabelecer um nível de dificuldade equilibrado correspondente à experiência do jogador esperada

***Bioma*** - Classificação de região com um conjunto bem definido de características como vegetação e fauna

***Core Loop*** - Conjunto de ações que determinam o ritmo e o modo como o jogo irá progredir

***Dungeon*** - Às vezes chamada de masmorra, é um local de exploração e coleta de recursos

***Grid*** - Uma matriz que representa a discretização de um espaço

***Inteligência Artificial*** - No contexto de jogos, a inteligência artificial é associada ao comportamento de um personagem controlado pelo computador

***Level*** - Um indicativo do nível de experiência/habilidade do jogador

***Manager*** - Um modo do jogo que consiste em gerenciar, mover e modificar

***PCG*** - Geração Procedural de Conteúdo (Procedural Content Generation) é o nome dado ao processo de criação de conteúdo por parte de um programa de computador

***Playthrough*** - Jogar um jogo do início ao fim ou, em jogos incrementais, entre resets

***Recursos*** - Propriedades que o jogador possui que são necessárias para realizar a maioria das ações no jogo

***Reset*** - Volta do jogo para o estado inicial. Em jogos incrementais, esta opção vem acompanhada de facilidades na evolução do jogador.

***Tile*** - A unidade básica que compõe um grid

***Wall*** - Ponto na progressão do jogo em que o jogador permanece estagnado.

# SUMÁRIO

# PARTE I

## ESPECIFICAÇÃO DO PROJETO

# 1 INTRODUÇÃO

## 1.1 Objetivo

O objetivo óbvio dos videogames é entreter as pessoas surpreendendo-as (1). Com isto em mente, o objetivo deste trabalho de conclusão de curso é desenvolver um jogo incremental, que traz ao jogador um sentimento de recompensa ao receber um número exponencialmente crescente de recursos para investir e de descoberta de novas mecânicas e experiências à medida em que progride, unido ao uso de geração procedural de conteúdo, para que estas experiências do jogador possam ser sempre diferentes, provocando surpresa e, consequentemente, prolongando-as.

## 1.2 Motivação

O gênero incremental tem vivido nos últimos anos um grande crescimento de popularidade - basta observar a lista de jogos mais jogados em diversas plataformas de jogos, principalmente mobile e web, como Play Store, Kongregate e Armor games, mas também atingindo plataformas tradicionais como a Steam. Interessante notar também que o gênero tem ganhado jogadores tanto casuais como hardcores. Isso se deve, principalmente, a dois fatores comportamentais humanos: o primeiro, mais histórico, é o sentimento de recompensa - humanos gostam de observar progresso e sua quantidade de recursos disponível crescendo. O segundo e mais conectado à atualidade é a possibilidade de progredir e se divertir com o jogo enquanto se realiza outras tarefas, em qualquer lugar. Por exigir pouca atenção do jogador, o gênero promove um sentimento de recompensa crescente enquanto o jogador trabalha, estuda ou executa quaisquer outras atividades do seu cotidiano. (2)

Por outro lado, para reter os usuários, os jogos desse gênero necessitam de geração de conteúdo adicional, por meio de manutenções frequentes. Nesse sentido, uma ferramenta que pode reduzir os custos de manutenção é a geração procedural de conteúdo, frequentemente utilizada em jogos de diferentes gêneros.

## 1.3 Justificativa

Atualmente, existe um grande número de jogos presentes nas mais diversas plataformas. Devido a este fator, tem se tornado cada vez mais difícil manter um usuário ativo em um jogo por muito tempo, uma vez que a jogabilidade é repetitiva e isto, por sua vez, causa tédio ao jogador e o incentiva a buscar outras opções no mercado.

Por este motivo, é importante que os desenvolvedores de jogos inovem a todo momento e aumentem o tempo de atividade dos jogadores. Os conceitos explorados neste trabalho visam solucionar este problema, podendo trazer uma nova tendência para o mundo dos jogos.

## 1.4 Organização do Trabalho

O trabalho está dividido em 5 partes inicialmente: Introdução, Aspectos Conceituais, Tecnologias Utilizadas, Metodologia do Trabalho e Especificação de Requisitos do Sistema. Na Introdução, é feito um breve resumo para que o leitor tenha noção do que se trata o trabalho e dos problemas que resolve. Em Aspectos Conceituais, serão apresentados os conceitos empregados no projeto (Geração Procedural de Conteúdo, Jogo Incremental). Em Tecnologias Utilizadas, citaremos as tecnologias utilizadas no desenvolvimento do jogo e os motivos pelos quais foram escolhidas. Em Metodologia do Trabalho, será apresentado um cronograma , que dividirá o projeto em fases, para melhor organização. Por fim, em Especificação de Requisitos do Sistema, serão definidas quais características são necessárias para garantir um bom desempenho do projeto

# 2 ASPECTOS CONCEITUAIS

## 2.1 Geração Procedural de Conteúdo

Geração Procedural de Conteúdo, ou Procedural Content Generation (PCG), está relacionada à criação de conteúdo de um jogo com entradas diretas ou indiretas de usuário, por meio de um algoritmo. Em outras palavras, baseia-se em um software para criar conteúdo por conta própria, podendo opcionalmente, ter auxílio de outros jogadores ou desenvolvedores.

Um conceito importante para que se entenda de fato a definição de PCG é o "conteúdo". Tudo o que está contido em um jogo pode ser considerado conteúdo: mapas, regras de jogo, texturas, histórias, itens, missões, armas, recursos, personagens, níveis, entre outros. É importante lembrar que o conceito de conteúdo de um jogo pode abranger outros aspectos não citados anteriormente, mas trabalharemos inicialmente com esta definição.

Por fim, serão utilizadas duas listas contidas no livro Procedural Content Generation in Games de Noor Shaker, Julian Tagelius e Mark J. Nelson (3), principal referência desse trabalho, para diferenciar e estabelecer os limites da PCG, na concepção dos autores. Primeiramente, são considerados PCG: uma ferramenta de software que cria masmorras para um jogo de aventura; uma ferramenta que popule rapidamente com vegetação o mundo de um jogo; um sistema que cria novas armas para seus jogadores baseando-se em outras armas bem avaliadas. Em outra via, não são considerados PCG: um editor de mapa para um jogo de estratégia que permite que o usuário posicione e remove os itens sem ter nenhuma ação de geração por si; jogador artificial de um jogo de tabuleiro; uma ferramenta capaz de integrar automaticamente uma vegetação gerada. É importante citar estas listas para definir concretamente o que é a geração procedural de conteúdo

## 2.2 Jogo

Dado que o objetivo final deste trabalho é a implementação de um jogo, este aspecto será formalizado nesta seção. Um jogo é definido como qualquer atividade que contenha regras e seja realizada por um ou mais participantes, com o objetivo de obter diversão ou entretenimento (4). Dentro do contexto em que estamos inseridos, o termo "jogo"será utilizado na maioria dos casos, em uma definição mais específica: a de um jogo eletrônico. Os jogos eletrônicos estão ganhando muita popularidade devido à grande ascensão da tecnologia e pelo fácil acesso proporcionado pelos aplicativos mobile, presentes em todos os smartphones atuais.

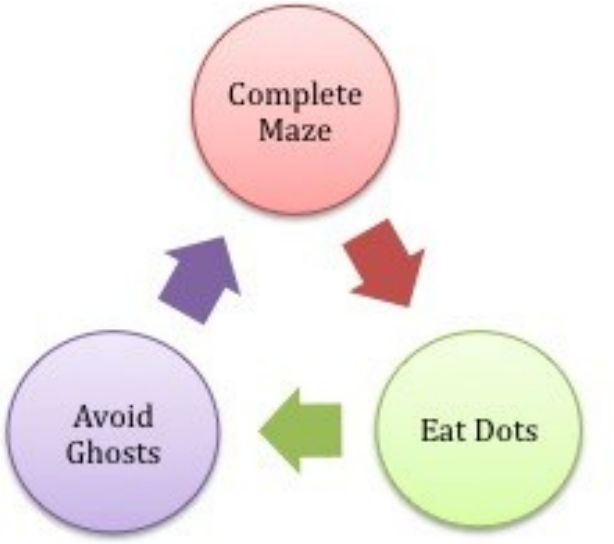
## 2.3 Core loop

Core loop é definido como um conjunto de ações que determinarão como e com que ritmo o jogo irá progredir (5) . Estas atividades fazem parte da estrutura do projeto no qual os jogadores estarão empenhando-se em repetir, em uma sequência de loop. É parte da essência do jogo, algo que não é possível remover sem alterar significativamente a experiência do usuário. Este aspecto será melhor explorado mais adiante, na seção de jogo incremental, uma vez que esta modalidade apresenta um loop bem definido.

Um bom core loop deve ser capaz de colocar o jogador em um estado de completa sintonia com o jogo. O usuário deve estar totalmente imerso e nada deve distrair sua atenção da tarefa a ser executada. Além disso, o desafio fornecido deve ser justo, no nível de dificuldade ideal à situação presente no momento. As pessoas gostam de se sentir no controle, de enfrentar situações que não sejam nem muito fáceis e nem muito difíceis (6).

É possível perceber uma relação de proporcionalidade entre a qualidade do core loop e o nível de retenção de jogadores: quanto mais o jogador se identificar e seguir com o loop proposto, mais a qualidade do jogo influenciará com a sua retenção. Nos jogos em geral, caso as mecânicas não sejam intuitivas ou os controles não estejam bem definidos, o jogador não se sentirá confortável e abandonará o jogo em minutos. É por este motivo que o core loop é a parte mais importante de um jogo.

Figura 1: Exemplo de core loop - Pacman




Fonte: Game Analysis -
http://jerrymomoda.com/the-core-loop-key-to-an-engaging-game/

## 2.4 Jogo incremental

Um jogo incremental é um tipo de jogo baseado no acúmulo de recursos, que explora o sentimento de recompensa de um jogador. Nesta modalidade há geralmente pelo menos um tipo de moeda ou número, que aumenta a uma taxa fixa que pode ou não ser definida pelo um input de um jogador. O input mais comum atualmente é o click, como em Cookie Clicker, Clicker Heroes, Cow Clicker, entre outros. Esta taxa de aumento também pode crescer exponencialmente em número e velocidade, de acordo com as ações que o jogador executa no jogo.

É perceptível a existência de um loop no jogo: o jogador acumula recursos, reinveste, acelera o modo como recebe os recursos e volta para o primeiro passo. Este loop define o gênero incremental e é isto que o difere de um jogo que simplesmente tem uma pontuação crescente. Outra característica marcante é que, os recursos aumentam mesmo quando o jogador não está presente no jogo. Isto faz com que não exista a necessidade do jogador interagir com o jogo a todo momento e, consequentemente, ser exposto a uma excessiva dose de repetições, que causa tédio e desistência.

Existe também um fundo psicológico por trás do sucesso dos jogos incrementais. É natural do ser humano o desejo pelo acúmulo e a aversão pela perda. O cérebro humano

é treinado para não gostar de perder recursos e, por outra via, nos dá um forte desejo de acumulá-los. Além disso, é agradável para o usuário visualizar números crescendo, por menor que seja a importância dos números em questão. Os jogos incrementais exploram estes conceitos e, por estes motivos, fazem tanto sucesso entre os usuários.

## 2.5 Grind

O termo grind está relacionado a quantidade de tempo gasto pelo jogador repetindo tarefas com o objetivo de evoluir no jogo, seja desbloqueando itens, melhorando personagens, coletando recursos, etc. Geralmente, envolve destruir um mesmo conjunto de oponentes para se fortalecer para uma nova etapa do jogo. O processo de grinding pode muitas vezes ser tedioso ao jogador, mas é um obstáculo pelo qual precisa passar para que possa evoluir e atingir novos objetivos. Neste sentido, é interessante o uso da geração procedural de conteúdo, para maior quantidade de conjuntos de inimigos a se enfrentar, diminuindo a repetitividade de desafios e consequentemente, a probabilidade de tédio do jogador.

## 2.6 Reset

O Reset é uma funcionalidade muito utilizada em jogos incrementais. A partir desta opção, o jogador perde todo o progresso no jogo em troca de algo que facilitará a sua nova jornada. Ou seja, todos os recursos conquistados voltam para a estaca zero mas, nesta nova etapa, o crescimento ocorre de maneira muito mais rápida e é possível reconquistar os recursos anteriores em menos tempo. Esta opção permite que o jogador evolua mais rápido e possa conquistar recursos de modo que, em etapas anteriores, são considerados inalcançáveis.

Utilizar este conceito no jogo a ser desenvolvido contribuirá para um dos objetivos principais do trabalho: proporcionar rejogabilidade infinita ao usuário. Uma vez que os resets são acionados, o jogador deve refazer seus passos , reaproveitando o conteúdo já utilizado. Além do mais, a cada reset, são mostrados novas metas e objetivos. Isto mostra ao jogador o quanto está evoluindo e incentiva-no a continuar jogando, muitas vezes, até de maneira mais determinada.

## 2.7 Playthrough

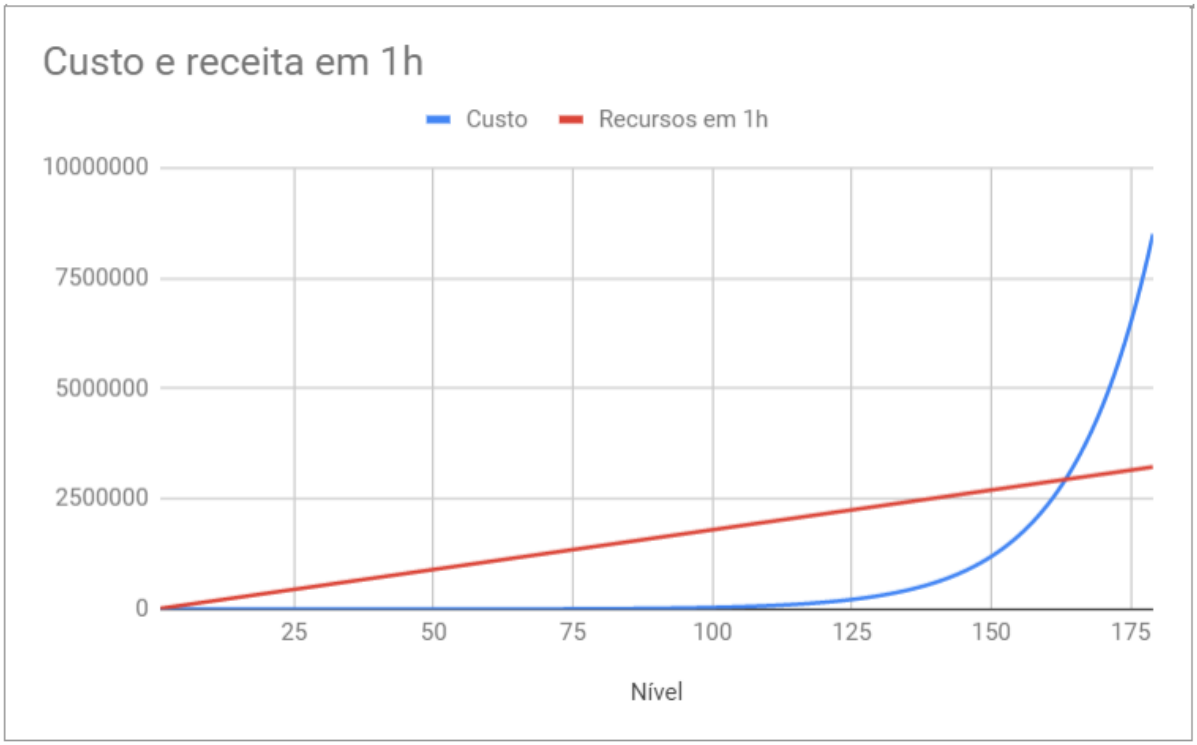
*Playthrough* em geral diz respeito a jogar um jogo do início ao fim. No contexto de jogos incrementais, *playthrough* consiste do período de jogo entre o início e um reset ou entre resets.

## 2.8 Wall

*Wall* ou barreira é o termo utilizado no contexto de jogos incrementais para um ponto na progressão do jogo em que o jogador necessita de um período muito longo de *grind* para poder continuar avançando. Normalmente, nesse ponto o jogador pode optar por um reset, que deve promover algum bônus que possibilite com que, no novo *playthrough*, a barreira anterior possa ser superada.

Tomando como exemplo um cenário em que um gerador possui custo inicial 50, renda por segundo 5 e cada incremento no gerador custa 7% mais do que o incremento anterior, aumentando a renda em 5, obtém-se o gráfico da figura 2.

Figura 2: Custo e Recursos por hora em função do nível do gerador




Fonte: Autores

Podemos observar que por volta do nível 160, o custo ultrapassa a receita por hora.

Supondo que a próxima meta imposta pelo jogo ao jogador seja adquirir um gerador nível 200, torna-se inviável atingir tal meta apenas esperando sua produção aumentar. Logo, o jogador atingiu uma barreira de progressão, ou *wall*. Supondo agora que um jogador que atingiu o nível 100 e reiniciou seu progresso consegue um bônus de 8 vezes na sua produção, obtemos o gráfico da figura 3.

Figura 3: Custo e Recursos por hora em função do nível do gerado, com reset


Fonte: Autores

Pode-se observar que o reset ocorreu no momento 100, em que o custo e a renda voltaram ao estado inicial, representando um *playthrough*. Durante o segundo *playthrough*, atingir o nível 200 com o gerador se tornou bem mais viável, visto que o custo apenas supera a quantidade de recursos por hora por volta do instante 300 (nível 200). Mesmo assim, é provável que um segundo reset em um momento intermediário fosse ainda mais eficiente, promovendo menos tempo de espera ao jogador.

## 2.9 Estilo de jogo idle ou ativo

A principal característica dos jogos idle é que não existe a necessidade de um input por parte do jogador para existir progresso no jogo. Ou seja, dependendo do estado de progresso em que se encontra, é possível construir monumentos que gerem recursos

somente com o passar do tempo (7). Esta modalidade de jogo combina com o jogo incremental, já que a união dos dois conceitos permite ao usuário realizar outras coisas em seu cotidiano e, mesmo assim, evoluir no jogo. Isso faz com que sempre exista a expectativa de jogar, pela evolução constante e, mesmo assim, não se prender totalmente ao jogo, diminuindo a probabilidade de cansaço e abandono.

Um estilo de jogo idle é adotado por um jogador que deseja evoluir enquanto o jogo roda em *background*, enquanto um estilo de jogo ativo dentro de um jogo idle é adotado por jogadores *hardcore* que estão sempre ativos, clicando e investindo seus recursos a todo momento para maximizar o progresso.

## 2.10 RPG

RPG é a sigla inglesa de Role-Playing Game, que consiste em um jogo no qual os jogadores desempenham o papel de um personagem em um cenário fictício. No contexto do trabalho, abordaremos o conceito de RPG eletrônico, que não possui alguns dos aspectos contidos no RPG de mesa, como a total liberdade criativa do jogador para mudar história em que está inserido. No RPG eletrônico, é possível controlar personagens e tomar decisões que mudam a trajetória do jogador. De maneira simplificada, pode ser exemplificado pela seguinte sequência: criação de um personagem fraco, realização de missões, receber recompensar, aumentar o nível e fortalecer o personagem.

## 2.11 Dungeon

*Dungeons*, masmorras ou calabouços, no universo de jogos, é um ambiente labiríntico onde os jogadores entram, coletam tesouros, batalham contra monstros e outros inimigos, resgatam pessoas, encontram armadilhas e, no fim, atingem um objetivo final ou encontram uma saída. O conceito provavelmente se originou do jogo de tabuleiro *Dungeons and Dragons* e, desde então, tem sido usado como ponto principal de inúmeros outros jogos de RPG (8).

## 2.12 Bioma

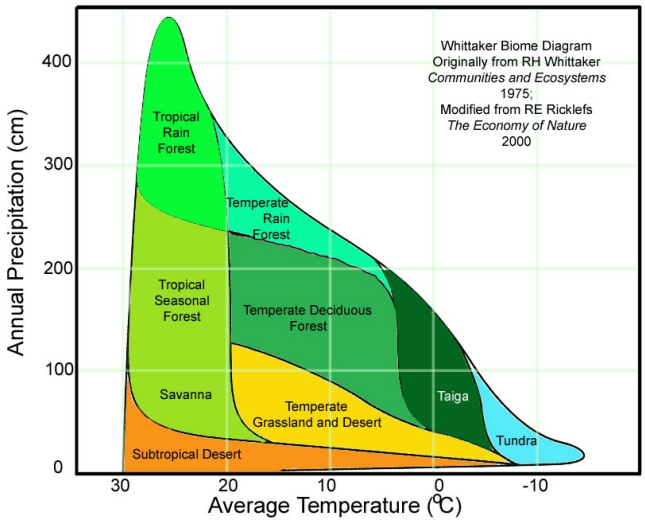
Um bioma é classicamente definido como uma área do espaço geográfico muito extenso que possui um macroclima, uma formação vegetal e uma fauna uniformes e bem definidos

(9). No contexto dos jogos, o termo é abstraído para regiões com um conjunto bem definido de características como vegetação e fauna.

## 2.13 Diagrama de Whittaker

Robert Whittaker foi um dos ecologistas mais influentes dos tempos modernos. Estudou a fundo a estrutura e o funcionamento dos ecossistemas e suas comunidades. Além da sua contribuição para a ecologia, na forma de classificação de reinos e comunidades, realizou um estudo sobre o padrão de vegetação de um bioma em função da temperatura e da precipitação (9), dando origem ao Diagrama de Whittaker.

Figura 4: Diagrama de Whittaker




Fonte: Procedural Content Wiki
(http://pcg.wikidot.com/pcg-algorithm:whittaker-diagram)

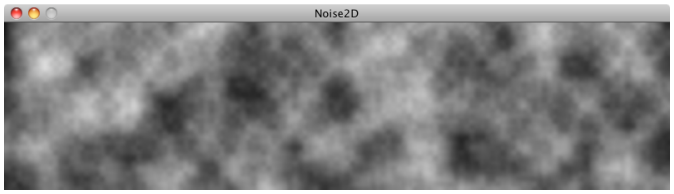
No universo de jogos, os estudos de Whittaker foram aplicados, principalmente, na modelagem de terrenos procedurais, com características próximas ao que se espera da distribuição de biomas no mundo real.

## 2.14 Ruído de Perlin

Ruído de Perlin (*Perlin Noise*) consiste em um gradiente gerado pseudo-aleatoriamente sobre um espaço n-dimensional(10). Apresentado na Siggraph de 1985, Ken Perlin propôs o seu método para sintetizar imagens a partir de ruído - valores pseudo-aleatórios distribuídos de forma suave ao longo do espaço. O método consiste em calcular vetores

gradientes, interpolando-os entre os vizinhos, partindo-se de uma tabela de números pré-determinados que serão utilizados de maneira randômica. Neste projeto, ruídos bidimensionais foram utilizados para gerar mapas de temperatura, altitude e umidade, utilizados conjuntamente com uma adaptação do Diagrama de Whittaker para construir diferentes terrenos com biomas distribuídos de forma crível.

Figura 5: Exemplo de Ruído de Perlin bidimensional




Fonte: Khan Academy - A Two Dimmensional Noise
(https://pt.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/two-dimensional-noise

Na figura, o espaço bidimensional populado com ruído associa cada ponto do plano a um valor entre 0 (preto) e 1 (branco), com transições suaves entre os vizinhos, resultando em uma espécie de nuvem.

# 3 ESTADO DA ARTE

## 3.1 Estado da arte de jogos incrementais

Uma análise de jogos populares e bem sucedidos foi realizada, com o objetivo de compreender os pontos positivos e negativos de cada produto. A análise foi feita considerando-se a relevância do título no contexto dos jogos incrementais, no feedback de usuários e métricas de popularidade, além de um estudo das mecânicas do jogo em si e como estas afetam a retenção e a experiência do jogador.

### 3.1.1 Clicker Heroes

Clicker Heroes, desenvolvido pela Playsaurus, é provavelmente um dos exemplares mais famosos do gênero. Com mais de 32 milhões de acessos na Kongregate (11), plataforma em que surgiu. Hoje conta também com mais de 5 milhões de downloads na Steam (12) e mais de 1 milhão na Play Store (13).

Serviu como base para muitos outros jogos incrementais que misturam-se com RPG. Possui alguns heróis, que recebem upgrades. Ao atingir um ponto no jogo em que não se consegue avançar facilmente, o jogador pode "ascender", voltando ao primeiro nível mas com alguns bônus. Ainda, o jogador pode transcender, resetando inclusive as ascensões, em troca de uma outra moeda para aumentar o progresso exponencialmente.

Figura 6: Interface principal de Clicker Heroes

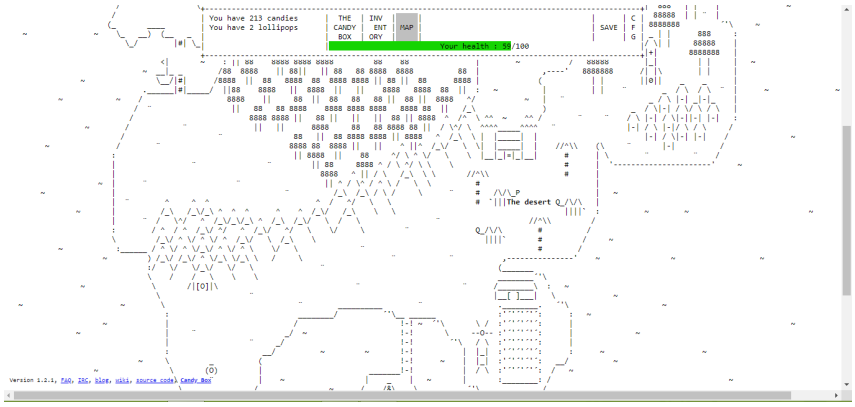

Fonte: Autores

Apesar da temática cativante, o jogo tende a se tornar repetitivo. Não há conteúdo novo, apenas os mesmos monstros cada vez mais fortes, acabando em um simulador de números gigantes. Ao menos isso proporcionou a criação de uma comunidade de *"theory crafters"*, como se denominam os jogadores que analisam a fundo a matemática ótima dos jogos.

### 3.1.2 Candy Box

Candy Box é um jogo de navegador composto inteiramente por arte ASCII pelo desenvolvedor independente "aniwey", que mescla um tema RPG, de escolha de missões, equipamentos e até segredos, com liberdade de exploração e com um sistema incremental, inserindo novas mecânicas e territórios a serem descobertos. Isso possibilita que o jogador ativo tenha uma progressão mais acelerada, enquanto a progressão enquanto ausente é linear. Cada novo lugar que o jogador pode visitar possui um desafio, que irá necessitar de uma quantidade considerável de tempo coletando recursos e de estratégia para ser superado.

Figura 7: Mapa mundi de Candy Box 2

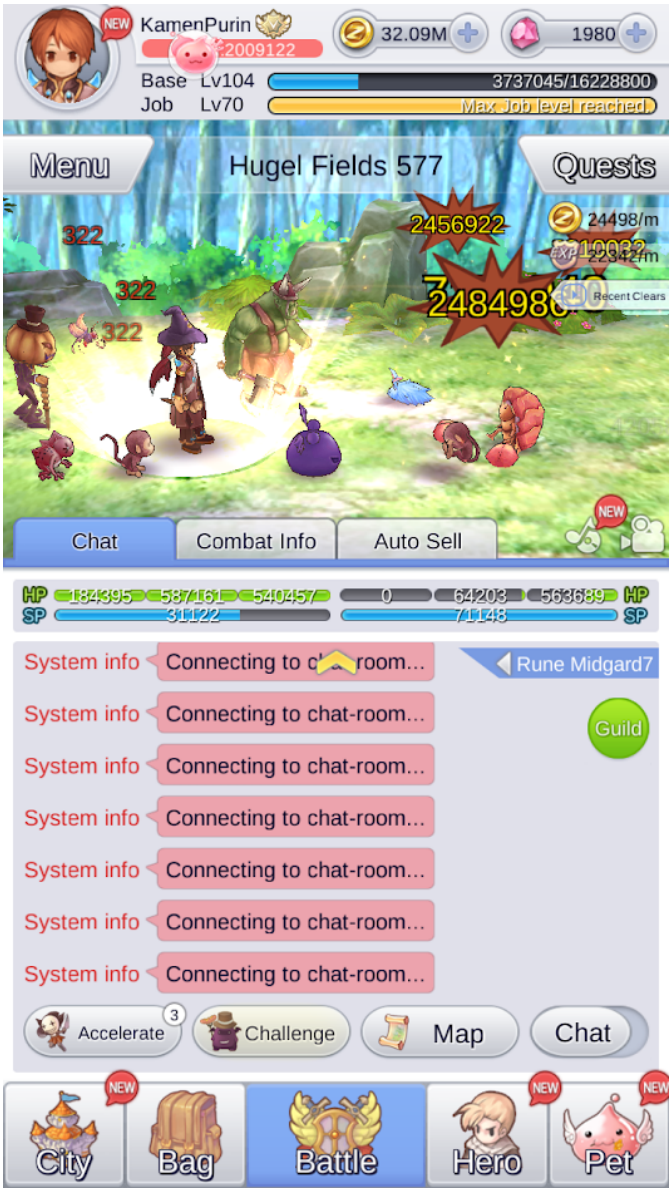



Fonte: Autores

### 3.1.3 Idle Poring

Idle Poring é um jogo incremental baseado em Ragnarök Online, um MMORPG (*Massively Multiplayer Online Role-Playing Game*) de 2002, desenvolvido pela Gravity, popular na época e com usuários ativos até hoje. Licenciado pela própria Gravity, conta com mais de 1 milhão de downloads na Play Store (14). Um aspecto interessante de se notar é o fato do jogo ser baseado em um MMORPG, gênero que, em geral, requer múltiplas horas diárias de *grind* para se obter progresso. Trasnpondo a temática para o gênero incremental, tem-se o *grind* substituído por simplesmente o progresso enquanto inativo, aliviando a pressão de estar sempre jogando para poder competir com outros jogadores.

Figura 8: Interface principal de Idle Poring




Fonte: Autores

### 3.1.4 Crusaders of the Lost Idols

Crusaders of The Lost Idols possui 17.5 milhões de gameplays na Kongregate (15), 50 mil jogadores na Play Store (16) e em torno de 400 mil na Steam (17).

A base do jogo é semelhante à de Clicker Heroes. Porém, o jogo conta com um sistema interessante de "objetivos", os quais o jogador deve iniciar com todos seus heróis no nível 0 e, ao completá-lo, recebe recompensas. Dessa forma, o jogo torna-se extremamente escalável. Ele possui um fim? Não exatamente. Enquanto os desenvolvedores adicionarem objetivos novos, haverá sempre conteúdo novo. A cada um período aproximado de 3

semanas, há eventos, com heróis novos a serem habilitados. Tudo isso contribui para a fidelidade dos seus usuários.

Na parte de upgrades, além das habilidades passivas e ativas desbloqueadas com o nível dos heróis, há ainda um sistema de árvore de habilidades em que o jogador pode alocar seus "ídolos", recurso obtido a cada vez que se reseta (concluindo ou não um objetivo). Além disso, existem equipamentos, obtidos em baús, que podem possibilitar vantagens substanciais para os heróis que os equipam.

Por fim, o jogo conta com um sistema de "formação". Ou seja, o jogador deve considerar as melhores estratégias que caibam dentro das restrições de cada objetivo, levando em conta os heróis que possui e seus equipamentos, além de como suas habilidades afetam cada membro da formação, tornando assim o progresso também estratégico.

Figura 9: Interface principal de Crusaders of The Lost Idols


Fonte: Autores

A ideia de proporcionar diferentes objetivos, cada um com seu nível de dificuldade, forçando o jogador a desenvolver novas estratégias, é um conceito que instiga o usuário a sempre progredir e evoluir. Por outro lado, jogadores mais antigos frequentemente se deparam com a falta de conteúdo novo, e são obrigados a repetir missões sem objetivo enquanto aguardam por atualizações.

### 3.1.5 Realm Grinder

Realm Grinder, desenvolvido pela Divine Games, é um gerenciador de reinos. Conta com 27 milhões de gameplays na Kongregate (18), 500 mil downloads na Steam (19), mais

de 1 milhão na Play Store (20). Recebe updates frequentes.

Figura 10: Interface principal de Realm Grinder


Fonte: Autores

Possibilita diferentes estilos de jogo, em cada "abdicação" (reset). Não possui uma base de construções muito vasta, mas trabalha bem o conceito incremental. A cada valor pré-determinado de construções, habilita-se uma atualização para esse tipo de construção, multiplicando sua produção. Sobre os recursos, existem o ouro, a moeda básica, as moedas de facção, que são utilizadas conforme a escolha do jogador no início de cada abdicação. O jogo cumpre bem o seu papel, embora a falta de mudanças visuais faça com que a maioria dos jogadores não tenha uma real percepção de progresso. Em geral, os números crescem. E é isso, não importa o quanto seu reino seja melhorado, o background sempre parece a mesma coisa (21).

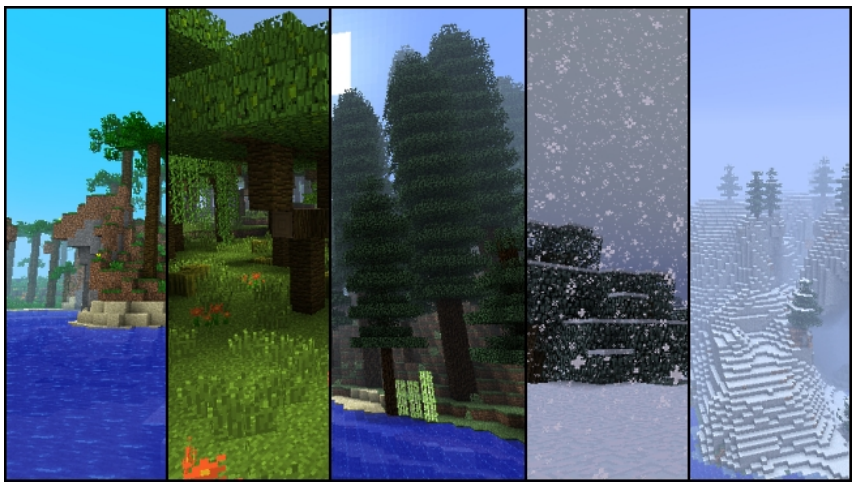
## 3.2 Estado da arte de jogos com geração procedural de conteúdo

Quando se trata de jogos que utilizam de geração procedural de conteúdo, existe uma gama bem variada de gêneros e nomes de sucesso. A análise foi feita levando em conta títulos de sucesso comercial, atividade de usuários atualmente e procurando abranger diferentes tipos de conteúdo gerado.

### 3.2.1 Minecraft

Provavelmente um dos nomes de maior sucesso comercial, com mais de 144 milhões de cópias vendidas, 75 milhões de usuários mensais ativos (22). Minecraft abusa de geração procedural criando um mundo novo utilizando sementes diferentes a cada mundo criado. Dá também a opção de utilizar alguma semente específica para esse propósito.

Figura 11: Biomas de Minecraft



Fonte: Planet Minecraft - https://www.planetminecraft.com/blog/my-input-in-the-future-of-minecraft-biome-update-and-seasons/

Graças à PCG, o jogo é capaz de proporcionar uma experiência nova toda vez que um jogador decide começar um novo mundo. A variedade de recursos diferentes disponíveis também contribui, já que a disponibilidade dos mesmos varia conforme o mundo - pode ser que o jogador comece em uma floresta, com muita madeira a ser coletada, ou no meio da neve, com uma infinidade de cavernas para explorar.

### 3.2.2 Série Diablo

No quesito geração de masmorras rogue-like, Diablo é a série que mais explodiu no mercado, apesar de fugir do núcleo do gênero (movimentação por turnos), voltando-se mais a um RPG de ação. Os jogos da série contam com o apoio da geração de itens e mapas das instâncias de cada masmorra, além da distribuição dos inimigos. É exatamente este o ponto que mantém os jogos da série vivos até hoje. Diablo 2, do ano de 2000, ainda possui uma comunidade ativa, enquanto Diablo 3, de 2013, consegue atrair uma parte dos jogadores de volta mensalmente.

Figura 12: Mapa gerado em uma Nephalem Rift em Diablo 3


Fonte: Autores

No caso de Diablo 3, o jogo prende seus jogadores com atualizações mensais ("temporadas"), e, dentro de cada temporada, com a progressão à medida em que se obtém itens melhores para avançar no nível de dificuldade das masmorras geradas proceduralmente.

### 3.2.3 Série Mystery Dungeon

A série Mystery Dungeon, que abrange diversos títulos de diferentes empresas (como Pokémon, Dragon Quest, Final Fantasy...), é a série rogue-like de maior sucesso no Japão. Os jogos contam com histórias mais bem preparadas (o que é muito valorizado no mercado de jogos japonês), com a jogabilidade no estilo de dungeon crawler/rogue-like. Além da história, frequentemente apresentam níveis mais difíceis e menos casuais, demandando maior dedicação dos jogadores.

### 3.2.4 Série Borderlands

Borderlands é um jogo de tiro em primeira pessoa com elementos de RPG desenvolvido pela Gearbox. O núcleo principal do jogo gira em torno de uma história principal e missões secundárias. Porém, o grande atrativo de Borderlands e que retém os jogadores que voltam a criar novos personagens ou concluir todas as missões do jogo, são os itens gerados proceduralmente. Cada arma, escudo e acessórios são construídos a partir de uma combinação semi-aleatória de cada parte do item (no caso de armas, por exemplo, temos uma lista de possíveis Corpo, Empunhadura, Pente, Cano, Mira, Coronha e Acessório) e de um fabricante.

Figura 13: Partes possíveis de submetralhadoras em Borderlands 2


Fonte: Borderlands Wikia - http://borderlands.wikia.com/wiki/Gun_Component_Charts

Cada componente influencia de uma certa forma o resultado dos atributos da arma. Dessa forma, a jogabilidade é altamente influenciada pelas armas geradas que cada jogador (ou cada personagem de cada jogador) obtém.

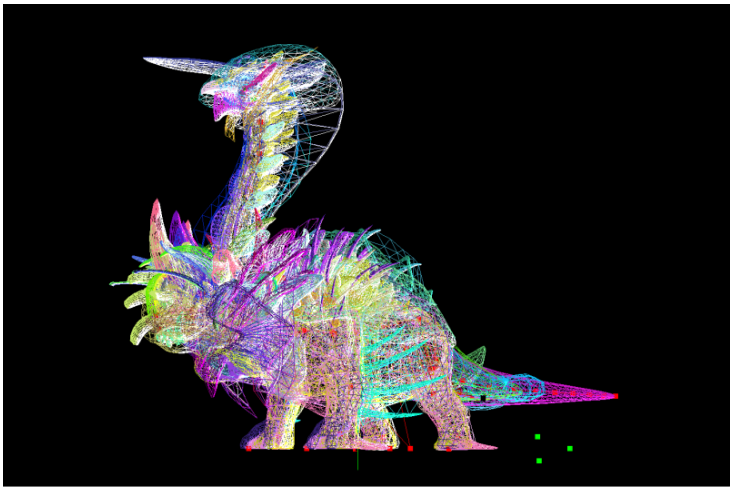
Figura 14: Exemplo de submetralhadora gerada em Borderlands 2


Fonte: Autores

### 3.2.5 No Man's Sky

No Man's Sky é um jogo de ação-aventura de sobrevivência desenvolvido pelo estúdio indie Hello Games, lançado em agosto de 2016 para Playstation 4 e Windows e para Xbox One previsto em meados de 2018. Teve uma recepção mista de usuários e críticos, mas o sistema de geração de conteúdo merece uma atenção especial. Monstros, naves, planetas, ecossistemas, galáxias foram todos gerados procedural e deterministicamente nesse jogo de universo aberto.

Figura 15: Estrutura base de criaturas triceratopes em No Man's Sky



Fonte: 3D Game Dev Blog - No Man's Sky Procedural Content - http://3dgamedevblog.com/wordpress/?p=836

Figura 16: Exemplos de criaturas geradas da estrutura anterior

Fonte: 3D Game Dev Blog - No Man's Sky Procedural Content - http://3dgamedevblog.com/wordpress/?p=836

Estima-se que existam mais de 18 quintilhões ($18 * 10^{18}$) planetas diferentes. Por outro lado, jogadores afirmam que após atravessar 256 galáxias, os planetas começam a se repetir e frequentemente criticam o gameplay repetitivo e enfadonho, alegando que o foco da equipe foi a geração de um universo procedural, deixando de lado a parte principal de um jogo - sua jogabilidade. No segundo semestre de 2018, os desenvolvedores responsáveis acrescentaram atualizações, refinando a jogabilidade, objetivos e história, além de possibilitar que o fosse jogado em modo cooperativo com equipes de até 4 jogadores, melhorando drasticamente a recepção do título tanto por jogadores quanto por críticos.

# 4 TECNOLOGIAS UTILIZADAS

## 4.1 Unreal Engine 4

Unreal Engine 4, desenvolvida pela Epic Games, é uma das *game engines* mais relevantes atualmente, ao lado da concorrente Unity. De 2016 em diante, a Unreal ganhou ainda mais relevância no mercado, com algumas gigantes do mercado de jogos como Square Enix (23), Bandai Namco (24) e Capcom (25) substituindo suas *engines* proprietárias por Unreal.

A ferramenta possui código aberto. Assim, toda nova atualização possui novidades adicionadas pela própria Epic Games, assim como alterações, melhorias e até *features* novas desenvolvidas por usuários comuns. Com isso, tem-se uma ferramenta poderosa e atualizada a todo momento, capaz de trabalhar com áudio, gravação de vídeos, construção de cenários, animações, diferentes dispositivos de entrada, iluminação estática e dinâmica de qualidade, scripts visuais, analisadores de performance, entre muitas outras capacidades da *engine*.

Foi escolhida por ser uma ferramenta robusta, poderosa e versátil, além de possibilitar prototipagem rápida graças à linguagem visual Blueprint. Além disso, sua licença é completamente gratuita, sendo necessário apenas repassar royalties com base no faturamento dos produtos, sendo então uma ótima escolha tanto para desenvolvedores pequenos como para o ambiente acadêmico.

### 4.1.1 Adobe Fuse CC Beta

Fuse é uma ferramenta da Adobe para criação de personagens humanoides. Enquanto em sua fase beta, está disponível gratuitamente e sem *royalties* ou quaisquer outras taxas para uso comercial ou não-comercial (26). Trata-se de uma ferramenta bastante versátil para obtenção de modelos 3D e texturas para jogos e experiências, principalmente para casos de uso em que as personagens sejam humanos comuns.

Conta com um editor complexo de feições corporais e faciais, além de vestimentas. Os modelos e texturas podem ser exportados para diversas extensões de arquivos. Neste projeto, foi utilizada a extensão FBX para melhor compatibilidade com a Unreal Engine 4.

### 4.1.2 Mixamo

Assim como o Fuse, o Mixamo é uma ferramenta da Adobe totalmente gratuita, disponível para uso no próprio site. É o primeiro serviço online de animação de personagens (27), constrói automaticamente esqueletos de modelos humanoides, disponibiliza cerca de 30 personagens de uso livre, além de possuir um vasto banco de animações a serem aplicadas a qualquer modelo que o usuário importe e passe pelo processo de construção de esqueleto ou aos próprios modelos disponíveis.

Todas as animações de humanoides do projeto foram extraídas desta ferramenta.

### 4.1.3 Versionamento de Código em Git

Git é a ferramente de controle de versão mais utilizada no desenvolvimento de software (28). Durante o desenvolvimento do projeto, todas as alterações foram armazenadas em um repositório hospedado no Git Lab, pois este apresenta maiores facilidades para armazenamento de arquivos grandes - como é o caso de modelos 3D e texturas de alta resolução.

# 5 METODOLOGIA DO TRABALHO

O projeto será dividido em etapas:

1. **Levantamento bibliográfico:** a base conceitual do projeto será estudada a partir de literaturas, em sua grande maioria livros de geração procedural e design de jogos.

2. **Busca de referências:** com o objetivo de compreender os elementos que proporcionaram o sucesso de alguns jogos no mercado, estes serão analisados e estudados.

3. **Elaboração de um documento de Game Design (GDD):** o documento é importante para definir todos os elementos do jogo, como tema, ambientação, condições de vitória e derrota, personagens, recursos, recompensas, entre outros.

4. **Desenvolvimento um protótipo do jogo:** obter uma versão viável do jogo, para iniciar testes com usuários e obter feedback para potenciais modificações de projeto. Será utilizado o motor de jogos Unreal Engine 4, que utiliza C++ como linguagem principal e Blueprint, uma linguagem visual própria da Unreal.

5. **Utilização de geração procedural:** aplicar PCG (Procedural Content Generation) no jogo, de modo que os mapas, níveis e desafios sejam gerados computacionalmente.

6. **Desenvolvimento de subsistemas:** desenvolver as features necessárias para contemplar os elementos estabelecidos no GDD.

   Para orientar o desenvolvimento e garantir o andamento do projeto, o cronograma a seguir foi desenvolvido:

Tabela 1: Cronograma e etapas do projeto

| Atividades | Meses | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Levantamento Bibliográfico | | | | | | | | | | | | |
| Busca de referências | | | | | | | | | | | | |
| Elaboração do GDD | | | | | | | | | | | | |
| Desenvolvimento do protótipo | | | | | | | | | | | | |
| Elaboração do documento de projeto | | | | | | | | | | | | |
| Revisão Bibliográfica | | | | | | | | | | | | |
| Aplicação de PCG | | | | | | | | | | | | |
| Desenvolvimento de subsistemas | | | | | | | | | | | | |
| Depuração e revisão | | | | | | | | | | | | |

# 6 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

A especificação de requisitos pode ser reduzida da descrição de Gameplay do Documento de Game Design (Apêndice A). Um trecho do capítulo que descreve o *gameplay* se encontra a seguir:

"O jogo conta com dois núcleos principais: o gerenciamento dos territórios e as batalhas nas cavernas. Na primeira parte, o jogador irá alocar recursos em uma vila. Ele irá recrutar camponeses, lenhadores, mineradores, metalúrgicos, entre outras funções para produzir recursos. Os camponeses gerarão comida, que permitirá manter mais pessoas na vila. Madeira, pedras e metais servirão para construir edifícios e equipamentos para tropas de combate (que serão utilizadas no segundo núcleo).

Na segunda parte, o jogador irá enviar um número pré-determinado de tropas de combates para cavernas ou outros tipos de dungeons. Ao concluí-las, o jogador irá ter a possibilidade de manter tropas no lugar, coletando recursos (itens, experiência e dinheiro, todos para evoluir a vila e as tropas), além de liberar novos terrenos a serem adquiridos e até descobrir novas profissões para suas tropas.

Dessa forma, o jogador irá acumular cada vez mais recursos, e os investirá em tropas para liberar novos terrenos mais produtivos e novas dungeons, que gerarão crescentemente mais recursos. As tropas podem equipar armas e armaduras, além de aprenderem habilidades que podem auxiliar no combate. Os combates acontecem de maneira automática. As tropas se movem até os inimigos e lutam conforme uma estratégia pré-estabelecida. Assim, a vitória se dará com uma boa estratégia, um bom setup de tropas e com base na força de cada componente.

Ao adquirir uma certa quantidade de territórios, o jogador pode escolher se tornar um "discípulo do apocalipse", recomeçando em um mundo diferente, porém com atributos extra (afinal, ele acabou de destruir e criar outro mundo, algo ele aprendeu). Podemos trabalhar com a ideia de recurso de reset para distribuir em alguma árvore de habilidades

e/ou adicionar mecânicas novas a cada reset, sendo que os objetivos de reset mudam."

## 6.1 Histórias de usuários

No contexto de desenvolvimento de software, as histórias de usuário são pequenas descrições que contém três informações essenciais para o desenvolvimento de uma funcionalidade: quem, o quê e por quê. Estas informações estão relacionadas respectivamente ao usuário que tem alguma necessidade; qual a sua necessidade; e o motivo pelo qual quer que esta necessidade seja possível. A partir da sentença formada por estas três informações, o desenvolvedor imagina-se na situação do usuário em questão e implementa a funcionalidade pensando nas necessidades do usuário em questão.

- Jogador deve ser capaz de receber recursos de acordo com suas construções para poder investí-los em outras áreas do jogo
- Jogador deve ser capaz de recrutar trabalhadores para gerar recursos para sua vila
- Jogador deve ser capaz de recrutar tropas de combate para a utilização no núcleo de combate
- Jogador deve ser capaz de selecionar membros de tropas de combate para enviar às *dungeons*
- Jogador deve poder utilizar os recursos obtidos em cada reset para customizar sua árvore de habilidades e talentos, criando seu próprio estilo de jogo
- Jogador deve ser capaz de alocar tropas para coletar recursos após concluir uma *dungeon*
- Jogador deve ser capaz de resetar o seu progresso, retornando ao nível 1 para ganhar pontos de maestria e gerar um novo mundo com novos objetivos
- Jogador deve ser capaz de popular sua vila com edifícios para que estes auxiliem em outros aspectos do jogo
- Jogador deve ser capaz de aumentar os níveis e evoluir as construções de sua vila para aumentar a influência destas em seu jogo
- Jogador deve ser capaz de optar pelo combate automático em *dungeon* para economizar seu tempo

## 6.2 Requisitos Não-funcionais

Os requisitos não-funcionais dizem respeito principalmente à geração de conteúdo e acesso ao servidor.

- Mobile-friendly: o jogo deve ser capaz de rodar em dispositivos móveis medianos, além de ser compatível com redes móveis.
- Capacidade e segurança do servidor: o servidor deve ser compatível com o volume de dados armazenado para cada jogador. Além disso, não deve tratar entradas suspeitas dos clientes, evitando trapaças.
- Persistência de dados: o jogo deve restaurar o seu estado anterior, a partir do ponto em que o jogador deixou de jogar pela última vez.

## 6.3 Lista e classificação de features

A lista de histórias de usuário foi expandida em features que cumpram cada requisito. Cada feature foi então classificada de acordo com o sistema MoSCoW (do inglês *"Must, Should, Could, Won't Have"*, ou, em português, "Deve ter, Deveria ter, Poderia ter e Não terá"). A tabela com essa classificação encontra-se a seguir.

Tabela 2: Tabela de features da extensão do sistema

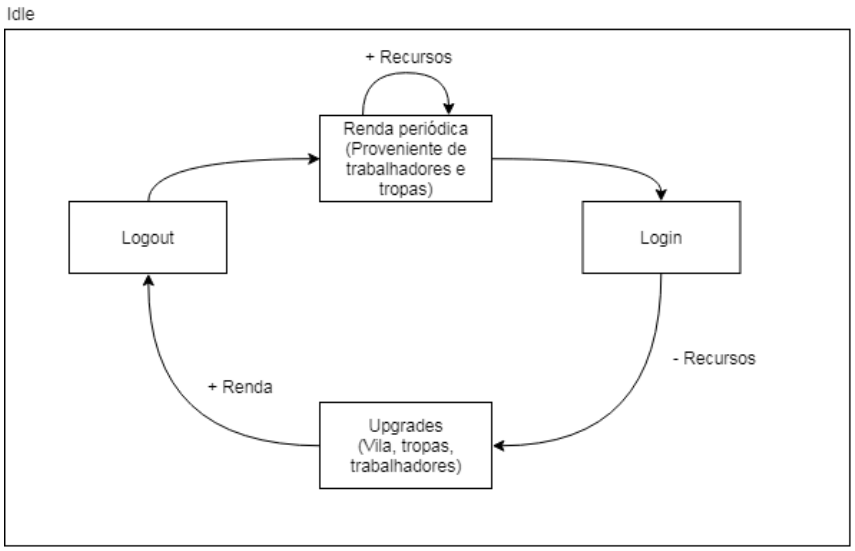
| Feature | Núcleo | Prioridade |
|---|---|---|
| Receber recursos | Gerenciador | Deve ter |
| Investir recursos | Gerenciador | Deve ter |
| Recrutar trabalhadores | Gerenciador | Deve ter |
| Recrutar tropas | Gerenciador | Deve ter |
| Alocar pontos do jogador | Geral | Deve ter |
| Formar um grupo de combate | Combate | Deve ter |
| Tropas coletando recursos | Gerenciador | Deve ter |
| Resetar o mundo | Geral | Deve ter |
| Construir edifícios | Gerenciador | Deveria ter |
| Evoluir construções | Gerenciador | Deveria ter |
| Evoluir tropas | Gerenciador | Deveria ter |

| | | |
|---|---|---|
| Combate automático | Combate | Deveria ter |
| Construir equipamentos | Gerenciador | Poderia ter |
| Gerenciar equipamentos | Combate | Poderia ter |
| Aprender habilidades | Combate | Poderia ter |
| Definir estratégia de combate | Combate | Poderia ter |
| Possuir objetivo | Geral | Poderia ter |
| Alterar objetivos | Geral | Poderia ter |
| Adicionar novas mecânicas | Geral | Poderia ter |
| Combate PvP | Geral | Poderia ter |

## 6.4 Esquematização do gameplay

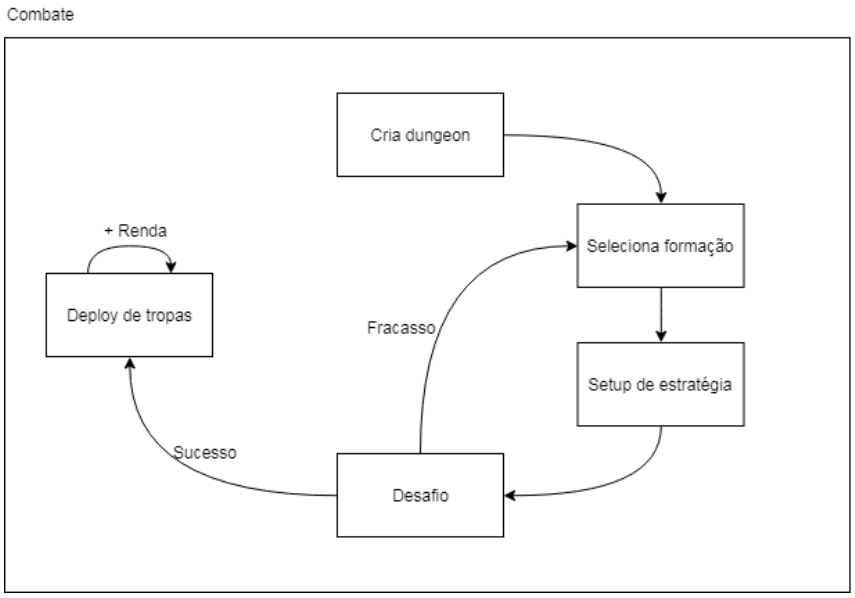
Para melhor entendimento das ações que o jogador pode realizar durante o gameplay, foram desenvolvidos diagramas representando os loops presentes na partes de combate e idle do jogo.

Figura 17: Diagrama Idle



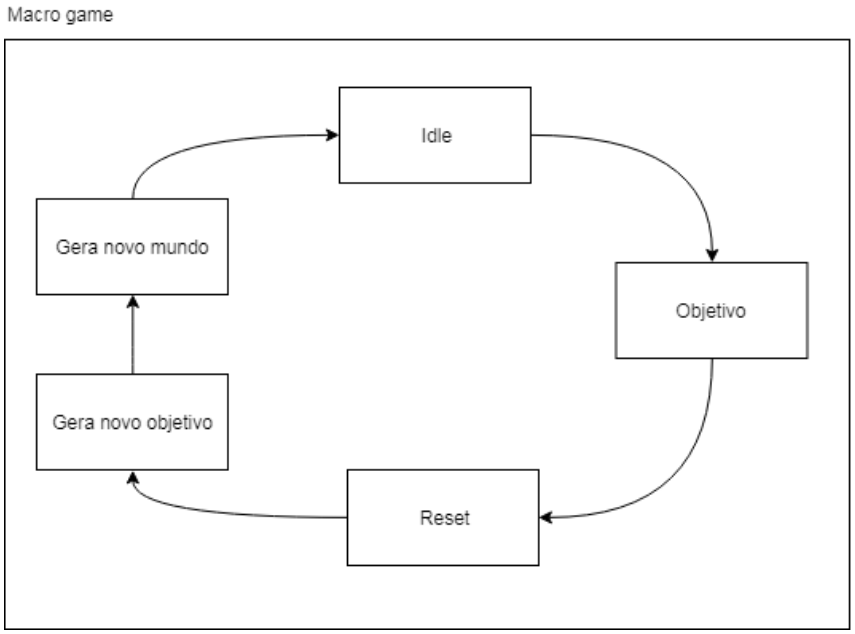

Fonte: Autores

Figura 18: Diagrama Combate




Fonte: Autores

Figura 19: Diagrama Macro




Fonte: Autores

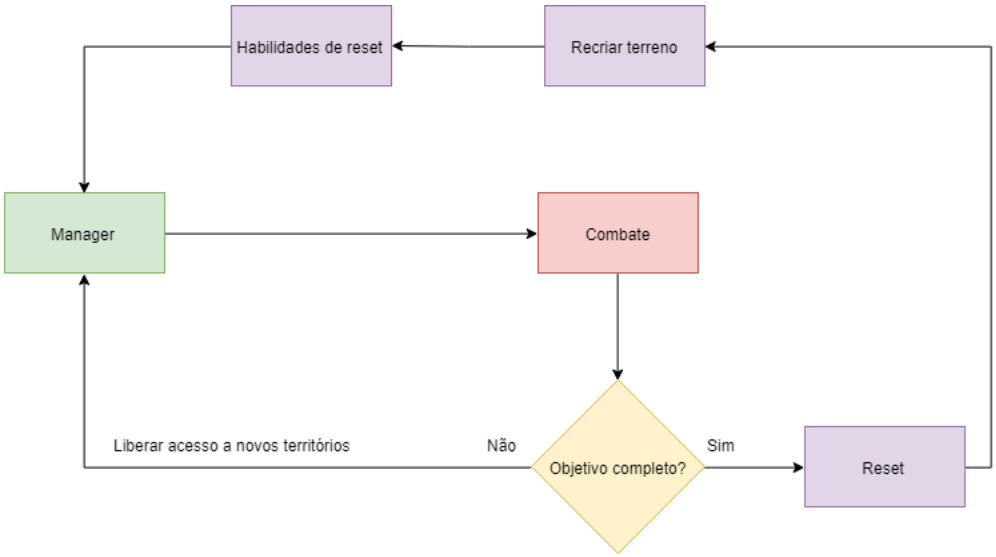
# PARTE II

## IMPLEMENTAÇÃO DO PROJETO

# 7 PROJETO E IMPLEMENTAÇÃO

Uma visão superficial da arquitetura do projeto pode ser visualizada no diagrama abaixo. De maneira simplificada, o núcleo principal do jogo se encontra no Manager e no Combate, que são dependentes entre si - durante a fase de Manager, o jogador irá recrutar e treinar tropas para que sejam utilizadas na fase de Combate, enquanto combates bem sucedidos aumentarão o território a ser utilizado pelo jogador na fase do Manager. Cada etapa, bem como a avaliação de objetivos e o reinício de jogo via reset serão detalhados nas próximas seções.

Figura 20: Visão geral do projeto




Fonte: Autores

## 7.1 Manager

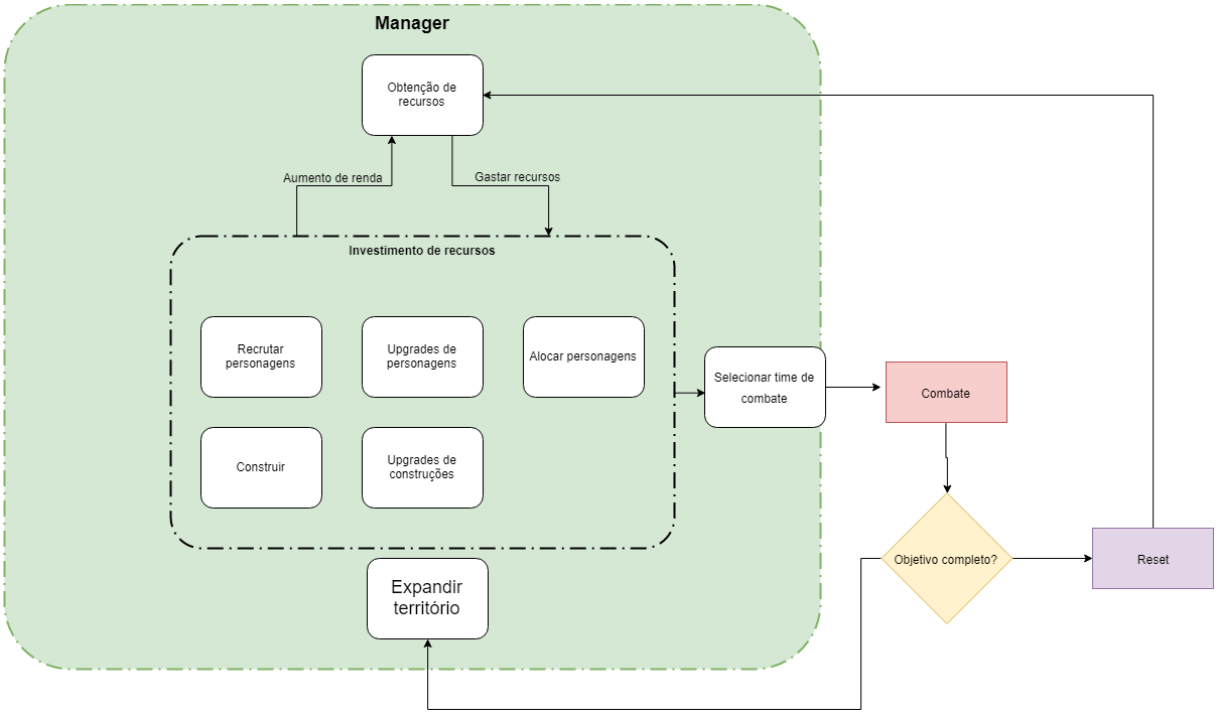
O manager consiste na parte do jogo em que se constrói edifícios, com o objetivo de iniciar e melhorar a geração de recursos do jogador (como madeira, ouro, rochas, ferro...), e se recruta personagens, que irão trabalhar nos edifícios construídos ou serão treinados

para combate.

### 7.1.1 Visão Geral

Nesta etapa do jogo, o mundo é construído sobre um sistema de *grid*. Cada *tile* corresponde a um tipo de bioma diferente, que impõe restrições sobre o que pode ser coletado ou construído. Por exemplo, não se deve construir uma madeireira sobre a água ou nas montanhas rochosas. Parte do terreno gerado não pode ser acessado inicialmente. Para liberar o acesso, é necessário que o jogador expanda seu território por meio de novas construções e completando as *dungeons*, que se comportam como uma construção geradora de recursos após sua conclusão.

Figura 21: Componentes do Manager



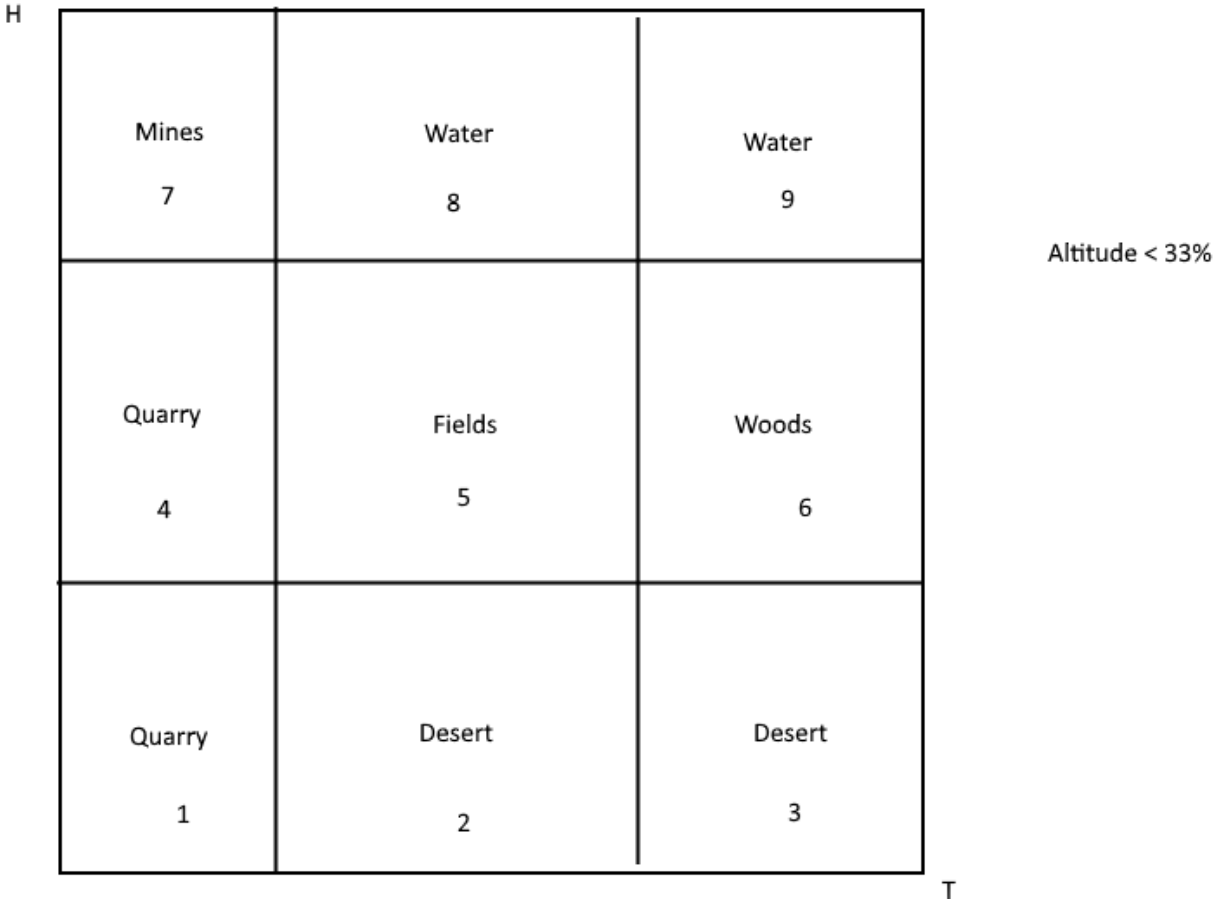

Fonte: Autores

Para poder vencer os desafios das masmorras, o jogador irá utilizar os seus recursos para obter uma equipe de personagens de combate que irão acompanhá-lo nas batalhas. Os personagens, tanto de combate quanto produtores de recursos, podem ser recrutados na Taverna. Cada personagem recrutável irá compôr a população da sua vila, que pode ser expandida através da construção de alojamentos. Além disso, todos podem ser treinados diretamente na vila por um custo, de acordo com a sua função. Portanto, para progredir, o jogador deverá manejar com sabedoria todos os recursos disponíveis no mundo em que se encontra.

### 7.1.2 Geração de Terrenos

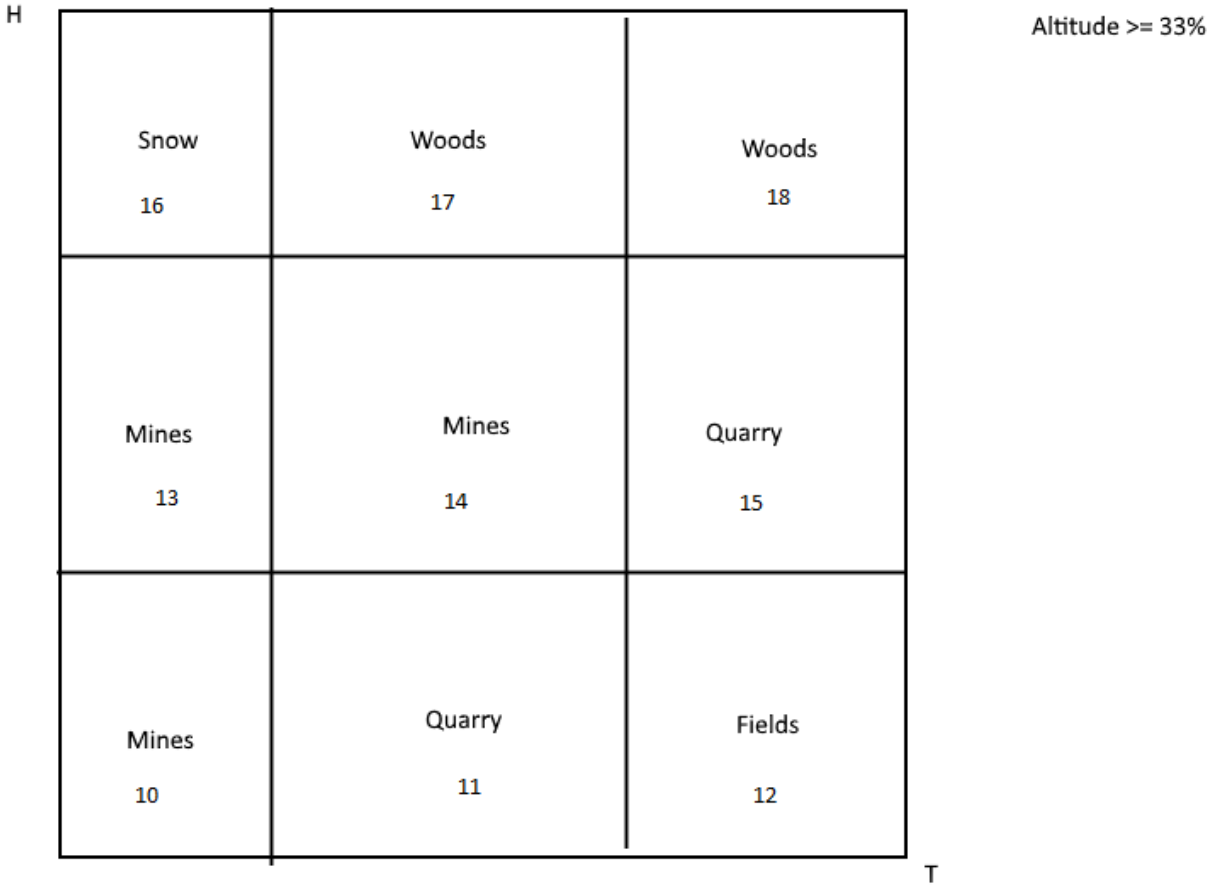
O mundo será diferente em cada *playthrough*. O terreno é gerado proceduralmente, de maneira que as proporções de biomas sejam diferentes a cada vez. Para a criação do terreno, um método baseado na geração de mapas de altitude, temperatura e umidade com ruídos de Perlin foi utilizado. Cada mapa de ruído foi gerado utilizando a biblioteca *Simplex Noise* para Unreal Engine 4. As informações geradas foram então associadas a um bioma, com uma abordagem semelhante a um diagrama de Whittaker. Por simplicidade e para tratar casos não contemplados pelo diagrama de Whittaker original, a seguinte adaptação foi realizada:

Figura 22: Diagrama de Whittaker adaptado - Altitudes baixas




Fonte: Autores

Figura 23: Diagrama de Whittaker adaptado - Altitudes altas




Fonte: Autores

As possibilidades de biomas foram divididas em nove quadrantes de umidade em função da temperatura. Então, essa lógica foi expandida para altas e baixas altitudes, resultando nos dois diagramas customizados. Como o balanceamento do jogo está criticamente ligado à distribuição de biomas ao longo do território, o valor exato que limita cada quadrante foi parametrizado para que ajustes possam ser feitos após testes de usuários.

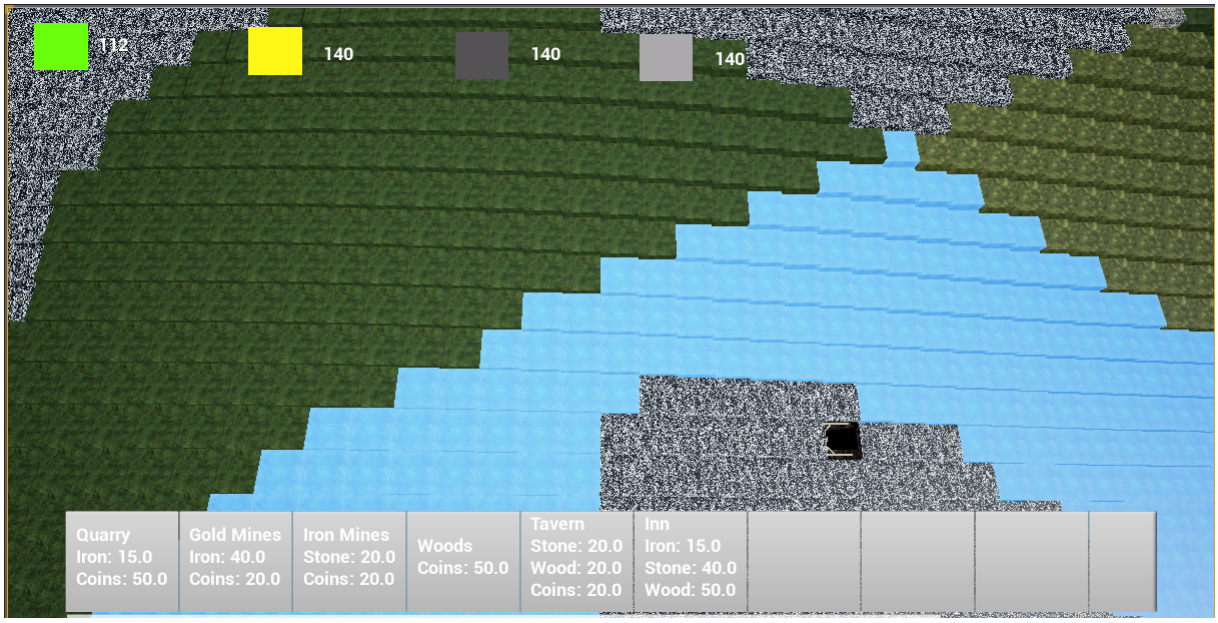
### 7.1.3 Distribuição de Biomas

Como espera-se que o jogador tenha um mínimo de recursos acessíveis no início do jogo, o local escolhido para início levará em conta regiões que não estejam nos limites do mapa e que contenham tais recursos essenciais. Caso não seja possível encontrar um quadrante que satisfaça tais condições, o terreno será gerado novamente. Ainda, o ponto inicial é selecionado buscando por entradas de masmorras próximas à região central do mapa, de forma que o jogador tenha maior liberdade para explorar em qualquer direção.

### 7.1.4 Implementação do Terreno com Biomas

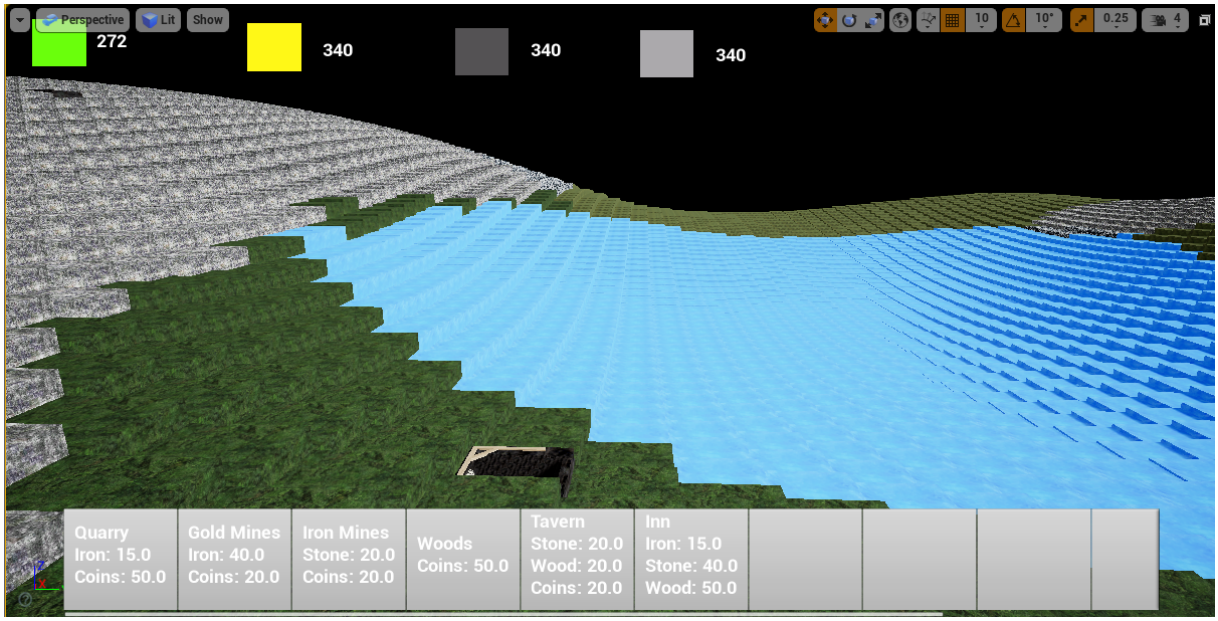
Inicialmente, o terreno foi construído gerando-se uma espécie de tabuleiro, formados por quadrados. Cada quadrado, correspondente a uma coordenada discreta no plano, foi associado a uma medida de temperatura, umidade e altitude para determinar o seu bioma local, bem como a sua respectiva altura. Desse modo, foi possível criar um mapa com relevo e biomas variados, que podem ser observados nas figuras 24 e 25.

Figura 24: Visualização do Manager com terreno gerado




Fonte: Autores

Figura 25: Terreno gerado - visualização de relevo




Fonte: Autores

### 7.1.5 Construções

As construções do jogo consistem em objetos capazes de serem movimentados através do *grid*. O jogador utiliza seus recursos para construí-las e cada construção possui um benefício diferente para o jogador. Ao obter múltiplas construções do mesmo tipo, seus efeitos se amplificam. Ainda, como o jogador só deve poder construir em regiões próximas a construções já existentes. Desse modo, as construções são, além de fornecedoras de benefícios, o modo de exploração do mapa - para acessar as cavernas espalhadas, o jogador deve antes construir próximo às entradas.

As opções disponíveis para construção são:

- Taverna: Possibilita ao jogador atrair trabalhadores para a região, que podem ser recrutados. Cada nova taverna aumenta a qualidade dos trabalhadores recrutáveis.
- Pousada: Aumenta a capacidade populacional da vila. Cada nova pousada aumenta o número total de trabalhadores que o jogador pode abrigar em sua vila.
- Madeireira: Apenas pode ser construída em terrenos de floresta. Produz grandes quantidades de madeira e cada nova madeireira multiplica a produção total de madeira.
- Pedreira: Apenas pode ser construída em terrenos rochosos. Produz grandes quantidades de pedras e cada nova pedreira multiplica a produção total de pedras.
- Minas: Apenas pode ser construída em terrenos ricos em minérios. Produz grandes quantidades de ferro e ouro. Cada nova mina multiplica a produção total de ouro e ferro.
- Fazenda: Apenas pode ser construída em campos. Produz grandes quantidades de alimento e cada nova fazenda multiplica a produção total de alimento.

### 7.1.6 Personagens

Os personagens são trabalhadores que podem ser alocados em cada construção para amplificar a produção ou nas cavernas para participar dos combates. Podem ser recrutados na taverna por um preço e têm sua quantidade limitada proporcional à quantidade de pousadas existentes na vila e à produção de alimento.

- Camponês: Quando alocado em uma fazenda, aumenta sua produção.

- Lenhador: Quando alocado em uma madeireira, aumenta sua produção.
- Minerador: Quando alocado em uma mina ou uma pedreira, aumenta sua produção.
- Espadachim: Personagem de combate corpo a corpo, com alta resistência e baixo dano.
- Arqueiro: Personagem de combate à distância, com baixa resistência e alto dano.
- Mago: Personagem de combate à distância, com baixa resistência e dano médio em área.
- Curandeiro: Personagem de combate cuja função é curar os ferimentos de seus aliados.

### 7.1.7 Interação do jogador

A interação do jogador nesse núcleo do jogo se dá por meio de toques ou cliques nos objetos, arrastando-os pelo *grid*. Toques simples em um objeto devem selecioná-lo, ativando seus menus de interação e permitindo sua movimentação através de gestos de deslize. A movimentação de construções e personagens pelo mapa deve respeitar os limites traçados pelas construções já sob domínio do jogador. Toques de deslize sobre a tela, sem que um objeto esteja selecionado, devem movimentar a câmera até os limites do mapa, enquanto gestos de pinça devem permitir que o jogador aproxime ou afaste a câmera, conforme a orientação do gesto. Interações que não dizem respeito a movimentação de objetos se dão por meio de interfaces tradicionais, compostas por janelas com textos e botões.

#### 7.1.7.1 Loja

A loja de construções pode ser acessada interagindo com o botão identificado pelo ícone da figura 26, que expande o menu. Nele, existem botões contidos em uma barra rolante horizontal, de tal maneira que cada botão está associado à criação de cada tipo de construção, identificando seu tipo e custo, conforme figura 27.

Figura 26: Ícone para abertura do menu de construções

Fonte: Autores

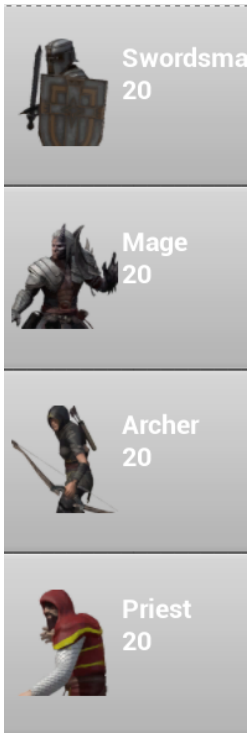
Figura 27: Menu de construções


Fonte: Autores

#### 7.1.7.2 Menu de recrutamento

O menu de recrutamento pode ser acessado ao interagir com uma taverna. Nele existem botões contidos em uma barra rolante vertical, de tal maneira que cada botão está associado a cada tipo de personagem, identificando sua classe, custo de manutenção e preço.

Figura 28: Menu de recrutamento



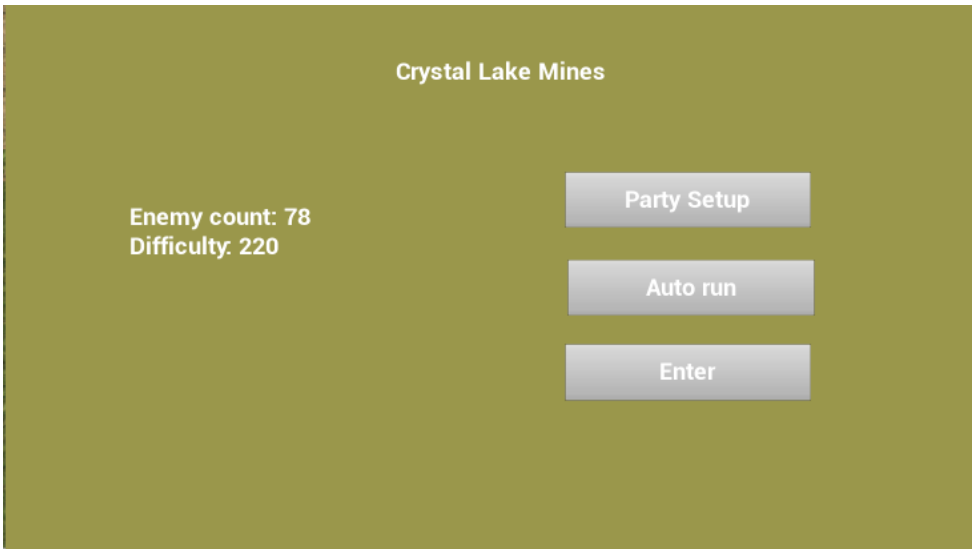

Fonte: Autores

#### 7.1.7.3 Menu de Entrada de Caverna

Ao interagir com a entrada de uma caverna, o menu da figura 29 é apresentado, no qual é possível visualizar três botões. O primeiro permite que o jogador selecione os seus aliados que irão auxiliar na primeira exploração da masmorra ou na auto-exploração após concluí-la pela primeira vez; o segundo só é ativado caso a masmorra já tenha sido concluída pelo menos uma vez, e aciona a auto-exploração; por fim, o terceiro inicia a

exploração, enviando o jogador para a etapa de combate juntamente com seus aliados.

Figura 29: Menu de recrutamento

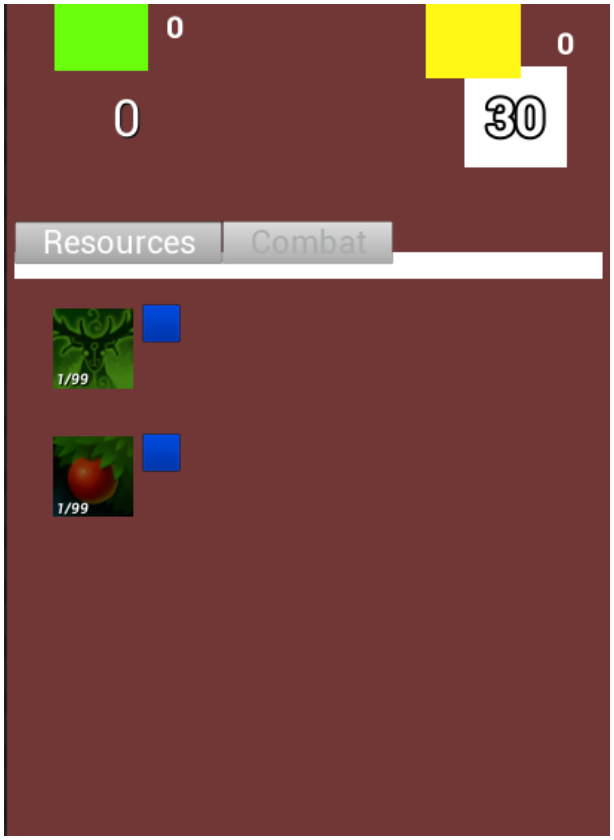



Fonte: Autores

#### 7.1.7.4 Menu de Maestrias

O menu de maestrias só é habilitado após cumprir o primeiro objetivo, ou seja, após o primeiro *reset*. Nele o jogador pode distribuir os pontos obtidos após finalizar cada objetivo em duas diferentes árvores de maestrias, adquirindo novas habilidades que podem aumentar a produção de recursos ou melhorando a performance de suas tropas nos combates.

Figura 30: Árvore de Maestrias




Fonte: Autores

#### 7.1.7.5 Menu de Relação entre Tropas e Construções

A relação entre tropas e construções apenas tem a função de visualizar quais personagens estão trabalhando em quais construções ou masmorras, permitindo que o jogador remova-os caso deseje. Mostra também informações numéricas como o aumento proporcional de produção de cada personagem em seus respectivos trabalhos.

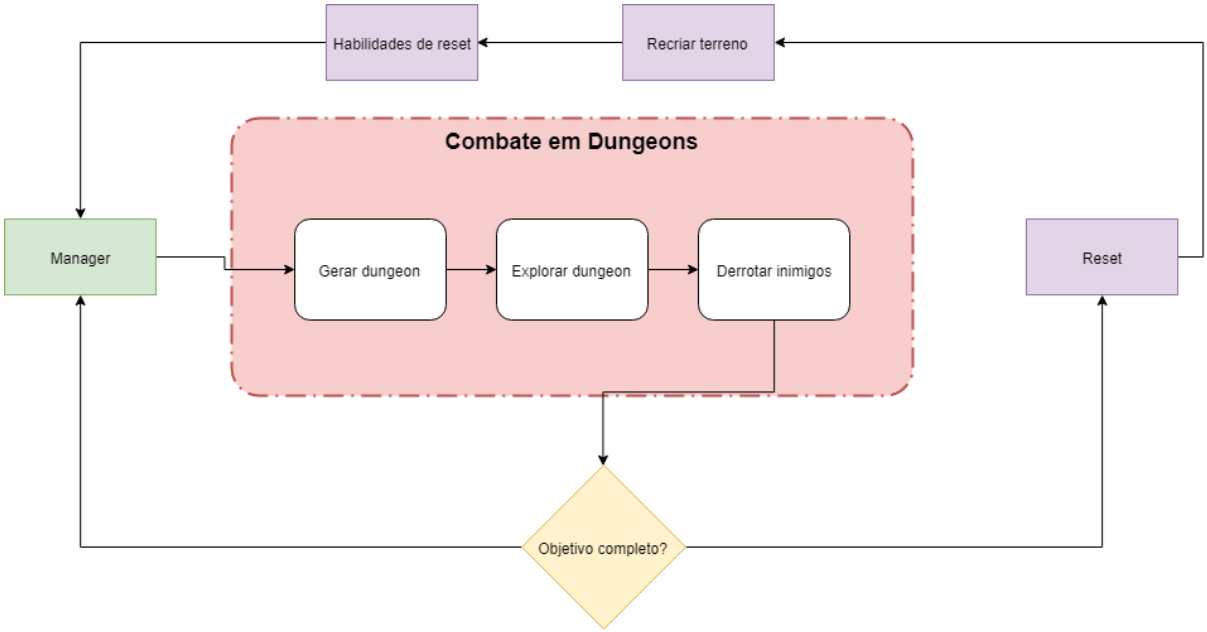
## 7.2 Geração de Dungeons

As *dungeons*, ou masmorras, são locais que podem ser explorados pelo jogador e, uma vez conquistadas, auxiliam na expansão do território e na geração de recursos. Para explorar uma masmorra, é necessário alocar pelo menos um personagem de combate, que terá como função derrotar todos os inimigos presentes, deixando-na um lugar seguro para coleta de recursos.

### 7.2.1 Visão Geral

Nesta etapa do jogo, as masmorras são construídas sobre um sistema de *grids* quadradas de salas. O tamanho da grid pode variar de 1x1 até 10x10, tendo relação com o nível de experiência do jogador. Em cada sala, o jogador pode encontrar inimigos e itens coletáveis. Sua missão é derrotar os inimigos através das tropas que enviou para combate e pegar os coletáveis, que representam recursos adicionais ao fim deste modo de jogo.

Figura 31: Componentes da fase de combate



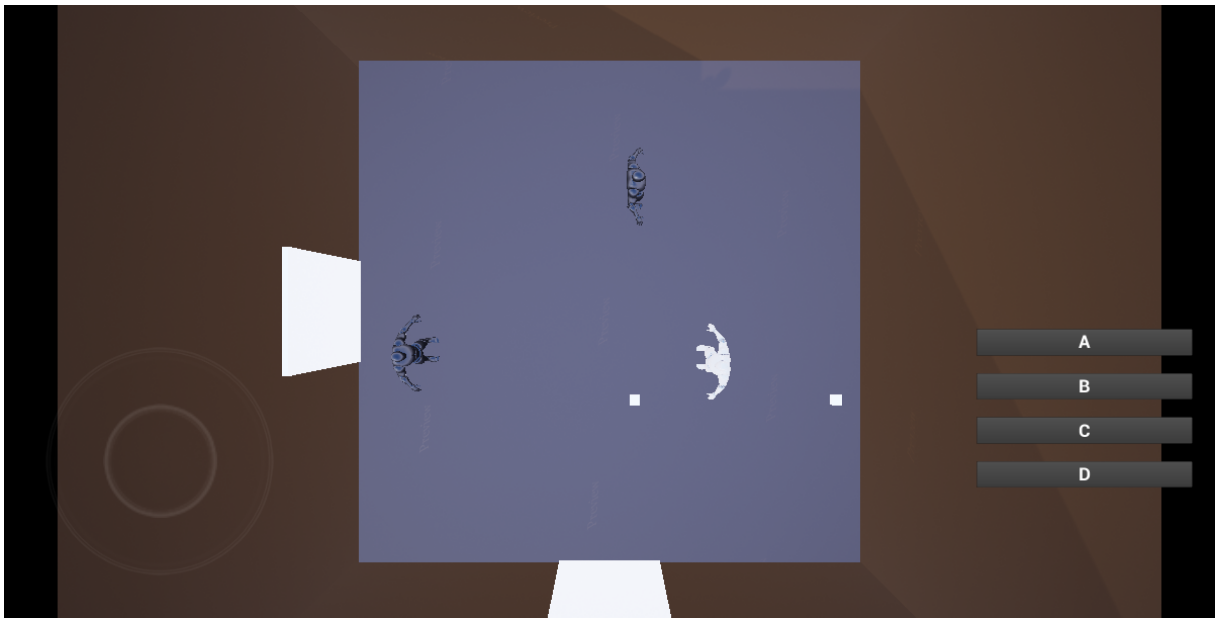

Fonte: Autores

### 7.2.2 Salas

As salas são as unidades básicas que compõem uma masmorra. Para a geração da *dungeon*, é necessário criar uma sala para cada elemento da *grid*. Basicamente, é feito um laço iterando por toda a grid, sendo cada índice responsável pela criação de uma sala.

Em todas as salas, há a possibilidade da existência de inimigos e de objetos coletáveis. Ambas as quantidades destes elementos é definida por um número gerado que encontra-se no intervalo de 0 a 4. O número máximo de inimigos e de itens foi definido após uma análise do espaço fixo contido em um quarto. Considerando a hipótese da quantidade máxima ser maior que a definida, haveria um grande risco da tela ficar poluída graficamente, tirando a qualidade da experiência do jogador.

Figura 32: Sala de uma dungeon gerada com arte não finalizada




Fonte: Autores

### 7.2.3 Inimigos

Os inimigos são personagens do jogo que causam perigo às tropas do jogador. Ou seja, para que a coleta de recursos de uma masmorra possa ocorrer sem problemas, é necessário que os inimigos sejam eliminados do local. Para isso, são enviadas tropas para combatê-los. A mecânica dos inimigos é simples: caso um membro da tropa entre em contato com um inimigo, seu corpo é empurrado para o sentido oposto e perde parte de sua vida (fator determinado e relacionado com o level do jogador).

Além disso, estas espécies também se movimentam em direção ao jogador quando este encontra-se no seu campo de visão, estacionando seu movimento apenas ao perdê-lo de alcance ou ao matá-lo. Por fim, os inimigos também têm pontos de vida e, para serem destruídos, este parâmetro deve chegar a zero. Para este fim, é necessário que os membros da tropa realizem ataques que o atinjam, causando dano e reduzindo assim, seus pontos de vida.

### 7.2.4 Recursos

Os recursos presentes no manager se encontram nas masmorras na forma de coletáveis. Estes itens ficam dispostos pelo mapa e são coletados quando um membro da tropa passa por cima. Além disso, após um inimigo ser destruído por um atacante, é possível que um coletável apareça no lugar, como forma de recompensa pelo feito. Ao sair da masmorra,

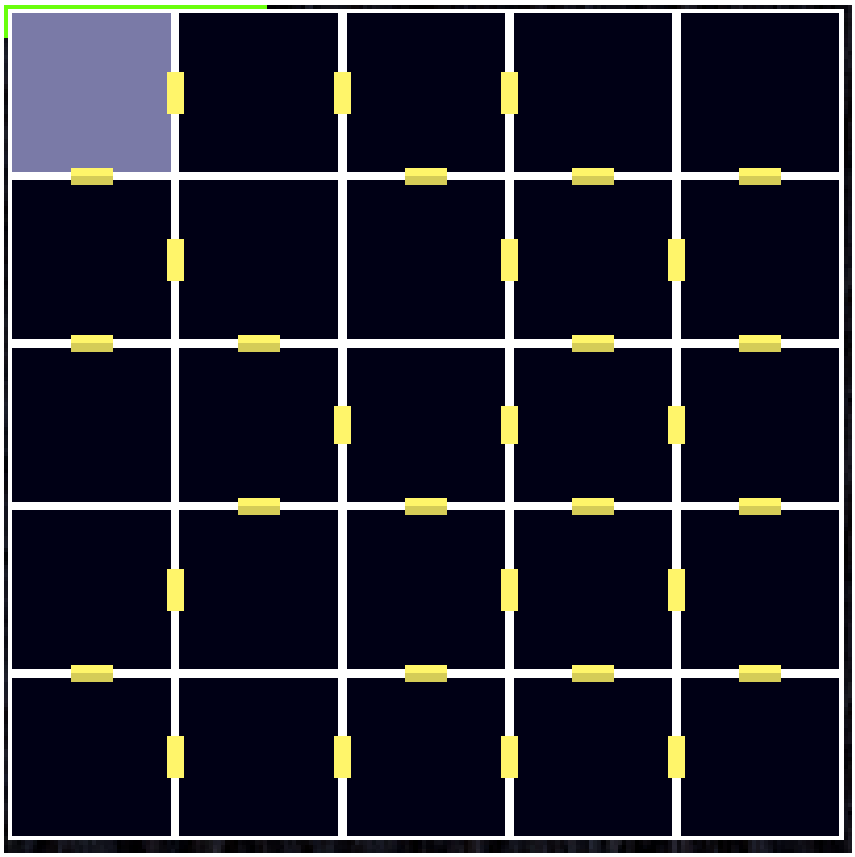
os itens coletados são traduzidos para recursos extras para o jogador, contribuindo assim, para a sua evolução.

### 7.2.5 Portas

As portas são de total importância para a estrutura de quartos presente nas masmorras. É através delas que a tropa vai para um quarto vizinho. Para que realmente se tenha a impressão de que os membros da tropa foram para um quarto vizinho, foi necessário transportá-los para o lado oposto da sala no qual a porta está posicionado. Este procedimento foi feito através de uma classe criada que marca a posição para a qual os membros devem ser movidos após o contato com a porta. Toda a porta possui este marcador e portanto, na etapa de geração das portas, que inclui a posição do ator no mundo, também foi necessária a criação destes marcadores.

Para facilitar a navegação do usuário, foi implementada uma interface de mini-mapa, presente no canto superior esquerdo da tela. O ponto de destaque do mini-mapa no contexto atual é a localização de todas as portas (em amarelo) presentes na masmorra, sendo possível uma análise dos trajetos que podem ser feitos para chegar a uma determinada sala. Como é possível saber a localização atual(sala com a cor mais clara) e as portas por todo o mapa, a navegação acaba tornando-se mais simples.

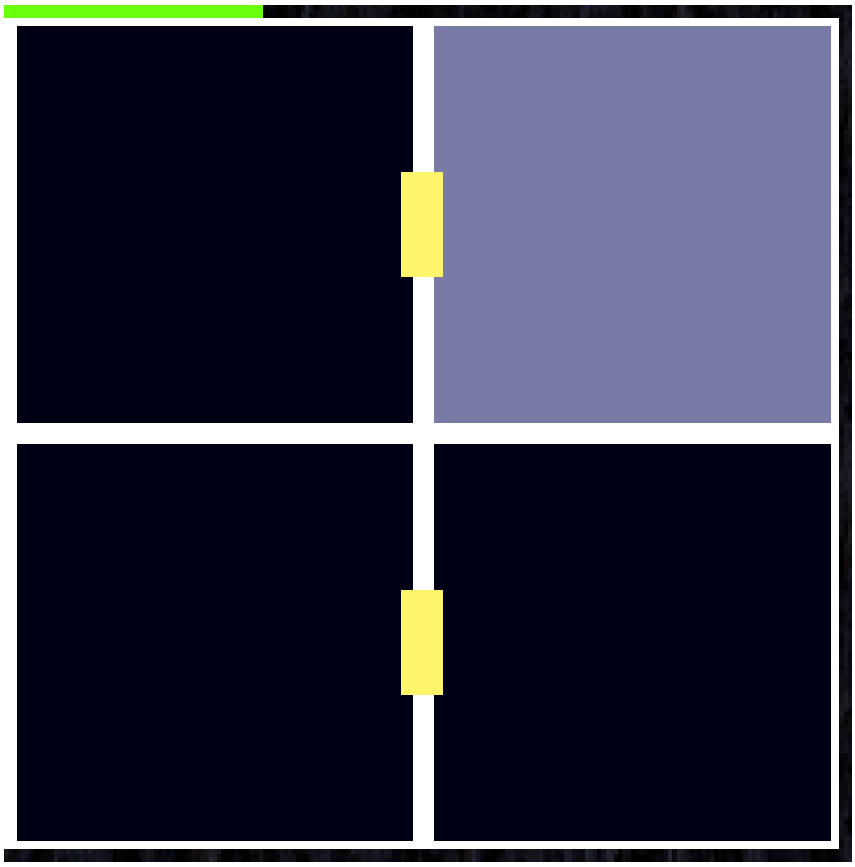
Figura 33: Mini mapa



Fonte: Autores

É importante ressaltar a importância do funcionamento correto do algoritmo de geração de portas, uma vez que, em caso de mal funcionamento, pode existir a possibilidade de algumas salas serem inacessíveis pela tropa, impedindo assim, que o jogador consiga concluir seu objetivo, que é derrotar todos os monstros presentes na *dungeon*, já que podem haver monstros nesta sala inacessível, que por consequência não serão derrotados. Na figura a seguir, é possível observar um exemplo de geração incorreta de portas: os quartos da segunda linha são inacessíveis.

Figura 34: Lógica de portas mal implementada



Fonte: Autores

## 7.3 Gameplay da parte de combate

O combate presente nas dungeons tem como principal objetivo simplificar o jogo e trazer controles intuitivos para o usuário. Basicamente, durante o combate é possível: movimentar-se através do joystick virtual presente no canto inferior esquerdo da tela; utilizar ataques do membro da tropa selecionado através dos botões localizados no canto inferior direito; e selecionar outro membro da tropa, através do toque no mesmo.

### 7.3.1 Ataques

Os ataques utilizados por membros da tropa foram modelados como projéteis que possuem um multiplicador de dano e um tempo de vida. Isto quer dizer que, caso o atacante tenha seu *linetrace* apontado para o inimigo e o tempo de vida for suficiente para que o projétil o atinja, ocorrerá um evento de causar dano no inimigo, implicando em diminuir seus pontos de vida e, caso estes cheguem a zero, o inimigo morrerá. O ataque utilizado pelo inimigo é de curta distância e é causado pelo contato entre o inimigo e o

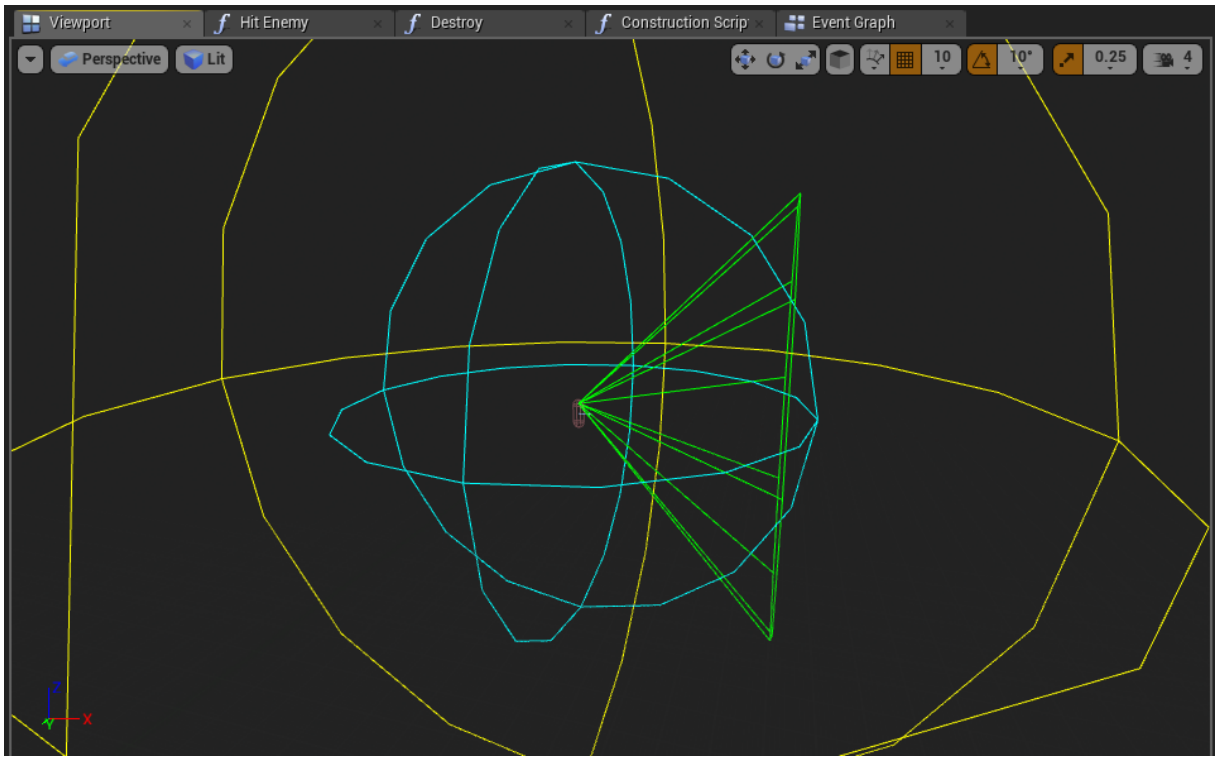
membro da tropa. Através dessa colisão, o aliado é empurrado para a direção oposta ao seu movimento e um dano é causado de acordo com a sua força. As mesmas condições de morte do inimigo se aplicam a morte do aliado: caso seus pontos de vida estejam abaixo de zero, o aliado é destruído.

### 7.3.2 Inteligência Artificial dos Inimigos

Os inimigos têm dois tipos de comportamentos possíveis: idle e perseguindo um membro da tropa. O segundo comportamento é ativado somente quando o aliado encontra-se em seu campo de visão, limitado por um ângulo a partir do seu ponto central e por uma distância máxima. Após a ativação do comportamento, o inimigo só voltará para o estado idle quando o membro da tropa estiver a uma distância maior que a definida pelo seu campo de visão ou quando cumpre seu principal objetivo: destruir o aliado através de ataques de curto alcance.

A figura a seguir ilustra a ferramenta utilizada para detectar se os aliados encontram-se no campo de visão do inimigo. As linhas em verde mostram o chamado *PawnSensing* da *Unreal Engine*. O seu funcionamento é simples: uma vez que um ator se encontrar dentro da área delimitada pelas linhas verdes, um evento será chamado e a partir disto, é possível mapear o que for necessário para engatilhar o comportamento desejado.

Figura 35: Campo de visão do inimigo




Fonte: Autores

### 7.3.3 Inteligência Artificial dos Aliados

A principal função dos aliados nas masmorras é dar suporte ao personagem selecionado, seja protegendo-no dos inimigos através de ataques, seja curando-no para que não morra. Foi implementada uma funcionalidade de trocar o personagem a ser controlado pelo jogador e, a partir do momento em que um personagem é selecionado, os outros membros da tropa formam uma fila atrás do membro escolhido e o seguem para auxiliá-lo a completar a *dungeon*. Para isto, possuem mecânica semelhante à implementada para os inimigos: ao detectar um inimigo em seu campo de visão, atacam-no causando dano e consequentemente, tornando mais fácil a missão do jogador. No caso dos curandeiros, ao invés de atacar os inimigos, curam seus aliados. Após executar a ação para auxiliar o membro da tropa selecionado, o aliado fica um tempo, que varia de acordo com o level do jogador, sem poder fazer nada além de seguir a fila composta pela tropa.

### 7.3.4 Condição de vitória

O principal objetivo da exploração da masmorra é derrotar todos os inimigos para garantir uma coleta de recursos segura e expandir os territórios. Portanto, para que o

usuário seja considerado vitorioso neste modo de jogo, é necessário que não exista nenhum inimigo ao final da exploração. Ao conquistar este objetivo, o jogo é pausado avisando o jogador do feito, dando as opções de continuar a exploração ou sair da masmorra. A primeira opção existe para que o a tropa possa coletar eventuais itens que ficaram nas masmorras. Ao sair, todos os itens coletáveis se transformarão em recursos adicionais e o jogador poderá ir além na sua jornada de exploração do terreno.

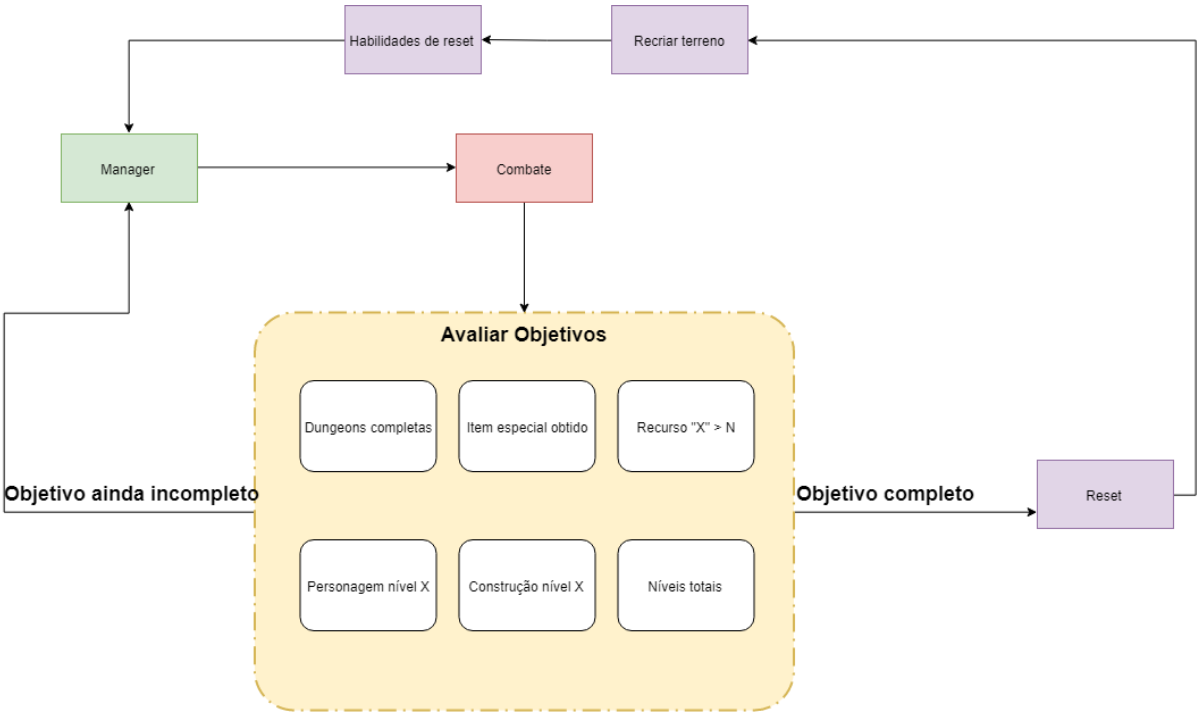
### 7.3.5 Condição de derrota

A exploração do jogador é considerada um fracasso quando não tem condições de cumprir o objetivo principal, que é derrotar todos os inimigos. Ou seja, caso todos os membros da sua tropa forem destruídos, tirando a possibilidade de cumprir com a missão, a exploração falha. Ao não conseguir cumprir o objetivo, o jogo é pausado e são mostradas duas opções no menu para o jogador: tentar novamente e sair da masmorra. Ao escolher tentar novamente, uma nova masmorra é gerada e o jogador tem uma nova oportunidade de explorá-la para expandir seu território no futuro. Ao escolher a segunda opção, o jogador volta para o *manager*, perdendo todos os recursos conquistados na masmorra anterior. Além disso, não consegue expandir seu território, tendo que tentar concluir a *dungeon* em um futuro próximo para continuar a exploração no jogo.

## 7.4 Missões ou Objetivos

Para impor metas atingíveis para cada partida, há um sistema de missões, geradas a cada partida.

Figura 36: Esquematização dos objetivos




Fonte: Autores

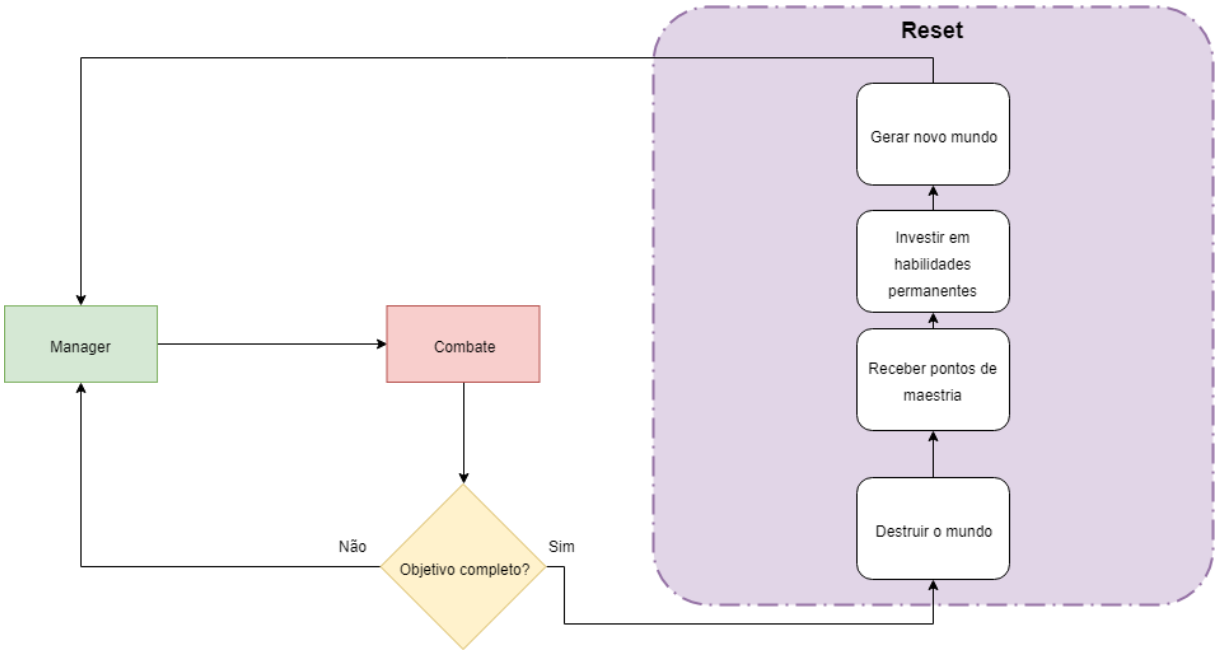
Existem 5 tipos de missões, cuja meta escala proporcionalmente à quantidade de *resets* realizados:

- Concluir um número específico de masmorras: o número de masmorras existentes no mapa aumenta conforme o jogador completa missões e reinicia o mundo. A cada masmorra completa, a dificuldade das próximas aumenta;
- Níveis totais: atingir um total de níveis somados entre todos os personagens e construções;
- Níveis de uma construção: atingir um nível específico com um tipo de construção específica. Por exemplo, atingir nível 50 com uma madeireira;
- Níveis de um personagem: atingir um nível específico com um tipo de personagem específico. Por exemplo, atingir nível 60 com um lenhador;
- Quantidade de recurso: atingir uma certa quantidade de um recurso específico. Por exemplo, acumular 15000 moedas.

## 7.5 Reset

O jogo checará constantemente para determinar a completude do objetivo da rodada. Quando a meta for atingida, o jogador receberá uma notificação propondo que ele inicie uma nova partida, reiniciando o mundo em troca de pontos de maestria.

Figura 37: Etapas de um reset




Fonte: Autores

Ao confirmar o reinício do mundo pela primeira vez, o jogador obterá acesso ao menu de árvore de maestrias, podendo adquirir habilidades que irão aumentar sua produção de recursos ou a performance de suas tropas de combate.

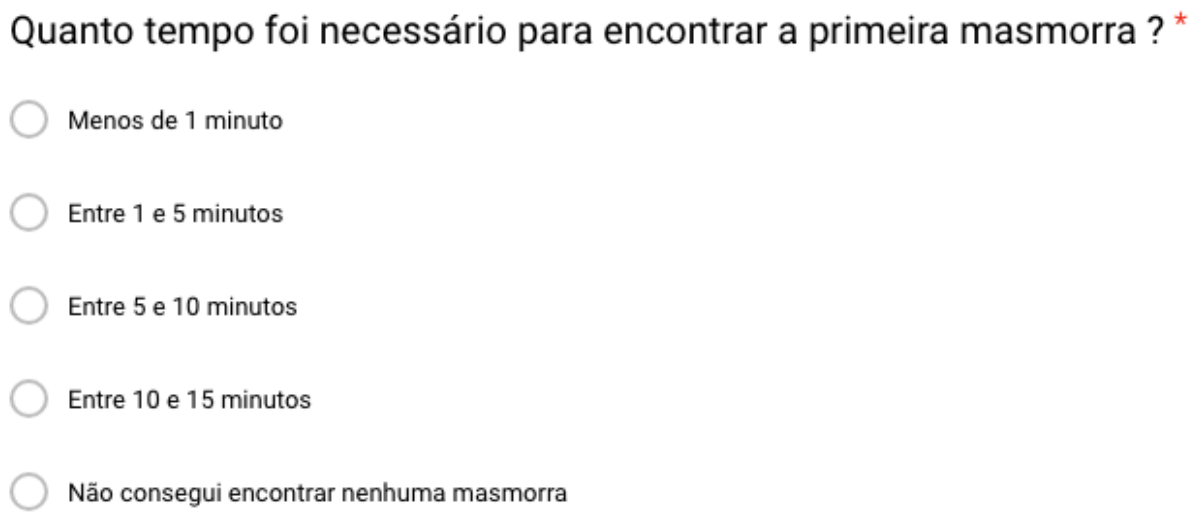
# 8 TESTES E AVALIAÇÃO

Para testar e avaliar o jogo desenvolvido, foi elaborado um pequeno questionário a ser respondido por cada *playtester*, contendo perguntas relacionadas aos aspectos mais relevantes do jogo: o *manager*, as *dungeons* e a expectativa de tempo no jogo. Além disso, ao final do questionário, será disponibilizado um espaço livre pra escrever comentários livres que não tenham relação com os temas das perguntas realizadas. O questionário será respondido após o usuário ter uma experiência de 15 minutos com o jogo.

## 8.1 Perguntas direcionadas

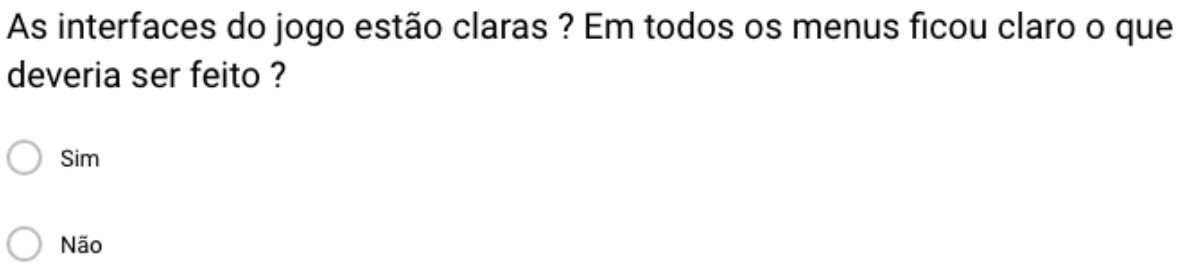
O maior objetivo das perguntas direcionadas é descobrir se os jogadores tiveram dificuldade para entender as mecânicas presentes no núcleo do jogo e se as interfaces estão claras e intuitivas. A maior preocupação com relação às interfaces está relacionada ao fato de que o projeto teve maior foco na junção do jogo incremental com a geração procedural de conteúdo, o que acabou acarretando em uma prioridade muito baixa para ajustes de interface.

Figura 38: Verificando se as interfaces do Manager estavam intuitivas




Fonte: Autores

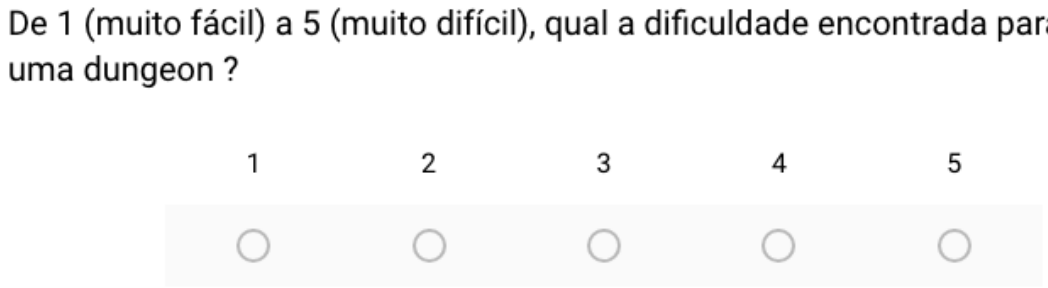
Figura 39: Verificando as interfaces do jogo estavam intuitivas




Fonte: Autores

Outro ponto abordado é a dificuldade dos jogadores nas masmorras. Foi feito um trabalho inicial de balanceamento de poder entre aliados e inimigos, que ainda pode conter discrepâncias e tornar este núcleo muito fácil ou muito difícil para o jogador, o que pode atrapalhar bastante na experiência do usuário. Após coletar as impressões, ajustes poderão ser feitos de acordo com a dificuldade encontrada.

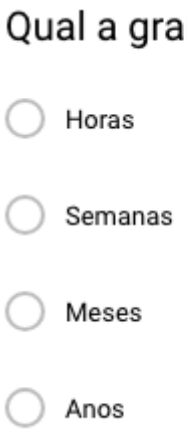
Figura 40: Verificando balanceamento entre aliados e inimigos




Fonte: Autores

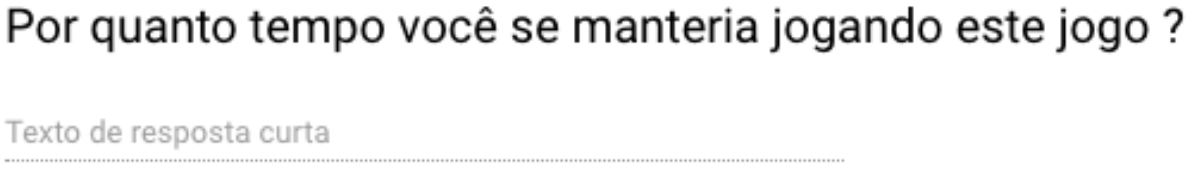
Além disso, as questões também visam saber se o jogo realmente cumpre com o objetivo, que é de aumentar o tempo de atividade do jogador com relação aos concorrentes presentes no mercado. Com base nas respostas relacionadas ao tempo, poderá ser possível observar quais aspectos precisam ser mudados para aumentar este índice cada vez mais.

Figura 41: Verificando quanto tempo durou a sessão mais longa do jogador




Fonte: Autores

Figura 42: Verificando se o jogo cumpriu com o objetivo inicial

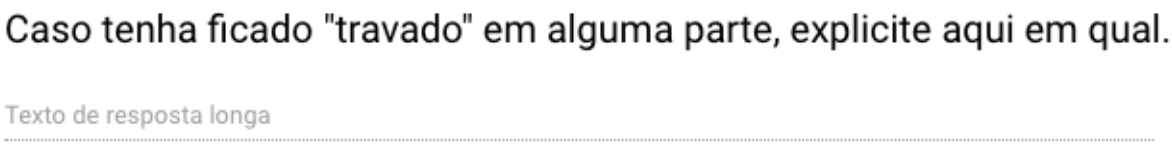



Fonte: Autores

## 8.2 Testes durante o desenvolvimento

Ocorreu um atraso com relação ao cronograma inicialmente feito e a rodada de testes com usuários não foi realizada. Contudo, ainda durante a implementação das funcionalidades do jogo, os desenvolvedores perceberam alguns problemas, sendo os principais: as interfaces do jogo e a performance gráfica. Devido ao tempo escasso, não puderam trabalhar para corrigir estes aspectos, que ficaram como débito técnico para correções futuras.

Figura 43: Pergunta genérica para coletar feedbacks de problemas de comportamento




Fonte: Autores

# 9 CONSIDERAÇÕES FINAIS

## 9.1 Conclusões

Para análise do resultado atingido, a tabela 2 da seção 6.3 foi recuperada e uma coluna com o estado final de cada feature foi adicionada, gerando a tabela 3 abaixo.

Tabela 3: Tabela de Resultados

| Feature | Núcleo | Prioridade | Resultado |
|---|---|---|---|
| Receber recursos | Gerenciador | Deve ter | Feito |
| Investir recursos | Gerenciador | Deve ter | Feito |
| Recrutar trabalhadores | Gerenciador | Deve ter | Feito |
| Recrutar tropas | Gerenciador | Deve ter | Feito |
| Alocar pontos do jogador | Geral | Deve ter | Feito |
| Formar um grupo de combate | Combate | Deve ter | Feito |
| Tropas coletando recursos | Gerenciador | Deve ter | Feito |
| Resetar o mundo | Geral | Deve ter | Feito |
| Construir edifícios | Gerenciador | Deveria ter | Feito |
| Evoluir construções | Gerenciador | Deveria ter | Feito |
| Evoluir tropas | Gerenciador | Deveria ter | Feito |
| Combate automático | Combate | Deveria ter | Parcialmente feito |
| Construir equipamentos | Gerenciador | Poderia ter | Não feito |
| Gerenciar equipamentos | Combate | Poderia ter | Não feito |
| Aprender habilidades | Combate | Poderia ter | Não feito |
| Definir estratégia de combate | Combate | Poderia ter | Não feito |
| Possuir objetivo | Geral | Poderia ter | Feito |
| Alterar objetivos | Geral | Poderia ter | Feito |

| | | | |
|---|---|---|---|
| Adicionar novas mecânicas | Geral | Poderia ter | Parcialmente feito |
| Combate PvP | Geral | Poderia ter | Não feito |

De forma geral, todas as *features* de alta prioridade foram implementadas, proporcionando um mínimo produto que pode ser jogado. Conteúdos relacionados à customização mais profunda de personagens e de interação multi-jogador não foram implementados. Existem dois tópicos parcialmente implementados: "Combate automático", que na primeira versão do jogo conta com ataques automáticos, porém não com progressão automática entre as salas de cada *dungeon*, e "Adicionar novas mecânicas", que não inclui todas as mecânicas propostas no design do jogo, mas inclui interação entre personagens e construções, mini mapa para visualização da *dungeon* e controles de câmera por gestos no *manager*.

Do ponto de vista dos requisitos não funcionais, o jogo é executável em celulares medianos, apesar de em alguns pontos terem leves quedas da taxa de quadros por segundo - problema que pode ser reduzido ao substituir os modelos atuais por modelos 3D com menor contagem de polígonos. Como não há comunicação com nenhum servidor, visto que se trata de um jogo totalmente para um jogador, não foi necessário garantir a segurança e a capacidade do servidor. Os dados são persistidos localmente, com gravações automáticas periódicas.

## 9.2 Contribuições

Durante o desenvolvimento do projeto alguns modelos 3D, texturas e bibliotecas foram utilizadas. Todas as contribuições e seus devidos créditos são dados a seguir:

- Modular Medieval Pack - Daniel Andersson (Creative Commons 0) - Módulos utilizadas para criar construções do manager
- Modular Village - Keith at Fertile Soil Productions (Creative Commons 0) - Módulos utilizados para criar construções do manager
- Landscape Assets v2 - eraccoon (Creative Commons 0) - Texturas de terrenos para diferenciar cada bioma
- Terrain Textures - Jenna Fearon (Creative Commons by Attribution 3.0) - Texturas de terrenos para diferenciar cada bioma

- Models for a Dungeon/Torture Chamber - Insomnia Arts (Creative Commons 0) - Estruturas de dungeon
- Mantis Hatchery Battle Creatures Pack - Mantis Digital Arts (Creative Commons by Attribution 3.0) - Inimigos das dungeons
- Simplex Noise - devdad - Biblioteca com implementação de Simplex Noise compatível com Unreal Engine 4
- Skill System - UnrealGaimeDev - Projeto utilizado para guiar o desenvolvimento do sistema de árvores de habilidades/maestrias
- Characters - Mixamo - Personagens de combate e suas animações, animações dos demais personagens
- Personagens customizados - Adobe Fuse - Modelos de personagens da vila, não combatentes
- Skill Icon Pack - Rexard - Ícones usados nas árvores de maestrias

## 9.3 Perspectivas de Continuidade

### 9.3.1 Transformar o jogo em produto

Um dos grandes objetivos deste trabalho era a publicação do jogo em plataformas virtuais e coletar *feedback* dos usuários para melhorar o jogo cada vez mais. A etapa de implementação não ocorreu como o esperado e as funcionalidades planejadas não foram desenvolvidas a tempo de realizar todos os testes e balanceamento. Ainda, para que o jogo fosse bem recebido enquanto produto, há a necessidade de trabalhar sua parte estética, que teve baixa prioridade no desenvolvimento do projeto.

Além da parte estética, seria produtivo para o resultado final implementar as *features* de baixa prioridade selecionadas durante a análise de requisitos.

### 9.3.2 Polimento de interfaces

A principal crítica dos usuários que testaram o jogo na primeira rodada de testes foi relacionada às interfaces, especialmente do ponto de vista estético. Alguns fluxos de telas não se mostraram totalmente agradáveis para todos os jogadores. Sendo assim, para o

lançamento do jogo seria necessário trabalhar sobre as interfaces, de modo a torná-las visualmente atrativas e intuitivas.

### 9.3.3 Áudio

O jogo carece de sons, elemento muito importante para compor a experiência de um jogador. Portanto, antes de um lançamento em lojas virtuais será necessário produzir e incorporar faixas de áudio ao projeto, tanto músicas de fundo como efeitos sonoros.

### 9.3.4 Melhoria visual

Para a primeira etapa do projeto, foram utilizados assets gratuitos disponíveis na internet e, devido à baixa prioridade dada ao quesito estética, em muitos casos o conjunto total de arte não correspondia à expectativa inicial da identidade visual de alguns personages e objetos, ou até contenha elementos destoantes entre si. Por este motivo, uma das perspectivas de continuidade é dedicar mais tempo na busca de *assets* que de fato correspondam à identidade visual desejada e, em caso negativo, contratar um profissional da área do design para fazer os assets de acordo com as especificações.

### 9.3.5 Controle de qualidade e correção de bugs

Poucos testes foram realizados e, por este motivo, com certeza ainda existem bugs de interações a serem corrigidos. Sendo assim, outra das perspectivas de continuidade é realizar mais rodadas de testes com usuários e com profissionais de controle de qualidade para detectar o maior número de problemas possível para trabalhar na resolução dos mesmos o mais rápido possível, melhorando assim, a qualidade de produto.

# REFERÊNCIAS


1 SAYRE, C. 10 Questions for Shigeru Miyamoto. *TIME Magazine*, July 2007.

2 EXTRACREDITS. *Idle Games - How Games Scratch Your Multitasking Itch*. 2014. Online Video. Disponível em: ⟨https://www.youtube.com/watch?v=g-LziX2HynI⟩.

3 SHAKER, N.; TOGELIUS, J.; NELSON, M. J. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. [S.l.]: Springer, 2016.

4 MAIN definitions of game - Oxford Dictionaries. 2014. Disponível em: ⟨https://en.oxforddictionaries.com/definition/game⟩. Acesso em: 09 abr. 2018.

5 ALLAN, R. *Core loop: the must have feature for every mobile app*. Disponível em: ⟨http://carnival.io/mobile-insights/core-loop-why-missing-it-can-mean-mobile-app-failure/⟩. Acesso em: 09 abr. 2018.

6 LOVATO, N. *How To Perfect Your Game's Core Loop*. 2017. Disponível em: ⟨https://gameanalytics.com/blog/how-to-perfect-your-games-core-loop.html⟩. Acesso em: 09 abr. 2018.

7 VILLAGE, C. *What is a Clicker/Idle/Incremental Game ?* 2015. Disponível em: ⟨http://clickervillage.com/what-is-a-clickeridleincremental-game/⟩. Acesso em: 18 abr. 2018.

8 TOGELIUS, J.; SHAKER, N.; NELSON, M. J. Constructive generation methods for dungeons and levels. In: SHAKER, N.; TOGELIUS, J.; NELSON, M. J. (Ed.). *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. [S.l.]: Springer, 2016. p. 31–54.

9 HAGEN, J. B. Five kingdoms, more or less: Robert whittaker and the broad classification of organisms. *BioScience*, January 2012.

10 ACADEMY, K. *Ruído de Perlin*.

11 KONGREGATE - Crusaders of The Lost Idols. 2018. Disponível em: ⟨https://www.kongregate.com/games/Playsaurus/clicker-heroes⟩. Acesso em: 01 abr. 2018.

12 STEAM Spy - Clicker Heroes. 2018. Disponível em: ⟨http://steamspy.com/app/363970⟩. Acesso em: 01 abr. 2018.

13 PLAY Store - Clicker Heroes. 2018. Disponível em: ⟨https://play.google.com/store/apps/details?id=air.com.r2gamesusa.clickerheroes⟩. Acesso em: 01 abr. 2018.

14 PLAY Store - Idle Poring. 2018. Disponível em: ⟨https://play.google.com/store/apps/details?id=com.gravity.poring.and⟩. Acesso em: 01 abr. 2018.

15 KONGREGATE Crusaders of the Lost Idols. 2018. Disponível em: ⟨http://www.kongregate.com/games/codename_enter/crusaders-of-the-lost-idols⟩. Acesso em: 01 abr. 2018.

16 PLAY Store - Crusaders of The Lost Idols. 2018. Disponível em: ⟨https://play.google.com/store/apps/details?id=com.kongregate.mobile.crusaders.google⟩. Acesso em: 01 abr. 2018.

17 STEAM Spy - Crusaders of The Lost Idols. 2018. Disponível em: ⟨http://steamspy.com/app/402840⟩. Acesso em: 09 abr. 2018.

18 KONGREGATE - Realm Grinder. 2018. Disponível em: ⟨http://www.kongregate.com/games/divinegames/realm-grinder⟩. Acesso em: 01 abr. 2018.

19 STEAM Spy - Realm Grinder. 2018. Disponível em: ⟨http://steamspy.com/app/610080⟩. Acesso em: 01 abr. 2018.

20 PLAY Store - Realm Grinder. 2018. Disponível em: ⟨https://play.google.com/store/apps/details?id=com.kongregate.mobile.realmgrinder.google⟩. Acesso em: 01 abr. 2018.

21 MEER, A. *Realm Grinder is the new AdVenture Capitalist, and it's destroying me*. 2017. Disponível em: ⟨https://www.rockpapershotgun.com/2017/06/22/realm-grinder-review/⟩. Acesso em: 01 abr. 2018.

22 NUNNELEY, S. *Minecraft has sold over 144 million copies and has 75 million monthly active users*. 2018. Disponível em: ⟨https://www.vg247.com/2018/01/23/minecraft-has-sold-over-144-million-copies-and-has-75-million-monthly-active-users/⟩. Acesso em: 01 abr. 2018.

23 SQUARE Enix Signs Long-Term Unreal Engine Licensing Agreement. 2017. Disponível em: ⟨https://www.unrealengine.com/ko/blog/square-enix-signs-long-trm-unreal-engine-licensing-agreement⟩. Acesso em: 02 dez. 2018.

24 GAUDIOSI, J. *Bandai Namco Goes For The KO With Unreal Engine 4 In Tekken 7*. 2017. Disponível em: ⟨https://www.unrealengine.com/en-US/developer-interviews/bandai-namco-goes-for-the-ko-with-unreal-engine-4-in-tekken-7⟩. Acesso em: 02 dez. 2018.

25 KAYSER, D. *Powered by UE4, Marvel vs. Capcom: Infinite Arrives in 2017*. 2016. Disponível em: ⟨https://www.unrealengine.com/en-US/blog/powered-by-ue4-marvel-vs-capcom-infinite-arrives-in-2017⟩. Acesso em: 02 dez. 2018.

26 MATHEWS, J. *Licensing, Royalties, Ownership, EULA and TOS Q&A*. Disponível em: ⟨https://forums.adobe.com/thread/1992542⟩. Acesso em: 02 dez. 2018.

27 ADOBE. *Mixamo*. Disponível em: ⟨https://www.adobe.com/devnet/author_bios/Mixamo.html⟩. Acesso em: 02 dez. 2018.

28 ATLASSIAN. *What is Git*. Disponível em: ⟨https://br.atlassian.com/git/tutorials/what-is-git\#version-control-with-git⟩. Acesso em: 02 dez. 2018.

# APÊNDICE A – GAME DESIGN DOCUMENT

## A.1 História

O jogo se passa do ponto de vista do líder de uma vila. O líder deve procurar expandir os territórios de seu povo, obtendo mais e diferentes recursos.

## A.2 Gameplay

O jogo conta com dois núcleos principais: o gerenciamento dos territórios e as batalhas nas cavernas.

Na primeira parte, o jogador irá alocar recursos em uma vila. Ele irá alocar camponeses, lenhadores, mineradores, metalúrgicos, entre outras funções para produzir recursos. Os camponeses gerarão comida, que permitirá manter mais pessoas na vila. Madeira, pedras e metais servirão para construir edifícios e equipamentos para tropas de combate (que serão utilizadas no segundo núcleo).

Na segunda parte, o jogador irá enviar um número pré-determinado de tropas de combates para cavernas ou outros tipos de dungeons. Ao concluí-las, o jogador irá ter a possibilidade de manter tropas no lugar, coletando recursos (itens, experiência), além de liberar novos terrenos a serem adquiridos e até descobrir novas profissões para suas tropas.

Dessa forma, o jogador irá acumular cada vez mais recursos, e os investirá em tropas para liberar novos terrenos mais produtivos e novas dungeons, que gerarão crescentemente mais recursos.

As tropas podem equipar armas e armaduras, além de aprenderem habilidades que podem auxiliar no combate.

Os combates acontecem de maneira automática. As tropas se movem até os inimigos e lutam conforme uma estratégia pré-estabelecida. Assim, a vitória se dará com uma boa estratégia, um bom setup de tropas e com base na força de cada componente.

Ao adquirir uma certa quantidade de territórios, o jogador pode escolher se tornar um “discípulo do apocalipse”, recomeçando em um mundo diferente, porém com atributos extra (afinal, ele acabou de destruir e criar outro mundo, algo ele aprendeu). Podemos trabalhar com a ideia de recurso de reset para distribuir em alguma árvore de habilidades e/ou adicionar mecânicas novas a cada reset, sendo que os objetivos de reset mudam

## A.3 Personagens

- Camponeses: produzem comida
- Lenhadores: produzem madeira
- Pedreiros: produzem rochas
- Mineradores: produzem ferro, pedras preciosas (usadas para habilitar habilidades), metais.
- Guerreiros: unidade básica corpo-a-corpo com dano médio, durabilidade alta e velocidade baixa.
- Arqueiros: unidade básica à distância, com dano alto, durabilidade baixa e velocidade alta.
- Magos: unidade básica de magias, com dano muito alto, durabilidade baixa e velocidade baixa.
- Padre: unidade básica de cura, com dano baixo, durabilidade média e velocidade média.
- Outras classes de combate
- Personagem principal: líder dos esquadrões de combate. O jogador pode selecionar seu estilo de combate no início de cada universo. À medida que novos universos são desbloqueados, o personagem principal pode avançar de classes.

## A.4 Controles

O jogador irá interagir com as construções e personagens, bem como seus menus associados utilizando toques na tela. Na seleção de formação de combate, poderá arrastar os personagens para os locais desejados na formação, bem como trocá-los de posição.

## A.5 Câmera

Em ambos os cenários, a visão do jogador será ortográfica top-down.

## A.6 Universo do Jogo

O mapa do mundo será recriado a cada playthrough, e será composto de trechos de terrenos proceduralmente gerados.

Os terrenos podem variar entre campos, florestas, pedreiras, cavernas, minas, vulcões, lagos, rios, montanhas, montanhas nevadas.

Cada parte do mundo estará conectada por acessos definidos pelas dungeons de batalha.

A estrutura interna das dungeons serão completamente isoladas do restante do mundo e serão compostas por salões rochosos ou temáticos, interligados por corredores coerentes.

## A.7 Inimigos

Os inimigos serão compostos por uma variedade de monstros. Alguns dos monstros básicos podem ser:

- Slimes (gelecas): Monstro mais básico do jogo. Pode haver variações de cor, forma, tamanhos e "materiais", alterando sua distribuição de atributos.
- Morcegos: Monstro voadores que não podem ser atingidos por ataques e habilidades corpo-a-corpo.
- Ogros: Monstros lentos e de alta durabilidade.
- Esqueletos: Podem ser guerreiros, magos ou arqueiros. Monstros frágeis e de dano alto.

Os inimigos possuem classificação de acordo com tipo de ataque: corpo-a-corpo, à distância ou magia. Existe um ciclo estilo pedra-papel-tesoura de vantagens e desvantagens em combate de acordo com o tipo de ataque. Uma vantagem significa inflingir mais dano em unidades do respectivo tipo de combate, bem como receber menos dano do tipo. O contrário ocorre para desvantagens, recebendo mais dano e inflingindo menos dano no tipo determinado. Unidades corpo-a-corpo possuem vantagens sobre unidades à distância e desvantagem sobre magos; unidades à distância possuem vantagens sobre magos e desvantagens sobre unidades corpo-a-corpo; magos possuem vantagens sobre unidades corpo-a-corpo e desvantagens sobre unidades à distância.

Ao derrotar os inimigos nas masmorras, pontos de experiência serão obtidos e há uma chance de adquirir itens.

Cada inimigo possui um conjunto de habilidades específicas, baseadas nas mesma habilidades que o jogador pode selecionar para suas tropas. Elas serão utilizadas automaticamente conforme tempo de recarga.