UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

Um jogo educacional baseado em conceitos de Gerência de Projetos com mecânicas de um jogo de entretenimento

Luís Filipe Oliveira da Silva

Orientador

Sean Wolfgand Matsui Siqueira

Catalogação informatizada pelo(a) autor(a)

S586 Silva, Luis Filipe Oliveira da
Um jogo educacional baseado em conceitos de Gerência de Projetos com mecânicas de um jogo de entretenimento / Luis Filipe Oliveira da Silva. -- Rio de Janeiro, 2019.
102.p

Orientador: Sean Wolfgand Matsui Siqueira.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2019.

1. Educação. 2. Jogos Digitais. 3. Gerenciamento de projetos. I. Siqueira, Sean Wolfgand Matsui, orient. II. Título.

RIO DE JANEIRO, RJ – BRASIL

NOVEMBRO DE 2019

Um jogo educacional baseado em conceitos de Gerência de Projetos com mecânicas de um jogo de entretenimento

Luís Filipe Oliveira da Silva

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Sean Wolfgand Matsui Siqueira (UNIRIO)

_____

Tadeu Moreira de Classe

_____

Geiza Hamazaki

Rio de Janeiro, RJ – Brasil.

Novembro de 2019

Agradecimentos

Gostaria primeiramente de agradecer ao professor Sean por orientar meu trabalho e pela paciência durante todo esse período em que estive desenvolvendo este projeto. Gostaria também de agradecer à Mariana, minha namorada, por todas as conversas sobre game design, pela revisão e por não me deixar desistir. Agradeço aos meus pais por todo apoio, não só na reta final, mas durante todo o curso. Agradeço ao meu irmão João por todas as conversas, apoio e pelas artes maneiras que se encontram no jogo e ilustram algumas páginas desse trabalho. Por fim, agradeço a todos os meus amigos que acompanharam o projeto e me apoiaram com testes e *feedback* sincero.

RESUMO

Quando usados na educação, jogos podem ser grandes aliados do aprendizado. Porém, muitos jogos educativos existentes não são divertidos como os jogos de entretenimento e acabam falhando em seu objetivo pedagógico. Este trabalho propõe a utilização de processos reais como mecânicas de um jogo de entretenimento, agregando valor educacional a ele sem comprometer o fator da diversão. O resultado foi a criação de um jogo chamado "*Dwarven Forge",* que teve como base para suas mecânicas conceitos da gerência de projetos, de forma que o jogador possa aprender e praticar tais conceitos de forma lúdica.

Palavras-chave: Gerência de Projetos, Jogos Digitais, Educação.

ABSTRACT

When used in education, games can be great allies of learning. However, many existing educational games are not as fun as entertainment games and end up failing in their pedagogical purpose. This paper proposes the use of real processes as mechanics of an entertainment game, adding educational value to it without compromising the fun factor. The result was the creation of a game called "Dwarven Forge", whose mechanics were based on concepts of project management, allowing the player to learn and practice these concepts in a ludic way.

Keywords*: Project Management, Video Games, Education.*

Índice

1 - Introdução

1.1 - Motivação

O Brasil tem cerca de 75.7 milhões de jogadores digitais, jogando nas mais diversas plataformas como PC, *smartphones* e *consoles*, segundo a *Newzoo*[1], sendo o 13º país maior no mundo neste mercado. Esses números mostram o forte relacionamento do brasileiro com os jogos. As tecnologias relacionadas vêm sendo aplicadas a áreas como construção civil, treinamentos, turismo e educação.

Quando usados na educação, jogos podem ser grandes aliados, pois ajudam no desenvolvimento de diversas capacidades como pensamento lógico, trabalho em equipe e comunicação eficiente, conforme exemplificado por Costa (2009):

> "o que se realiza no RPG[2] não é algo similar à cooperação, mas sim a própria cooperação. [...] não se realiza algo similar à interpretação de textos, e sim interpretações de textos legítimas. [...] tal igualdade estrutural com o objeto de conhecimento não existe em detrimento do jogo como entretenimento, pelo contrário, ela o promove."(2009, p. 8)

Além disso, conforme apresentado por McGonigal (2012), quando estão participando de um jogo, os jogadores ficam "intensamente envolvidos" e isso os deixa com a disposição mental e a condição física adequadas para gerar todos os tipos de emoções. Isso colabora com o aprendizado, pois deixa as pessoas mais dispostas e preparadas para serem as protagonistas dessa experiência.

Desta forma, a motivação está neste potencial educacional dos jogos, nos conhecimentos adquiridos no curso de Sistemas de Informação e em como transmitir parte deles na forma de um jogo eletrônico.

Dentro das disciplinas cursadas, a Gerência de Projetos foi escolhida por sua grande importância dentro das organizações. Segundo Silva (2015), a gerência de projetos

---

[1] Newzoo newzoo.com/insights/infographics/brazil-games-market-2018/

[2] Role Playing Games - Jogos de interpretação de papéis.

está se caracterizando como uma necessidade profissional tão básica quanto administração do tempo, técnicas de negociação, princípios fundamentais de gestão e uso de computadores, e os resultados de seu entendimento por parte dos profissionais envolvidos trazem diversos benefícios à organização, como aumento na produtividade, melhor conhecimento do negócio em que atuam, melhor compartilhamento de conhecimento gerado pelos projetos - o que, consequentemente, acaba reduzindo o retrabalho em projetos futuros.

1.2 - Problema

Segundo Costa (2009): "Percebe-se que os jogos com fins pedagógicos não são divertidos como os de entretenimento, e que – ironicamente – estes, quando utilizados para fins pedagógicos, são mais efetivos do que aqueles." Este problema acaba por afastar os jogadores, o que leva o produto final a falhar tanto em seu objetivo pedagógico, tanto como jogo. Segundo Costa (2009) isto se dá principalmente por "uma tendência em utilizar, quase que aleatoriamente, estruturas de jogos de entretenimento já consagrados para relacionar o conteúdo dos objetos de conhecimento, sem levar em conta as estruturas destes".

1.3 - Objetivo

O objetivo deste projeto é a aplicação dos conceitos de gerência de projetos como base para construção de um jogo digital de entretenimento, de forma que o usuário final, o jogador, possa aprender e praticar tais conceitos de forma lúdica e divertida.

1.4 - Justificativa

Segundo Koster (2014), a diversão nos jogos vem do domínio, da compreensão. É o próprio ato de resolver problemas que torna o jogo divertido. Ou seja, com jogos, o "vício" está em aprender. Um jogo de entretenimento desta forma, pode ser o suficiente para transmitir o conhecimento desejado, desde que construído a partir do objeto de conhecimento que se deseja ensinar.

Como apresentado por Costa (2009), um jogo educativo deve ser um jogo de entretenimento criado a partir da estrutura do objeto de conhecimento, e não um jogo de entretenimento adaptado. Desta forma, o público se sentiria atraído não pelo conteúdo pedagógico, mas pelo entretenimento que só esse jogo proporcionaria por meio das peculiaridades inerentes ao objeto de conhecimento. Em outras palavras, para o jogador, "como" está aprendendo é o que importa mais.

1.5 - Organização do Texto

Este trabalho está estruturado em oito capítulos. O segundo capítulo se chama "Estudo dos Temas", em que são apresentados os conceitos necessários para o entendimento do projeto. Em "Processo de *Design* de Jogos", o terceiro capítulo, são apresentados o processo usado para elaboração do jogo e suas etapas. "Ferramentas" é o quarto capítulo e nele são apresentadas as ferramentas utilizadas para desenvolver o jogo. No capítulo cinco, "Implementação: Código e *Assets*", é relatado o processo de implementação do jogo, desde criação de código até as artes. No sexto capítulo "Manual" há um guia do jogo. No sétimo, "Testes e Avaliação", há uma avaliação do que foi produzido assim como o resultado dos testes com usuários. Por fim, "Conclusão e Trabalhos Futuros" é o capítulo final.

2 - Estudo dos Temas

2.1 - Introdução

Neste capítulo são apresentados alguns conceitos por trás deste projeto, com a finalidade de melhor ilustrar as escolhas feitas e fornecer melhor entendimento dos temas abordados. Desta forma, são apresentados conceitos de gerência de projetos, jogos e *design* de jogos.

2.2 - Gerência de Projetos

Como o objetivo deste trabalho é a construção de um jogo de entretenimento baseado em gerência de projetos, é importante entender suas estruturas e funcionamento, mas antes que se possa falar de gerência de projetos, é importante também entender o que é um projeto. De acordo com o PMI (2018), projeto é um conjunto de atividades temporárias realizadas em grupo, destinadas a produzir um produto, serviço ou resultado únicos. O projeto é temporário pois tem um início e fim definidos no tempo, sendo assim, um escopo e recursos definidos, e é único no sentido de não se tratar de uma operação de rotina, mas sim de um conjunto de operações destinadas a atingir um objetivo específico.

E quando se fala de projeto, não se fala apenas de *software*. Projetos podem ser a reforma de uma casa, uma viagem, a construção de um automóvel ou o lançamento de um foguete. Independentemente da área de atuação, todo projeto deve ser gerenciado de forma especializada para apresentar os resultados, aprendizado e integração necessários para as organizações dentro do prazo e do orçamento previstos.

Segundo o PMI (2018), a gerência de projetos é a aplicação de conhecimentos, habilidades e técnicas para a execução de projetos de forma efetiva e eficaz. Ela é

importante pois permite que organizações unam os resultados dos projetos aos seus objetivos de negócios, o que possibilita que sejam mais competitivas dentro de seus mercados.

A gerência de projetos pode ser dividida em cinco grupos de atividades (SILVA, 2015):

- **Início:** Fazem parte desse grupo as atividades necessárias para a definição do projeto, bem como a autorização para seu início.
- **Planejamento:** Neste grupo, encontram-se as atividades relacionadas à definição e refinamento dos objetivos de projeto, além da definição e planejamento do escopo para que os objetivos sejam alcançados.
- **Execução:** Este grupo contém atividades relacionadas à integração de pessoas e recursos para a execução do que foi planejado para o projeto.
- **Monitoramento e Controle:** As atividades deste grupo têm como objetivo monitorar e fazer medições do progresso, identificando alterações e problemas que possam desviar o projeto de seus objetivos e tomando ações corretivas quando necessário.
- **Encerramento:** Neste grupo, é feita a formalização da entrega e aceitação do resultado do projeto, além de realizar o encerramento do projeto ou fase em que se encontra.

Foram apresentadas nos parágrafos anteriores cinco grupos de atividades da gerência de projetos. No entanto, é possível também dividi-la por áreas de conhecimento (PMI, 2018):

- **Gerenciamento da Integração:** é a área de conhecimento responsável por integrar e manter em sincronia todas as outras. Além disso, esta área de conhecimento também envolve desenvolvimento dos termos de abertura e

encerramento, planejamento do projeto, monitoramento e orientação do trabalho.

- **Gerenciamento de Escopo:** é a área de conhecimento responsável por definir as atividades que devem ser realizadas para entregar o produto, serviço ou resultado e garantir que somente o trabalho necessário para que o projeto seja finalizado. Também define critérios para determinar se o projeto foi completado.

- **Gerenciamento de Custos:** é a área de conhecimento responsável por estimar o custo do projeto, planejar e gerenciar os gastos para garantir que o projeto seja realizado dentro do orçamento definido.

- **Gerenciamento de Qualidade:** é a área do conhecimento responsável por garantir que o projeto satisfaça os objetivos e funções para os quais ele foi concebido. Normas e padrões de qualidade são definidos nos processos dessa área, bem como auditorias, buscando sempre a melhoria contínua.

- **Gerenciamento das Aquisições:** é a área do conhecimento responsável por adquirir bens e serviços externos à organização executora, além de gerenciar as entregas, pagamentos e contratos referentes a estas aquisições.

- **Gerenciamento dos Recursos:** é a área do conhecimento responsável por organizar e gerenciar a equipe do projeto, bem como a identificação da necessidade de aquisição de materiais, equipamento ou espaço para realização do trabalho.

- **Gerenciamento das Comunicações:** é a área do conhecimento responsável por garantir o desenvolvimento, recolhimento, distribuição, armazenamento, recuperação e entrega das informações sobre o projeto de forma oportuna e adequada.

- **Gerenciamento de Risco:** é a área do conhecimento responsável por monitorar e controlar constantemente riscos que possam aparecer ao longo do

projeto. O risco de um projeto é uma incerteza, que pode ter efeitos negativos ou positivos sobre algum dos aspectos do projeto, como escopo, custo, tempo ou qualidade. O principal objetivo desta área de conhecimento é aumentar a probabilidade e o impacto dos eventos positivos e reduzir a probabilidade e o impacto dos eventos negativos.

- **Gerenciamento do Cronograma:** é a área do conhecimento responsável por garantir que o andamento das etapas e atividades estejam de acordo com o cronograma e que a entrega do projeto ocorra no prazo, da iniciação ao encerramento do projeto. Para isso, as atividades do projeto são sequenciadas de acordo com sua importância, urgência e/ou prioridade, além de receberem uma estimativa do tempo necessário para que sejam concluídas.
- **Gerenciamento das Partes Interessadas:** é a área do conhecimento responsável por identificar os grupos, pessoas ou organizações que podem impactar ou ser impactados por uma decisão, atividade ou resultado do projeto e entender suas motivações e necessidades para melhor atendê-las.

Outro conceito importante em projetos é o ciclo de vida do projeto. O ciclo de vida de um projeto é o conjunto de etapas ou fases pelo qual um projeto passa desde sua concepção até sua entrega. Segundo Bigão (2015), muitos desses ciclos possuem fases similares entre projetos, mas raramente os ciclos idênticos, podendo ser consideradas como habituais as fases de Iniciação, Planejamento, Execução e Encerramento. Não é por acaso que elas possuem os mesmos nomes dos grupos de processos, pois, com exceção do grupo "manutenção e controle" que ocorre durante todas as fases dos projetos, elas apresentam as mesmas atividades que os grupos.

A dinâmica do projeto, por outro lado, é a forma como o projeto é executado, como o projeto se move entre essas fases. Para isso, existem diversos modelos e metodologias que têm como objetivo guiar o trabalho do gerente de projetos, estruturando essas fases e dando a ele uma melhor visão do que está sendo realizado.

São alguns destes o modelo em cascata, o modelo em espiral e as metodologias ágeis como o *Scrum*[3] e o *XP*[4].

Para este trabalho foi escolhido um modelo sequencial, assim como no modelo cascata, em que cada etapa só ocorre após a conclusão da anterior, pois assim é possível utilizá-lo como meio de demonstrar mais claramente ao jogador o funcionamento de cada etapa.

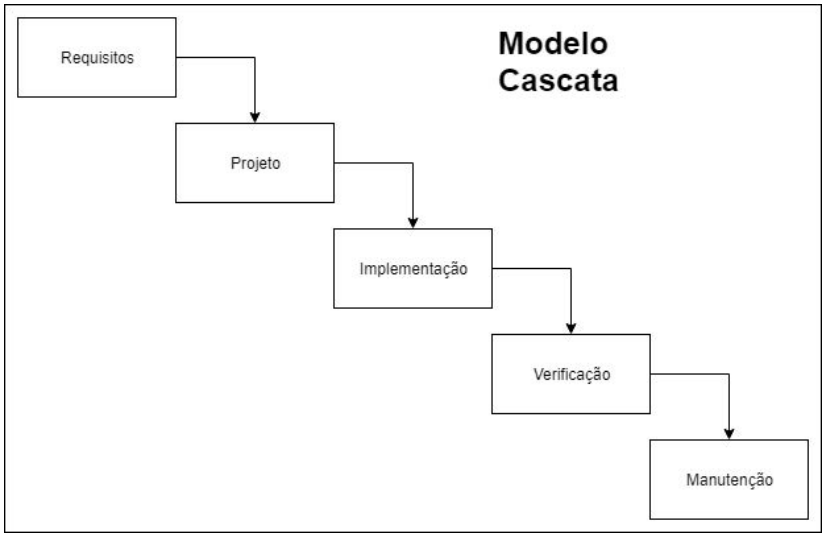

Figura 2.1 - Modelo em cascata.

Fonte: Figura produzida a partir do modelo de Royce (1970).

Idealizado por Royce (1970)[5], o modelo em cascata é um modelo sequencial em que, para se atingir uma fase, é necessário ter completado todas as atividades da fase anterior, conforme apresentado na Figura 2.1. Sendo assim, só é possível iniciar a fase de planejamento após o término da fase de iniciação e assim por diante, como uma cascata que flui de cima para baixo.

[3] Metodologia Scrum - https://www.scrum.org/resources/what-is-scrum
[4] Extreme Programming - https://en.wikipedia.org/wiki/Extreme_programming

[5] Modelo em cascata de Royce - https://pt.wikipedia.org/wiki/Modelo_em_cascata

2.3 - Jogos

Tendo em vista o objetivo deste trabalho, desenvolver um jogo, é importante também entender o que é um jogo, seus componentes e o que é o *design* de jogos.

O que são jogos? Parece uma pergunta trivial, porque todo mundo em algum momento na vida já jogou um jogo, seja ele digital, de tabuleiro ou até pedra-papel-tesoura. Mas o que faz com que um jogo seja considerado um jogo e não uma outra mídia, por exemplo? Segundo Schell (2011), existem dez qualidades que podem ser observadas em qualquer jogo:

1. Jogos são jogados voluntariamente.
2. Jogos têm objetivos.
3. Jogos têm conflitos.
4. Jogos têm regras.
5. Jogos podem levar à derrota ou vitória.
6. Jogos são interativos.
7. Jogos têm desafios.
8. Jogos podem criar valores internos próprios.
9. Jogos envolvem os jogadores.
10. Jogos são sistemas fechados e formais.

Segundo o autor, as qualidades observadas dos itens 2 ao 10 também podem ser observadas no processo que é realizado ao se tentar resolver um problema. Com base nesta lista, Schell (2011, p.37) define então, que "um jogo é uma atividade de

solução de problemas, encarada de forma lúdica", em que a primeira qualidade se traduz na "forma lúdica" da definição.

É importante ressaltar que o campo do *design* de jogos é relativamente novo e, por isso, ainda não existe uma regulamentação que possua definições e conceitos válidos universalmente. Dessa forma, é comum que cada autor ofereça sua própria versão sobre o que são jogos. Segundo McGonigal (2012), por exemplo, apenas 4 qualidades são necessárias para definir um jogo:

1. Metas
2. Regras
3. Sistema de feedback
4. Participação voluntária

Todas as outras características como interatividade, competitividade ou visuais complexos são complementares, um esforço para consolidar e fortalecer esses quatro elementos principais. Suits (2005), citado por McGonigal (2011, p.31), diz que "dedicar-se a um jogo é a tentativa voluntária de superar desafios desnecessários", o que para a autora é uma perfeita definição do que os jogos representam.

2.4 - *Design* de Jogos

Com as definições apresentadas, é possível avançar para a definição de *design* de jogos, uma área do *design* assim como *web design* ou *design* gráfico que tem como objetivo projetar jogos. Segundo Schell (2008, p. xii) "*Design* de jogos é o ato de decidir o que um jogo deve ser". Dessa forma, *game designers* são os responsáveis por definir todos os aspectos de um jogo, a experiência que se espera que jogador tenha, como um personagem se move pela tela, o tipo de arte que será aplicado, que tecnologias serão usadas, os tipos de sons, músicas etc. Ou seja, é no *design* de jogos

que são tomadas decisões a respeito do *gameplay* - ou, utilizando o termo em português, da jogabilidade.

Uma das formas de se pensar no *design* do jogo é por meio da Tétrade Elementar proposta por Schell (2008), cujo objetivo é agrupar e categorizar os diversos elementos que podem compor um jogo.

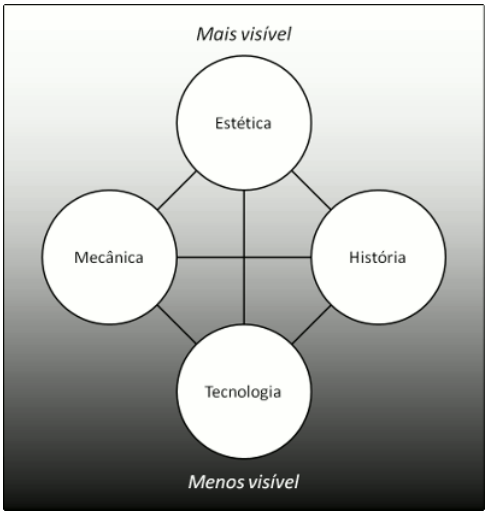

Figura 2.2 - Tétrade Elementar de Schell.

Fonte: A Arte de Game Design - O livro original (SCHELL, 2008).

Na tétrade, todos os elementos possuem igual importância e as decisões tomadas para cada um desses elementos impactam diretamente a experiência do jogador. A organização da Figura 2.2 tem como objetivo apenas explicitar a visibilidade de cada um dos elementos pelo jogador, sendo a estética o elemento mais visível (mais percebido) pelo jogador, seguido por mecânica e narrativa que ficam no meio termo dessa visibilidade, e por último a tecnologia. A seguir, uma breve explicação dos elementos da tétrade segundo Schell (2008):

- Mecânica: representa os objetivos e regras do jogo. A mecânica dita o que o jogador pode fazer, como pode fazer e por que deve fazer.

- Narrativa: é a sequência dos eventos que ocorrem no jogo, a história que é contada.
- Estética: está relacionada às sensações - não somente à parte visual, que é mais comumente associada à estética, mas com todos os sentidos - tato, audição etc.
- Tecnologia: é o meio que torna a existência do jogo possível - seja um computador, um celular, um tabuleiro ou simplesmente lápis, papel e dados.

Tendo como base as definições apresentadas, pode-se afirmar que o *design* de jogos é crucial para que o jogo atinja seus objetivos, sejam eles de entretenimento puro ou de aprendizagem, sendo este o foco do próximo capítulo.

3 - Processo de criação e *Design* final do jogo

3.1 - Descrição do processo

O processo usado no desenvolvimento deste projeto é o resultado de estudos e experiências adquiridos como desenvolvedor e *designer* de jogos. A área de *design* de jogos ainda não tem padrões bem definidos sobre como são feitos jogos, ou como devem ser feitos. O que são oferecidas são ferramentas para ajudar nas tomadas de decisões como a Tétrade de Schell, formas de documentação como o *Game Design Document* (GDD)[6] (sendo que até sobre este existem divergências a respeito do formato, de como e quando deve ser feito), ou técnicas mais ligadas ao processo criativo como sessões de *brainstorming* e afins. Com base neste ferramental e no estudo de metodologias de projetos, principalmente processos ágeis, foi elaborado este processo que tem como foco a identificação rápida do tema, experiência, mecânicas principais (core mechanics) e necessidades do projeto. O fluxo do processo tem como base as seguintes etapas:

1. Conceituação
2. Desenvolvimento
3. Validação

Na etapa de Conceituação, busca-se entender o problema, o que nas primeiras iterações pode ser o próprio tema proposto na iteração. Em seguida, listam-se as soluções possíveis para aquele problema, que podem vir a partir de uma sessão de *brainstorming* ou de uma base prévia de conhecimentos. Por fim, é escolhida uma dentre as soluções propostas.

[6] Game Design Document - https://en.wikipedia.org/wiki/Game_design_document

Na etapa de Desenvolvimento, a solução proposta é implementada. Essa implementação pode abranger desde o desenvolvimento de protótipos no início do projeto, como a configuração de efeitos especiais e o balanceamento das mecânicas em fases mais avançadas do projeto. Nesta etapa, está contido todo tipo de desenvolvimento: códigos, *sprites*, animações, músicas, efeitos sonoros, entre outros.

Na etapa de Validação, o resultado da etapa de desenvolvimento é validado. O critério para validação pode ser um teste interno, uma avaliação da qualidade do *asset*, um teste com usuários ou até mesmo uma reflexão sobre o quão divertido é o produto final (no caso das mecânicas). Caso seja aprovada, a solução é mantida e adicionada ao projeto; caso contrário, pode ser alterada para atender às necessidades do projeto (em uma nova iteração) ou substituída por uma outra solução.

Ao final da Validação, um novo problema é selecionado, iniciando uma nova iteração. Como pode ser visto na Figura 3.1, o processo se dá de forma cíclica, com cada iteração passando por todas as etapas apresentadas. Desta forma, é possível avaliar e validar o resultado final de cada iteração, fazendo as alterações necessárias para que o jogo resultante cumpra com seu objetivo principal.

A escolha de um processo mais simples como este ao invés da aplicação de metodologias mais conhecidas como o SCRUM ou até mesmo o modelo em cascata se dá pelo fato do tamanho da equipe que neste caso era de uma pessoa só.

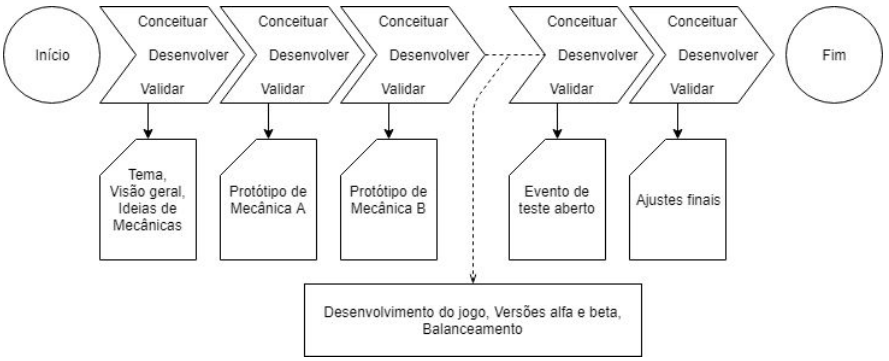

Figura 3.1- Exemplo de Processo de Conceituação, Desenvolvimento e Validação.

Fonte: Criado por Luís Filipe Oliveira.

3.2 - Aplicando o processo

Nos tópicos a seguir, são apresentados os principais passos no processo de *design* do jogo em questão, partindo da proposta inicial, que era: "A criação de um jogo sobre gerência de projetos." O processo iterativo funcionou por meio de refinamentos de *design*, ou seja, após as primeiras definições e a criação do primeiro protótipo foram sendo feitas análises e ajustes para que pudesse ser criado o próximo protótipo, até que se chegasse no resultado final. A tabela 1 detalha o passo a passo deste processo, bem como seus resultados.

Iteração:	**Resultado:**
1	Definição da proposta inicial do jogo
2	Definição da experiência do jogo
3	Definição das mecânicas principais
4	Protótipo 1 - Gerenciando um projeto de Software

5	Protótipo 2 - Gerenciando a produção de itens de histórias de fantasia
6	Definição do mundo do jogo
7	Definição dos personagens envolvidos
8	Definição do mapa/área de jogo
9	Definição das mecânicas de gerência de projetos
10	Definição do comportamento dos personagens
11	Protótipo 3 - Movimentação dos personagens na Unity
12	Definição do escopo dos itens que podem ser criados no jogo
13	Design da interface de usuário
14	Implementação dos itens e atividades na Unity
15	Protótipo 4 - Implementação do comportamento autônomo dos personagens
16	Desenvolvimento de assets gráficos - Interface de Usuário
17	Desenvolvimento de assets gráficos – Personagens
18	Desenvolvimento de assets gráficos – Itens
19	Desenvolvimento de assets gráficos – Cenário
20	Implementação dos assets gráficos na Unity
21	Implementação da Interface de Usuário

22	Protótipo 5 - Fluxo de jogo
23	Teste com usuários e Coleta de feedback
24	Revisão da Interface de Usuário
25	Implementação da nova Interface de Usuário
26	Revisão da movimentação de personagem
27	Implementação do sistema de pathfinding multi-agente
28	Correção de bugs críticos
29	Revisão do fluxo do jogo
30	Implementação de modo livre para Testes
31	Protótipo 6 - Sandbox, Nova Interface de Usuário

Tabela 1 - Iterações do processo de *design* do jogo.

Fonte: Criada por Luís Filipe Oliveira.

As iterações mostradas na tabela 1 relacionam-se a diversas definições que precisariam ser decididas durante a criação do jogo. Elas foram consideradas parte das iterações porque era necessário que o jogo possuísse características e mecânicas que estivessem de acordo com a proposta inicial - assim, a mesma definição poderia precisar ser revista mais de uma vez ao longo do processo, buscando-se sempre que determinado elemento apoiasse, cada vez mais, a proposta. Assim, cada novo protótipo não era um jogo inteiramente novo, mas sim um incremento ao protótipo anterior.

Foram criados seis protótipos até que se decidisse o *design* final para "*Dwarven Forge*". A identificação de que o sexto protótipo equivalia ao *design* final foi feita com base em parâmetros como o cumprimento da experiência desejada, a utilização de processos da gerência de projetos na mecânica principal, o funcionamento do sistema para que o jogo se tornasse um produto concretizado, a geração de um valor educacional de acordo com os elementos incluídos no jogo e, essencialmente, o cumprimento da proposta inicial deste trabalho.

O resultado dessas iterações e o *design* final apresentados também dão insumo para melhorias futuras, novas mecânicas, funcionalidades, personagens, itens etc. com base nos *feedbacks* e testes com usuários realizados ao final deste trabalho.

3.2.1 - A ideia

O primeiro passo foi refletir sobre a proposta de criar um jogo sobre gerência de projetos. Essa descrição poderia levar a muitas interpretações, que dependeriam do *designer* projetando o jogo. Por exemplo, poderia ser criado um jogo que apresentasse a gerência de projetos de maneira expositiva, em que o jogador navegasse pelo conteúdo, respondesse perguntas e ganhasse pontos por isso, ou talvez um jogo em que o jogador controlasse personagens engravatados em uma sala de empresa para que eles conseguissem fabricar algum *software*.

Porém, é importante entender sobre o que faria de um desses exemplos um jogo realmente sobre gerência de projetos. A gerência de projetos, neste caso, deveria permear todos os elementos do jogo, desde estética até jogabilidade, ao invés de ser apresentada mascarada sobre uma mecânica genérica de jogos de simulação.

Gerência não é uma mecânica nova em jogos. Diversos elementos da gerência de projetos podem ser observados em jogos já existentes, como o "RollerCoaster Tycoon"[7], em que o jogador administra um parque de diversões, ou jogos de

[7] Rollercoaster Tycoon - http://www.rollercoastertycoon.com/

estratégia (em tempo real ou por turnos), como os da série "Command & Conquer"[8], em que o jogador precisa gerenciar diversos recursos e unidades em uma tentativa de varrer seu oponente do mapa.

Um dos objetivos deste projeto é aplicação, em um jogo, de conceitos fundamentais da gerência de projetos. Sendo assim, é interessante que o jogador experiencie a fundo a gerência de projetos: que ela realmente faça parte do jogo, que esteja presente em todos os seus aspectos e elementos e não seja tratada apenas como tema superficial. Sendo assim, a proposta inicial pode ser atualizada: o jogo criado não deve ser apenas sobre gerência de projetos, o jogo deve *ser* gerência de projetos.

Com a nova descrição da proposta, é possível passar para o próximo problema a ser resolvido: a experiência.

3.2.2 - A experiência

Em sua obra, Schell (2008) apresentou mais de cem perguntas que um *designer* de jogos deveria fazer a si mesmo para ajudar a validar diversos aspectos de seu jogo. Para ajudar a definir a experiência do jogo, foram escolhidas dentre elas três perguntas relacionadas à experiência de jogo especificamente: Qual experiência desejo que o jogador tenha? O que é essencial para essa experiência? Como meu jogo pode captar essa essência?

Primeiramente, foi definido que, para este jogo, deseja-se que o jogador tenha a experiência de gerenciar um projeto. Ao ter essa experiência e participar ativamente na gerência de um projeto, o jogador pode aprender com mais facilidade enquanto se diverte, pois a diversão do jogo está diretamente relacionada ao aprendizado. Segundo Koster (2014), jogos podem servir como uma poderosa ferramenta de aprendizado, uma vez que toda informação necessária para o jogador já foi processada e estruturada, pronta para seu cérebro absorver. E, de forma complementar, Costa (2009) reforça que, para atingir seu potencial educacional, os

[8] Command & Conquer - https://www.ea.com/pt-br/games/command-and-conquer

jogos precisam que suas estruturas sejam similares às do objeto de aprendizado, que sejam perceptíveis ao jogador, que sejam necessárias para o jogador atingir os objetivos do jogo e acima de tudo, divertidas.

Então, respondendo às outras duas perguntas de Schell, para que o jogador tenha a experiência de gerenciar um projeto, é essencial que a estrutura do jogo seja similar à da gerência de projetos. O jogador deve usar a gerência de projetos para atingir o objetivo do jogo, ou seja, ela deve estar nas mecânicas principais do jogo. Com a gerência de projetos presente nas mecânicas principais, o jogo consegue ter o essencial para proporcionar ao jogador a experiência de gerenciar um projeto.

3.2.3 - Mecânicas principais

Para que o jogo proporcione a experiência de gerenciar um projeto, são usados os conceitos de gerência de projeto apresentados no capítulo 2. Com base no conceito de projeto, pode-se esboçar uma possibilidade (dentre muitas outras, como foi mostrado no tópico "3.2.1 - A ideia") de mecânica principal do jogo: um grupo realiza um conjunto de atividades com o objetivo de entregar um produto final.

Porém somente esta mecânica, apesar de ser estruturada similarmente a um projeto, não garante a parte da gerência, podendo o jogador ser apenas quem executa, não sendo parte da tomada de decisão. Portanto, é necessário adicionar elementos que possibilitem ao jogador influenciar o fluxo do projeto. De acordo com as decisões do jogador, o projeto correria bem - levando à vitória no jogo - ou mal - levando à derrota. Dessa forma, o jogador seria obrigado a pensar de forma crítica a respeito de suas ações dentro do jogo, ao mesmo tempo em que, sem perceber, estaria praticando a gerência de projetos.

Ao adicionar a perspectiva da gerência ao jogo, pode-se ter como mecânica principal: o jogador coordena um grupo de personagens, atribui a eles atividades e acompanha

sua execução, para que possam finalizar a criação de um produto e entregá-lo dentro do limite de tempo proposto.

A partir da proposta de mecânica principal e dos grupos de processos de gerência de projetos apresentados no capítulo 2, pode-se definir um fluxo de jogo que seja similar à gerência de projetos:

- **Iniciação:** O jogador recebe um resumo do que é o projeto. O objetivo é apresentado ao jogador.
- **Planejamento:** O jogador seleciona personagens para sua equipe. O jogador atribui atividades a esses personagens.
- **Execução:** O jogador dá início à execução. Os personagens executam as atividades de forma automatizada.
- **Controle e Monitoramento:** O jogador acompanha a execução das atividades.
- **Encerramento:** Os personagens finalizam o produto pedido. O jogador realiza a finalização do projeto.

Neste momento, o objetivo do jogo é conseguir entregar um produto dentro do tempo esperado. Partindo disso, pode-se esboçar o primeiro protótipo.

3.2.4 - Protótipo 1 - Gerenciando um projeto de software

A primeira proposta de tema de jogo foi o ambiente de uma empresa de desenvolvimento de software. Esta temática apresenta um ambiente atual, onde o jogador assume o papel de gerente e os personagens não jogáveis são seus funcionários. O jogador deve atribuir atividades para os funcionários e estes as

realizarão, de forma ordenada, até que o produto final - neste caso, um software - esteja completado.

Para representar este jogo, foi criado um protótipo de papel com cartas (leia-se recortes de papel escritos à mão) representando os personagens, as atividades e o projeto; turnos usados como unidade de tempo; além de papel e lápis para anotar as mudanças de estado do jogo.

- **Carta Projeto:** Contém um nome, um resultado esperado e tempo limite.
- **Carta Atividade:** Contém um nome, um resultado esperado e um pré-requisito.
- **Carta Personagem:** Contém um nome.

A partir disso, pode-se encaixar estes elementos na mecânica:

A. O jogador recebe uma carta de projeto.
B. O objetivo é terminar o resultado esperado dentro do limite de tempo.
C. O jogador seleciona as cartas de personagem que farão parte da equipe.
D. O jogador atribui atividades para a equipe.
E. O jogador realiza a execução do projeto (pois, neste momento, não há automatização).
    a. O jogador atribui uma carta de atividade a cada um de seus personagens.
    b. Caso possua um pré-requisito não concluído, o personagem fica em espera.
    c. Caso não possua, o personagem executa a atividade.

F. O jogador repete a execução até que não haja mais atividades na pilha, ou que o tempo tenha acabado.

G. Em caso de finalização das atividades o jogador ganha a partida, caso contrário, perde.

É possível ter uma ideia do jogo a partir disso, porém parecem faltar algumas informações como: Quanto tempo um personagem leva para terminar uma atividade? O que é um pré-requisito? E o resultado esperado?

O requisito, neste caso, é o produto de uma outra atividade, algo que se desenvolve como um documento ou parte do software. Ou seja, requisito e resultado podem ser representados por um único elemento, que chamaremos de Artefato.

- **Carta Artefato:** Contém um nome.

Para este jogo, foi levado em consideração que, de forma geral, pessoas diferentes possuem características e habilidades diferentes, o que interfere em sua capacidade de realizar tarefas. Além disso, diferentes atividades pedem diferentes tipos de habilidade. O conjunto de habilidades para este protótipo foi uma simplificação de habilidades comuns ao desenvolvimento de software, apenas com o intuito de testar o fluxo do jogo, sendo estas: Análise, Programação, Interface e Teste, cada uma contendo um valor de 1 a 5. A habilidade necessária para a atividade possui um valor de 1 a 6 e representa a dificuldade daquela atividade. Sendo assim, o tempo de execução de uma atividade é igual à dificuldade da atividade menos o nível de habilidade do personagem, nunca podendo ser menor que 1.

Esta nova regra leva a uma alteração das cartas de Personagem e Atividade:

- **Carta Atividade:** Contém um nome, um resultado esperado, um pré-requisito e uma *habilidade necessária*.

- **Carta Personagem:** Contém um nome e *habilidades (Análise, Programação, Interface e Teste)*.

Com as novas cartas, é possível atualizar o fluxo de jogo, completando as lacunas existentes e possibilitando a realização de testes. O novo fluxo passa a ser:

A. O jogador recebe uma carta de projeto.

B. O objetivo é terminar o resultado esperado dentro do limite de tempo.

C. O jogador avalia as cartas de personagem disponíveis.

D. O jogador seleciona as cartas de personagem que farão parte da equipe.

E. O jogador verifica as atividades necessárias para realizar o projeto.

F. O jogador atribui atividades na pilha de cada personagem.

G. O jogador realiza a execução do projeto (pois, neste momento, não há automatização).

    a. O jogador atribui a carta de atividade do topo da pilha a cada um de seus personagens.

    b. Caso o jogador não possua o pré-requisito em sua mão, o personagem fica em espera.

    c. Caso possua, o personagem executa a atividade.

    d. Ao fim da atividade o jogador adiciona o artefato à sua mão.

H. O jogador repete a execução até que não haja mais atividades na pilha, ou que o tempo tenha acabado.

I. Em caso de finalização das atividades o jogador ganha a partida, caso contrário, perde.

Com o fluxo finalizado, o protótipo pode ser testado. Neste momento, foi feita uma partida de teste com duas pessoas, em momentos separados, tendo ambos o costume de jogar jogos digitais ou físicos e com alguma experiência em desenvolvimento de jogos. Por meio de uma partida guiada, foi possível observar que a mecânica funcionou bem, mas a temática tornou alguns elementos do jogo abstratos, como um "módulo de *login*" ou um "caso de teste", que são conceitos com as quais nem todas as pessoas têm contato, o que pode causar um distanciamento entre jogador e jogo. Ou seja, o tema é interessante, mas não necessariamente "divertido".

3.2.5 - Protótipo 2 - Gerenciando a produção de itens de histórias de fantasia

Como segunda versão de proposta, a ideia foi algo mais tangível e de mais fácil reconhecimento pelos jogadores: itens fantásticos. No caso de uma situação parecida com o mundo real, diferenciar dentro do jogo o ícone de dois documentos ou criar uma representação para pedaços de código ou testes pode ser complicado, causando dúvidas ao jogador. Porém espadas, machados e escudos possuem silhuetas bem únicas, o que torna sua interpretação mais rápida pelo cérebro humano.

A inspiração veio de universos fantásticos como o de O Senhor dos Anéis e de contos e lendas da mitologia nórdica. Nestes, os anões são vistos como exímios artesãos, conhecedores da arte da forja e capazes de criar artefatos incríveis como o *Mjölnir*, o martelo de Thor na mitologia nórdica. Por isso, decidiu-se por utilizar anões como personagens jogáveis neste projeto.

Adaptando-se ao novo tema, é possível pegar elementos de mecânica propostos anteriormente e reorganiza-los a fim de encaixar-se na nova proposta. Desta forma, as cartas passam a ser:

- **Carta Projeto:** Contém um nome, resultado esperado, tempo limite e estoque inicial.
- **Carta Atividade:** Contém um nome, resultado esperado, materiais necessários e habilidade necessária.
- **Carta Personagem:** Contém um nome e habilidades (Forja, Refinamento, Têmpera e Montagem).
- **Carta Item:** Contém um nome.

A primeira mudança foi em relação à atividade, que agora tem uma lista de materiais necessários. Esses materiais são itens, que podem ser resultado de outra atividade ou adquiridos externamente. Mecanicamente são muito parecidas, porém esta mudança cria uma abertura para a estética e narrativa que podem impactar a visualização e o entendimento do jogo pelo jogador.

A segunda mudança foi em relação às habilidades dos personagens, que foram alteradas para refletir alguns dos processos comuns na representação da forja em jogos. Já que o objetivo de ensino do jogo não é a forja dos itens, é possível usar processos simplificados e focar em sua gerência. Na carta projeto, entra o estoque inicial, que representa os itens disponíveis para o início do projeto. E por fim, a alteração da carta Artefato para Item, o que representa melhor o universo em que o jogo se insere.

Sendo assim, o fluxo de jogo passa a ter a seguinte forma:

A. O jogador recebe uma carta de projeto.
B. O objetivo é terminar o resultado esperado dentro do limite de tempo.
C. O jogador adiciona à sua mão as cartas de item indicadas.
D. O jogador avalia as cartas de personagem disponíveis.

E. O jogador seleciona as cartas de personagem que farão parte da equipe.

F. O jogador verifica as atividades necessárias para realizar o projeto.

G. O jogador atribui atividades na pilha de cada personagem.

H. O jogador realiza a execução do projeto (pois neste momento não há automatização).

    a. O jogador atribui a carta de atividade do topo da pilha a cada um de seus personagens.

    b. Caso tenha os materiais necessários, o jogador atribui os materiais ao personagem e o personagem executa a atividade.

    c. Caso contrário, o personagem fica em espera.

    d. Ao fim da execução, o jogador descarta os itens usados na atividade e adiciona à sua mão o resultado da atividade.

    e. O jogador repete a execução até que não haja mais atividades na pilha, ou que o tempo tenha acabado.

I. Caso o jogador tenha o item que representa o resultado do projeto na mão, ele ganha a partida, caso contrário, perde.

Esta nova abordagem, apesar de mecanicamente similar à do protótipo anterior, apresenta novas interações do jogador, como o gerenciamento dos materiais, interação esta que permite que o jogador veja o resultado de suas escolhas e o resultado do trabalho dos personagens que está controlando por meio dos itens transformados. Para avaliar esta nova versão, o mesmo teste foi realizado com os mesmos participantes, o que resultou numa preparação mais rápida e uma mais fácil identificação do que se estava sendo feito, por conta de uma transformação visual pelo qual os itens passam.

3.3 - Design Final

O resultado final dessas iterações foi organizado em uma estrutura tópicos incluindo a tétrade de Schell (2008), conforme pode ser visto a seguir.

3.3.1 - O Tema

O tema decidido para o jogo foi: gerenciar a produção de itens fantásticos em uma oficina de anões ferreiros. Gerenciar um projeto ainda é um elemento central para o jogo, mas agora acompanhado por uma ambientação. O jogador não apenas gerencia um projeto, agora ele gerencia anões ferreiros na criação de itens fantásticos. A adoção de um tema fantástico, como dito anteriormente, facilita o aprendizado e ajuda a proporcionar uma boa experiência de jogo.

3.3.2 - Narrativa

Com a escolha do tema, a criação de uma narrativa que o suportasse era essencial. Neste jogo, o jogador é o líder de uma forja mágica de anões e o seu papel é garantir que seus subordinados executem os projetos encomendados por seus clientes, entregando o que for pedido dentro das limitações propostas. Para isso, o jogador deve conhecer as habilidades de sua equipe, planejar as atividades, iniciar e finalizar o projeto e resolver problemas que possam acontecer durante a execução.

Como foi apresentado anteriormente, nos mundos de fantasia, os anões são capazes de criar verdadeiras obras primas na forja, mas para que os anões sejam capazes de chegar a esse nível de habilidade manual, pode-se supor que exista um processo de aprendizado e anos de prática desses grandes ferreiros das lendas - e foi justamente a partir desta suposição que foi desenvolvida a narrativa para este jogo.

Como apresentado anteriormente, as habilidades dos personagens foram escolhidas pensando no processo de forja de armas, como geralmente é apresentado nesses mundos fantásticos, adicionados de um pouco de magia e outros elementos incomuns. Os anões devem coordenar seu trabalho e utilizar bem suas habilidades

individuais para conseguir fabricar corretamente o item e deixar o comprador satisfeito. Nesta versão, as habilidades disponíveis são:

- **Refinamento:** é o processo de transformar materiais brutos em algo utilizável na forja. O mais comum é a transformação do minério em lingotes.
- **Forja:** neste processo, se dá a forma geral do item que se está tentando criar, por exemplo a criação da lâmina para uma espada.
- **Têmpera:** neste processo, o metal incandescente é mergulhado em água ou óleo gelados, para aumentar a resistência e rigidez do metal.
- **Montagem:** é o último processo, onde todas as partes do artefato são encaixadas e presas, atingindo assim sua forma final.

Para testar os protótipos, foram criados três personagens com habilidades fixas. Para versões futuras, estes podem ser substituídos por personagens com habilidades aleatórias, tornando assim as partidas ligeiramente diferentes a cada vez que se joga. É importante, porém, ter em mente que a adição de elementos aleatórios em um jogo tende a acrescentar grande complexidade, dificultando o balanceamento do jogo.

A evolução na dificuldade do jogo, do ponto de vista da narrativa, poderia ser representada pela fama que a forja vai adquirindo. Conforme o jogador fosse completando os desafios, mais seres passariam a se interessar pelas maravilhas geradas nela, o que também aumentaria a complexidade dos pedidos. O jogador começaria produzindo lingotes de ferro para uma outra forja mais famosa e, com tempo e experiência, poderia um dia chegar a criar uma "espada suprema da luz divina" encomendada por um lorde elfo. Porém, o aumento da dificuldade ao longo do jogo será implementado em versões futuras.

3.3.3 - Mecânica

As mecânicas principais propostas no Protótipo 2 foram mantidas, porém alguns elementos foram adicionados visando a versão final e digital do mesmo, e por versão

digital entende-se um jogo feito para ser jogado em consoles, aparelhos móveis e/ou computadores. Uma versão digital possibilita ao jogo representar e agir sobre aspectos puramente mecânicos, deixando para o jogador apenas o essencial daquela experiência.

Nessas iterações dois novos elementos foram adicionados ao jogo, as Salas e o Inventário. As salas são áreas do jogo em que um personagem executa algum tipo de ação, enquanto o inventário é onde se encontram os itens envolvidos naquela partida (sejam eles os disponibilizados inicialmente ou os criados pelos personagens). Para esta versão inicial, três tipos de sala foram criados: Saguão principal, Estoque e Fornalha. O saguão principal é a área onde os personagens começam no jogo. Nele existe um painel de tarefas que os personagens consultam para descobrir se possuem alguma tarefa pendente. O estoque é a área onde os personagens interagem com o inventário, seja para buscar ou entregar itens. Por fim, a fornalha é a área onde os personagens executam suas atividades, e exclusivamente nesta versão, acumula os usos das quatro habilidades, forja, refinamento, têmpera e montagem.

Com essa adição, é necessário que o personagem tenha independência das ações do jogador, ou seja, que por meio de ações simples do jogador como atribuir uma atividade a um personagem, este seja capaz de realizar todas as ações necessárias para que aquela atividade seja executada, como buscar itens no estoque, executar a transformação do item na fornalha, devolver o resultado da transformação no estoque, retornar ao painel de tarefas para buscar outras atividades, além de mover-se entre estas salas. O personagem deve possuir algum tipo de inteligência.

Outro ponto explorado nas iterações foi a identificação das áreas de conhecimento de projeto no jogo:

- **Gerência de Escopo**: acontece na criação e planejamento das atividades, quando o jogador define quais serão executadas para a criação do produto final.

- **Gerência de Recursos**: acontece na seleção de personagens que possuam habilidades que atendam às necessidades do projeto, no controle e na aquisição de materiais.

- **Gerência de Cronograma**: acontece na divisão das atividades entre os personagens, para que estes terminem o produto final e no acompanhamento da realização das mesmas.

- **Gerência de Comunicação**: acontece na distribuição de atividades. As atividades ficam disponíveis para os personagens no painel do saguão, tendo estes que voltar ao saguão para consultar novas tarefas para executar.

- **Gerência de Integração:** acontece no acompanhamento do projeto, na atenção com as áreas para que estas funcionem em conjunto e o projeto aconteça da melhor forma possível.

Para esta versão do jogo, apenas as áreas apresentadas acima foram implementadas. Para as demais, foram pensados alguns exemplos de mecânicas que poderiam funcionar, podendo serem implementadas em versões futuras do jogo:

- **Gerência de Risco:** eventos aleatórios que impactam o rendimento dos personagens, como doenças, quebra de equipamentos, acidentes. Aquisição por meio de aventuras, em que se envia um personagem em busca de um item precioso, podendo este custar o próprio personagem.

- **Gerência de Aquisição:** mercadores vendem materiais para sua oficina, manutenção de equipamentos, contratação de novos personagens.

- **Gerência de Custos:** adição de uma moeda ao jogo, que pode ser usada para aquisição de materiais, serviços e pagamento dos personagens.

- **Gerência de Qualidade:** adição de um modificador de qualidade aos itens, adicionando atividades que melhoram a qualidade do item. Poderia também

estar envolvido na diferença entre nível do item e valor de habilidade do personagem.

- **Gerência de Partes Interessadas:** enfatizar o papel do cliente no jogo, podendo adicionar elementos de negociação, mudança de requisito e escopo.

Por fim, nesta proposta de design do jogo, o jogador passa pelas etapas mais comuns do ciclo de vida de projetos, de forma rápida e sem a pressão de acertar de primeira. O jogador assim tende a se sentir mais à vontade para tentar soluções diferentes e, caso erre, tem como saber onde errou e pode tentar novamente. Isso é uma característica interessante dos jogos em geral: o fracasso não é o fim - e, portanto, o jogador não se sente punido por falhar, mas é incentivado a continuar tentando, conforme explica McGonigal (2011) quando fala sobre o feedback positivo do fracasso.

3.3.4 - Estética

A estética é um elemento que engloba os sentidos. Para jogos eletrônicos em geral, os mais importantes do ponto de vista projetual são a visão e audição; ou seja, artes, interface de usuário, efeitos sonoros e música. Vale ressaltar que também é possível levar em consideração, dentro das decisões do desenvolvimento, sentidos mais "incomuns" como o tato e o olfato - porém, este é um jogo mais tradicional nesse aspecto e por isso não englobará tais sentidos. Para este jogo, as decisões estéticas relacionaram-se exclusivamente à parte visual para se adequar ao conjunto de habilidades do desenvolvedor, capaz de projetar, programar e criar alguns *sprites* em *pixel art*, mas não capaz de criar música e efeitos sonoros para o jogo por conta própria.

Figura 3.2 - Arte conceitual de um personagem.

Fonte: Desenho de Luís Filipe Oliveira.

Algumas artes conceituais como a apresentada na figura 3.2 foram criadas para definir a aparência dos personagens, objetos e outros elementos gráficos do jogo. As artes conceituais não necessariamente representam a forma final, mas servem como um guia, o que ajuda a manter a coerência visual do jogo.

Figura 3.3 - Arte conceitual de um personagem em *pixel art*.

Fonte: Desenho de Luís Filipe Oliveira.

Visualmente, foi decidido que o estilo artístico a ser seguido seria o *pixel art*, que consiste na criação de imagens com poucos *pixels* e todos eles bastante visíveis, tendo como origem as limitações gráficas dos consoles dos anos 80 e 90. A escala dos elementos pode variar, sendo bastante comum o uso de números quadrados como 16x16 *pixels*, 32x32 *pixels* ou 64x64 *pixels*, podendo também possuir uma limitação de paleta de cores. O tamanho base escolhido para ser usado na criação de todos os

elementos do jogo foi o de 32x32 *pixels*, que dá margem para detalhes o suficiente para deixar o personagem reconhecível e ao mesmo tempo facilita o processo de animação do *sprite*.

As animações devem seguir um estilo cartunizado, o que dá um ar mais leve e descontraído para o jogo. As animações também devem conter uma quantidade relativamente baixa de frames, adicionando apenas o necessário para demonstrar as ações desejadas.

Para manter a coerência visual com os demais elementos do jogo, a interface de usuário foi criada pensando no *pixel art*, contando também com um estilo mais "quadrado" como os dos sistemas antigos e fonte estilo *pixel font*.

3.3.5 - Tecnologia

Este tópico trata principalmente da escolha das ferramentas que foram utilizadas no projeto. No desenvolvimento de jogos, existem várias áreas envolvidas; portanto, várias ferramentas são utilizadas durante todo o processo de desenvolvimento, como editores de texto para programação, editores de imagem para ilustração e animação, programas de modelagem 3D, *game engines*, entre outros tipos de *software*.

Para este projeto foram escolhidas as seguintes ferramentas: *Unity* como *game engine*, *Visual Studio* como editor de texto e Aseprite como editor de imagens. A *Unity* é uma *game engine* gratuita, de capacidade profissional, que atende hoje a múltiplos mercados como o de jogos, aplicativos, cinema e construção, tem como linguagem de programação o C# e capacidade para exportar para múltiplas plataformas como PC, Mac, Linux, mobile e diversos consoles. O Aseprite é um editor de imagens especializado em *pixel art*.

Para esta versão, foi considerada uma versão para apenas PC, podendo futuramente ser exportado para outras plataformas como web, mobile, Mac e Linux.

3.4 - Identificação das necessidades

Durante o desenvolvimento das mecânicas do jogo, foi identificado que para cumprir com a visão do projeto e entregar a experiência prevista, seriam necessárias algumas ferramentas além das que já eram oferecidas pela *Unity*. Somente a *game engine* não seria capaz de abranger toda a criação do jogo.

3.4.1 - A inteligência dos anões

Durante as iterações, foi identificada a necessidade dos anões funcionarem de forma autônoma, como é comum em jogos de estratégia em que o jogador não controla o personagem diretamente, mas sim comanda o jogo por meio de interfaces de usuário. Para criar essa independência, o comportamento dos personagens foi definido através de uma máquina de estados, representada a seguir:

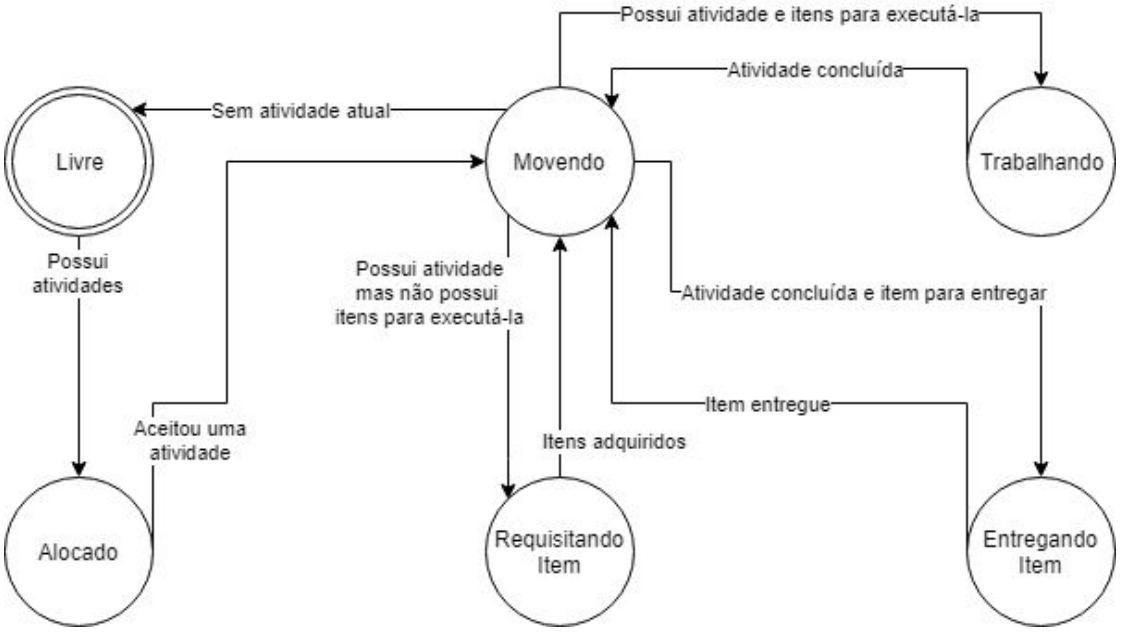

Figura 3.3 - Representação da máquina de estados.

Fonte: Criada por Luís Filipe Oliveira.

Neste primeiro momento, os estados representados na imagem são suficientes, podendo ser posteriormente ampliados de acordo com as necessidades que possam aparecer. O comportamento dos personagens pode ser representado também por:

- Livre:
    - O personagem se mantém neste estado até que possua pelo menos uma atividade.
- Alocado:
    - O personagem toma a primeira atividade dentre as disponíveis como atividade atual.
- Movendo:
    - O personagem decide para onde deve ir.
        - O personagem deve ir para o estoque requisitar materiais caso não possua o necessário para executar sua atividade.
        - O personagem deve ir para o estoque entregar itens caso já tenha terminado de produzir o item de sua atividade.
        - O personagem deve ir para a estação de trabalho adequada caso já possua os materiais necessários para execução de sua atividade.
        - O personagem deve voltar para o saguão como disponível caso já não possua uma atividade atual.
- Requisitando Itens:
    - O personagem faz uma requisição de itens ao estoque.
    - O personagem termina a requisição quando recebe todos os itens necessários para execução de sua atividade.

- Trabalhando:
    - O personagem executa a atividade.
    - O personagem termina o trabalho quando os materiais forem transformados no produto final de sua atividade.
- Entregando Itens:
    - O personagem entrega o produto de sua atividade ao estoque.
    - O personagem conclui a atividade atual, passando a não ter outra.

Com base nessa representação, foi definido que o jogo deve contar com um sistema baseado em máquinas de estados para controlar o comportamento dos personagens.

3.4.2 - Movimentando os anões

Outra necessidade identificada durante as iterações foi a de navegação dos personagens pelo mapa. O jogo possui uma visão *Top-Down*, ou seja, a câmera é posicionada acima da área do jogo. Este ângulo permite uma visão geral do ambiente de jogo e da área de movimentação para o personagem. De forma complementar, o mapa foi construído na forma de *tilemap* (em tradução livre, "mapa de azulejos"), que permite o uso de imagens recortadas em blocos e ordenadas em uma grade.

Por já ter à disposição um mapa em forma de grade, a movimentação do personagem deve implementar um algoritmo de *pathfinding* (em tradução livre, "encontrar o caminho") que seja compatível com a grade e que funcione bem para múltiplos agentes. Detalhes sobre a implementação do sistema de movimento e *pathfinding* estão no próximo capítulo.

4 - Ferramentas

4.1 - *Unity* e C#

A *Unity* foi a *game engine* escolhida para a implementação deste jogo. Trata-se de uma *game engine* gratuita (para desenvolvedores que tenham um faturamento de até cem mil dólares) com um vasto arsenal de ferramentas, serviços e recursos para auxiliar o desenvolvedor na produção de seu jogo.

A *Unity* usa como linguagem de programação o C#, uma linguagem orientada a objetos criada pela *Microsoft* e que faz parte da plataforma *.NET*. Além disso, possui uma ferramenta para fazer a conexão entre seu editor e o *Visual Studio*, tornando a ligação entre *engine* e editor de textos fluida com suporte às ferramentas de *debug* e autocompletar presentes no *Visual Studio* para classes criadas na *Unity*.

A Imagem 4.1 apresenta a interface do editor da *Unity*, onde é possível ver alguns dos elementos mais importantes da ferramenta. A aba *Scene* é onde o desenvolvedor monta o jogo, onde são montados e exibidos os elementos gráficos que fazem parte do jogo. Na aba *Hierarchy* se encontram todos os elementos que fazem parte da cena ativa. Na aba *Project*, são apresentados todos os *assets* que fazem parte do jogo como um todo, como cenas, *scripts*, imagens entre outros tipos de arquivos. A aba *Inspector* é responsável por exibir as informações do *asset* selecionado, seja ele parte do projeto ou de uma cena, permitindo ao desenvolvedor alterar detalhes e configurações de alguns deles como os *prefabs* e *scriptable objects*.

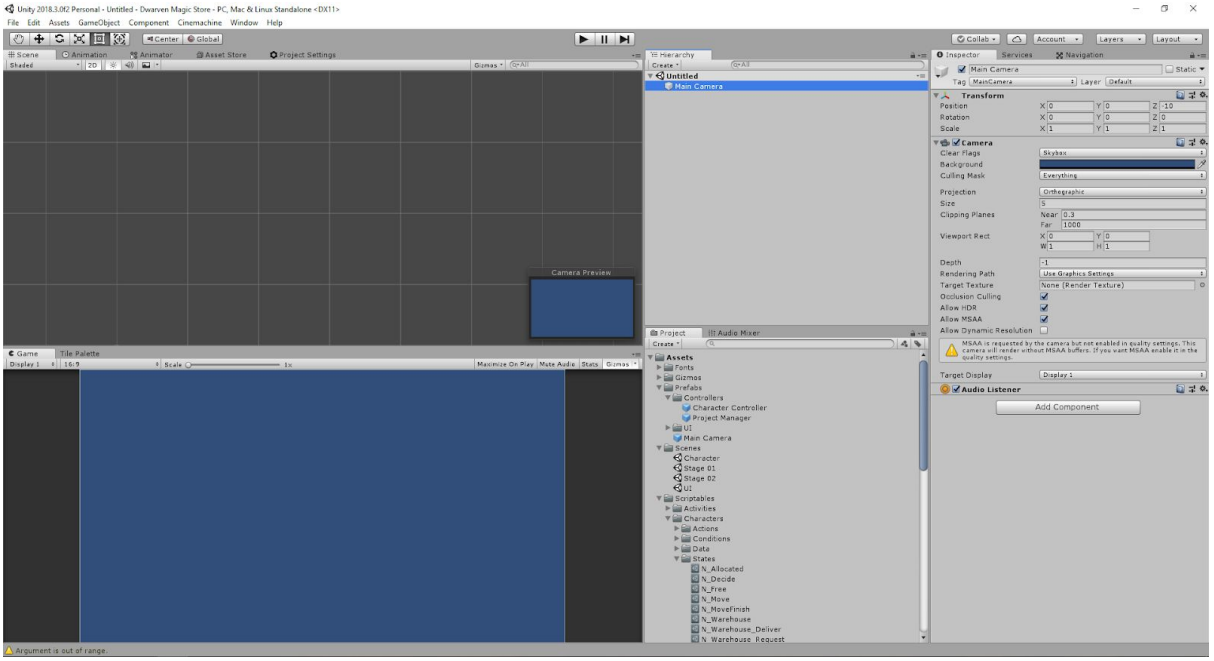

Figura 4.1 - Interface do editor da *Unity*.

Fonte: *Printscreen* da *game engine Unity*.

A *Unity* usa uma estrutura baseada em componentes. Nessa estrutura, o programador cria *scripts* que herdam de uma classe chamada *MonoBehaviour*, o que possibilita que eles sejam anexados a objetos de jogo diretamente pelo editor por meio do *Inspector* e ter seus valores configurados sem a necessidade de explicitá-los em código. Os *MonoBehaviour*s possuem um conjunto de métodos que são chamados em momentos especiais na execução de um jogo, sendo os principais o *Start*, que é chamado quando a cena se inicializa e o *Update*, que é chamado a cada *frame* enquanto o objeto existir na execução do jogo. Um *MonoBehaviour* só existe durante a execução da cena em que foi instanciado, portanto, ele é removido pelo *garbage collector* quando a cena deixa de existir.

Outra classe muito utilizada neste projeto é a *ScriptableObject*, que diferentemente do *MonoBehaviour*, pode existir independente de uma cena e por isso é muito utilizada para persistência de dados. Porém, ela pode ser utilizada também para escrever lógica, pois é uma classe como o *MonoBehaviour*, com a exceção que não

possui acesso aos métodos *Start* e *Update*. Outra vantagem dos scriptable objects é a possibilidade de salvar arquivos *.asset*, que podem ser utilizados para guardar configurações, dados e até lógica, podendo ser referenciados diretamente via *Inspector*, o que facilita o trabalho de *game designers*.

Para este projeto, a *Unity* foi utilizada para toda parte de implementação do código do jogo, configuração dos *assets* gráficos e geração de executável.

4.2 - *Aseprite*

O *Aseprite* é uma ferramenta paga de edição gráfica, com foco na criação de *sprites* e animações no estilo *pixel art*. Ela contém uma interface simples e algumas das funcionalidades encontradas nos principais editores gráficos. Uma vantagem do *Aseprite* é a possibilidade de visualizar as animações sem a necessidade de exportá-las e também a possibilidade de exportar as animações como *gifs* ou como *spritesheets*, um arquivo *.png* com transparência, que dispõe os quadros da animação de forma ordenada e padronizada, reduzindo o trabalho na hora de importar e configurar na *engine*.

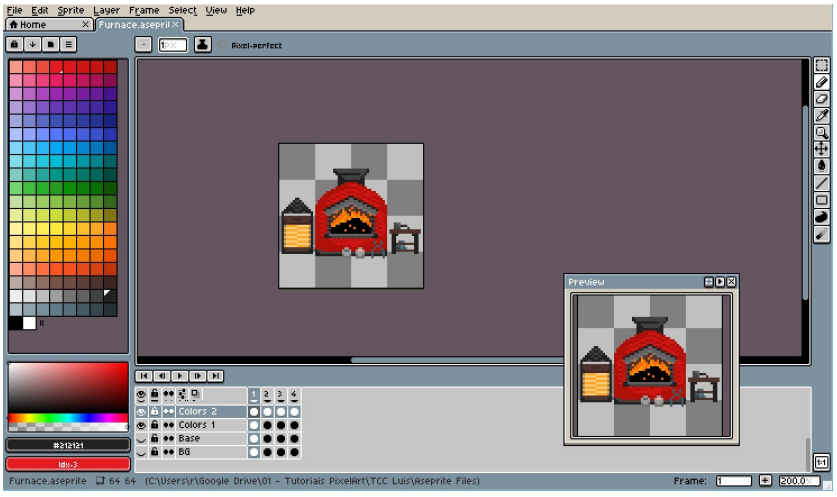

Figura 4.2 - Interface do *Aseprite*.

Fonte: *Printscreen* do *software Aseprite*..

Por essas facilidades, todos os *assets* gráficos foram criados utilizando o *Aseprite* e exportados na forma de *spritesheets*, o que para uso na *Unity* facilita o processo de importação e configuração dos mesmos.

4.3 - *Scriptable State Framework*

O *Scriptable State Framework* é um *framework* de máquina de estados altamente configurável criado pelo autor deste trabalho, baseado em *scriptable objects* para uso exclusivo com a ferramenta *Unity*, desenvolvido com base em uma apresentação da *Unity* sobre *scriptable objects* no evento *Unite* de 2017[9].

O framework tem como base cinco classes: *StateController*, *State*, *Transition*, *Action* e *Condition*. *StateController* é um *Monobehaviour*, portanto pode ser adicionado a um objeto da cena. Tem como objetivo controlar o fluxo de estados, transicionando entre eles quando necessário, mantendo sempre uma referência para seu estado atual. De uma forma geral, o fluxo de execução do *state controller* segue como apresentado na figura 4.3.

[9] Unite 2017 - Game Architecture with Scriptable Objects https://www.youtube.com/watch?v=raQ3iHhE_Kk

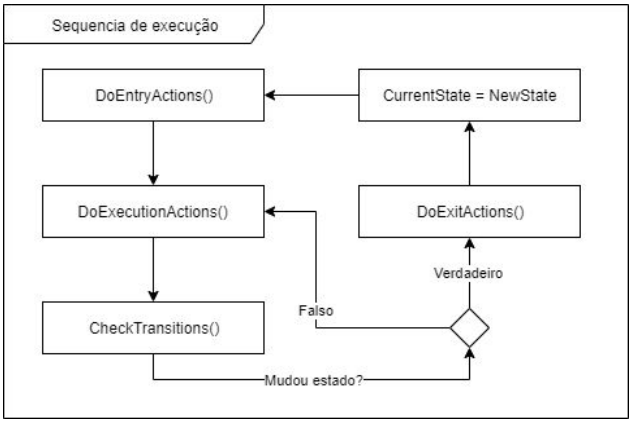

Figura 4.3 - Fluxo de execução do *state controller*.

Fonte: Criada por Luís Filipe Oliveira.

State é a representação de um estado. É um *ScriptableObject*, o que permite que seja possível a criação de arquivos configuráveis pelo editor. Desta forma, o próprio *game designer* poderia configurar estados para um determinado objeto, sem que para isso precisasse escrever qualquer código. Um *state* possui as instruções de funcionamento daquele estado, com ações de entrada, execução contínua e saída, além de uma lista de transições.

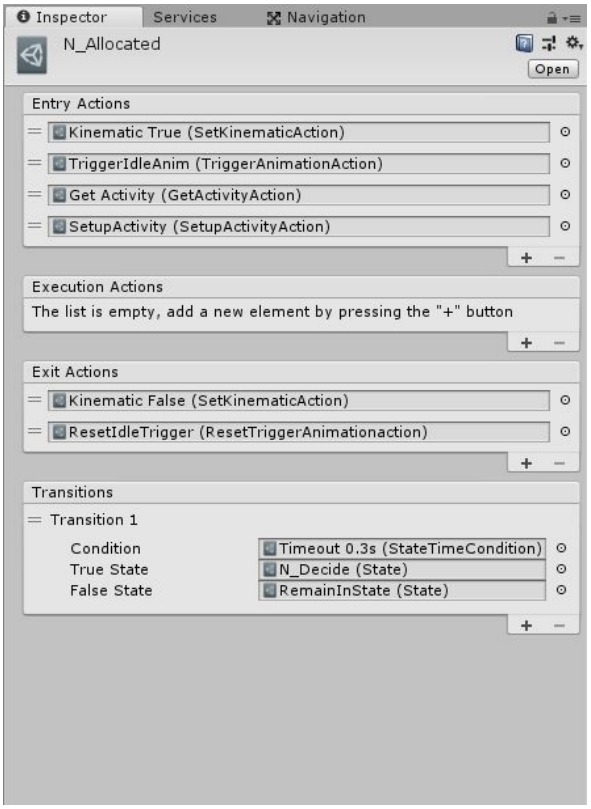

Figura 4.4 - Visualização de um *State*

Fonte: *Printscreen* da *game engine Unity*.

As ações de entrada são aquelas que executam apenas uma vez ao entrar no estado, enquanto as de saída executam apenas uma vez ao sair do estado. As ações de execução contínua são executadas repetidamente enquanto o estado for o estado ativo. As transições contêm todos os estados alcançáveis por aquele estado, bem como as condições para alcançá-los.

- *Transition* é a representação de uma transição. Possui uma condição a ser testada, um estado destino caso seja verdadeira e um estado destino caso seja falsa.
- *Action* é uma classe abstrata que herda *ScriptableObject*. Possui um único método abstrato *Act*(*StateController controller*). Dessa forma, as classes derivadas de *Action* criadas pelo programador podem acessar os outros

componentes ligados ao *StateController*, por exemplo física, controladores de animação ou colisores.

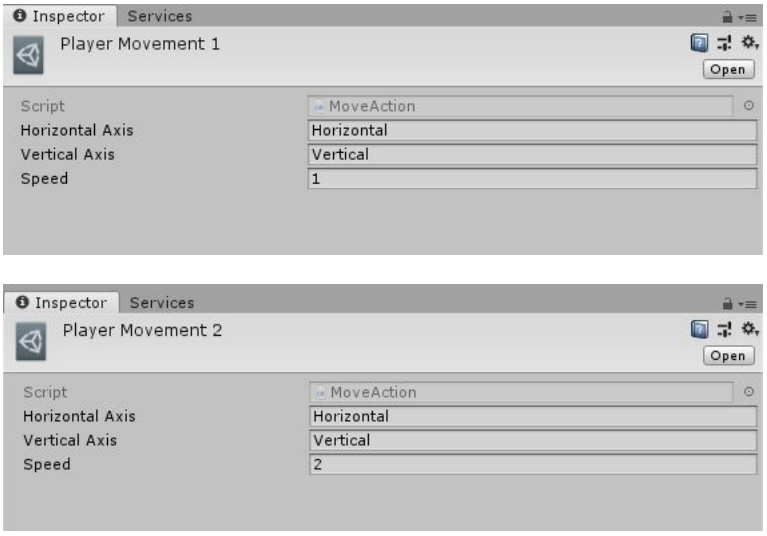

Figura 4.5 - Visualização de uma *Action*

Fonte: *Printscreen* da *game engine Unity*.

Condition, assim como *Action*, é uma classe abstrata que herda de *ScriptableObject*. Possui apenas um método abstrato *Verify*(*StateController controller*), que retorna um *bool* como resultado da verificação. O papel desta classe é verificar se as condições foram atendidas, retornando uma *booleana* para que a transição seja ativada ou não, podendo ser o apertar de um botão, a posição de um objeto na tela ou a colisão do personagem com o inimigo, por exemplo. Assim, como a *Action*, pode ser personalizada via *Inspector* pelo *designer*, sem a necessidade de alteração de códigos.

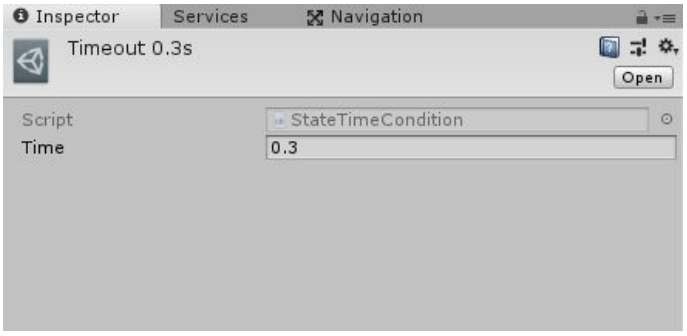

Figura 4.6 - Visualização de uma *Condition*

Fonte: *Printscreen* da *game engine Unity*.

4.4 - *Tilemap Navigator System*

O *Tilemap Navigator System* é uma ferramenta que adiciona um componente de navegação compatível com a ferramenta de *Tilemap* da *Unity*. Desenvolvido pelo autor deste trabalho, este sistema tem como base o algoritmo de *pathfinding* A* tendo o *Manhattan Distance* como heurística e um algoritmo auxiliar que permite o funcionamento para múltiplos agentes, com o mínimo de colisões entre si.

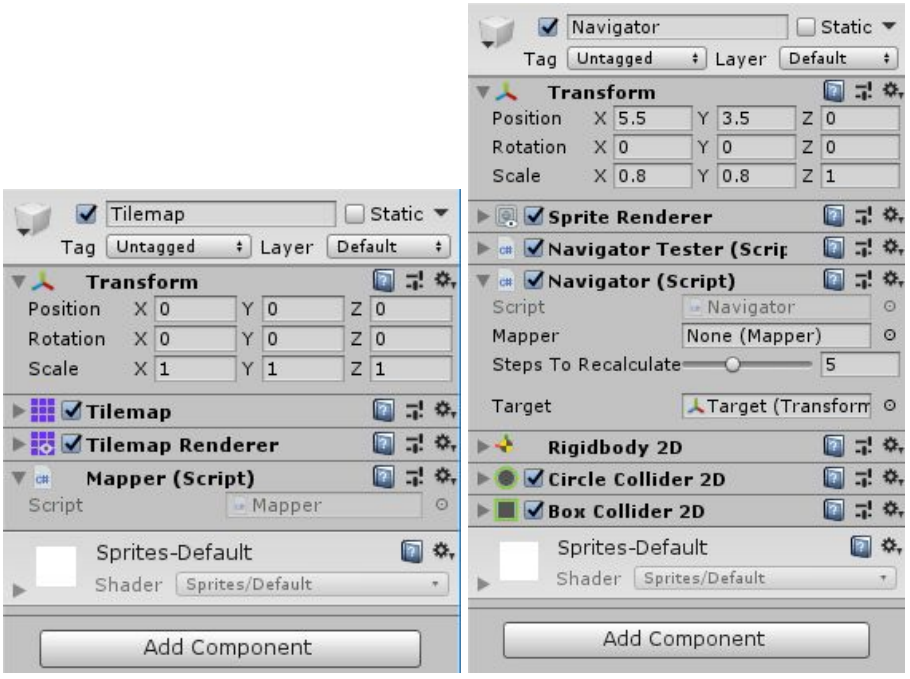

Figura 4.7 - Componentes do *Tilemap Navigator* em um *gameobject* na *Unity*

Fonte: *Printscreen* da *game engine Unity*.

As cinco classes que compõem este sistema são: *Mapper*, *Navigator*, *Obstacle*, *PathNode* e *PathTile*. *Mapper* é um *Monobehaviour* que deve ser adicionado ao

Tilemap que se deseja navegar sobre. O *Mapper* identifica os *Navigators* e *Obstacles* da cena, aplica o algoritmo de *pathfinding* e retorna o melhor caminho para aquele *navigator* chegar ao seu destino. O *PathTile* é uma extensão do *Tile* padrão da *Unity*, que contém uma variável *booleana* que indica se é passável ou não. O *PathNode* é a representação de um ponto na grade, ou no caso no *Tilemap*, somente com as informações necessárias para realizar a navegação. O *Navigator* é um *Monobehaviour* que adiciona funcionalidades de agente de navegação a um objeto do jogo, que trafega pelo mapa de acordo com o caminho trilhado pelo *Mapper*. O *Obstacle* é um *Monobehaviour* que representa um obstáculo no mapa, e diferentemente dos *PathTiles*, pode ser alterado em tempo de execução.

5 - Implementação: Código e Assets

5.1 - Uma visão geral

Seguindo o que foi definido no capítulo três com relação ao *design* do jogo e utilizando as ferramentas apresentadas no capítulo quatro, este capítulo tem como objetivo a apresentação da implementação de um protótipo digital do jogo proposto neste trabalho.

Em um primeiro momento, inspirado pelas cartas criadas para o protótipo de papel, foi elaborado um diagrama apresentando as interações e relacionamentos entre os elementos do jogo e suas relações, conforme pode ser visto na Figura 5.1.

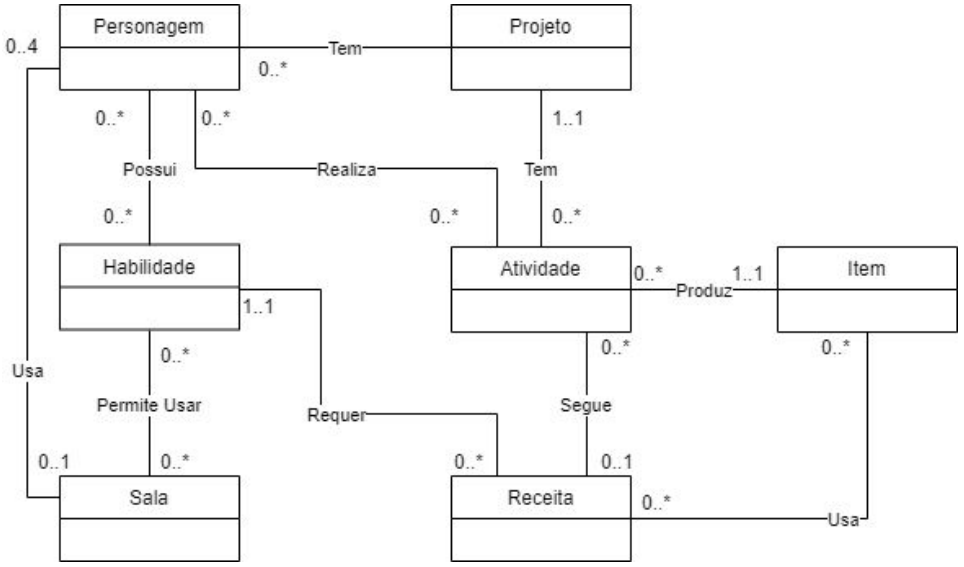

Figura 5.1 - Diagrama das classes baseado nas cartas do Protótipo 2

Fonte: Criada por Luís Filipe Oliveira.

Como pode ser notado, o diagrama da Figura 5.1 apresenta uma ideia do que podem vir a ser classes do modelo, que têm como principal função a representação das informações do jogo. Dentro da *Unity*, estas classes podem ser implementadas como

ScriptableObject, o que permite a criação arquivos *asset* configuráveis pelo *Inspector* e independentes de cena.

5.2 - O comportamento dos anões

Durante o *design* do jogo, foi definido que os personagens do jogo deveriam agir de forma independente e que este comportamento poderia ser representado como uma máquina de estados. Para transpor esta representação para dentro da *Unity*, foi utilizado o *Scriptable State Framework*, em conjunto com o *Tilemap Navigator* para resolver a movimentação.

Primeiramente, foi criado um *GameObject* do tipo *Sprite* para representar o anão. Neste primeiro momento a forma não tinha importância, portanto foi usado um quadrado para representá-lo. Neste objeto foram adicionados os componentes *StateController* e *Navigator*. Em seguida, foram criados os arquivos de estado, dando a eles nomes conforme o que foi definido anteriormente, Livre, Alocado, Movendo, Trabalhando, Requisitando Item e Entregando Item.

Para facilitar a leitura dos estados, foi criado um modelo para representá-los, que corresponde aos elementos configuráveis da classe *State*: Ações de entrada, Ações de Execução, Ações de Saída e Transições, conforme pode ser visto na Figura 5.2.

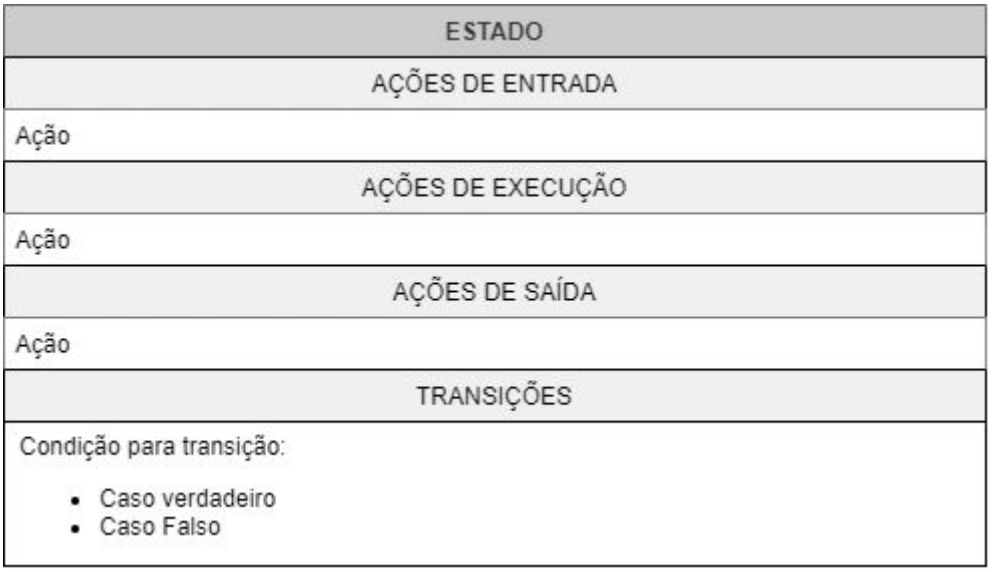

Figura 5.2 - Modelo de Estado

Fonte: Criada por Luís Filipe Oliveira.

Este modelo tem como objetivo facilitar o entendimento e visualização do fluxo de estados e auxiliar na identificação das ações e condições a serem desenvolvidas. Vale mencionar que, por definição e buscando semelhança entre o modelo e o *framework*, o termo "Mantém o mesmo" e o uso do mesmo estado possuem significados diferentes. "Mantém o mesmo" significa que não há transição entre estados, o estado atual continua rodando as ações de execução e testando as condições de transição, enquanto que "usar o mesmo estado" significa que há uma transição do estado atual para um novo estado que por acaso é o mesmo, o que implica na execução das ações de saída do estado atual e ações de entrada do novo estado.

O estado "Livre" (Figura 5.3) representa o momento em que o personagem não possui nenhuma atividade. Desta forma, ele não executa nenhuma ação, apenas verificando se alguma ação foi atribuída a ele. Tendo ao menos uma atividade, o personagem então passa para o estado "Alocado" (Figura 5.4), em que pega a primeira atividade disponível para ele ao entrar no estado e em seguida passa para o estado "Movendo" (Figura 5.5). Neste estado o personagem se move até um dos destinos possíveis: "Requisitando itens" (Figura 5.6) caso tenha uma atividade para realizar mas não tenha os materiais para a mesma, "Trabalhando" (Figura 5.7) caso já

tenha os materiais para realizar sua atividade, “Entregando Itens” (Figura 5.8) caso já tenha concluído sua atividade ou volta para “Livre” caso não possua mais uma atividade atual.

Em “Requisitando Itens” o personagem se mantém pedindo os itens para realizar sua atividade até que possua todos os itens necessários para realizar sua atividade e assim voltar a se mover. Em “Trabalhando” o personagem executa a sua atividade, transformando os itens que possui no item pedido e voltando a se mover ao terminar. Em “Entregando Itens”, o personagem realiza a entrega do resultado de seu trabalho e finalizando assim sua atividade, ficando assim livre para pegar uma nova atividade.

LIVRE
AÇÕES DE ENTRADA
Nenhuma
AÇÕES DE EXECUÇÃO
Nenhuma
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Possui atividades: • Caso verdadeiro: Alocado • Caso Falso: Mantém o mesmo

Figura 5.3 - Estado Livre

Fonte: Criada por Luís Filipe Oliveira.

ALOCADO
AÇÕES DE ENTRADA
Personagem toma a primeira atividade dentre as disponíveis
AÇÕES DE EXECUÇÃO
Nenhuma
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Possui atividades: • Caso verdadeiro: Alocado • Caso Falso: Mantém o mesmo

Figura 5.4 - Estado Alocado

Fonte: Criada por Luís Filipe Oliveira.

MOVENDO
AÇÕES DE ENTRADA
Nenhuma
AÇÕES DE EXECUÇÃO
Personagem se move até o destino
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Possui atividade e não possui itens para executá-la: • Caso verdadeiro: Requisitando Itens • Caso Falso: Mantém o mesmo
Possui atividade e itens para executá-la: • Caso verdadeiro: Trabalhando • Caso Falso: Mantém o mesmo
Possui atividade concluída e item para entregar: • Caso verdadeiro: Entregando Itens • Caso Falso: Mantém o mesmo
Sem atividade atual: • Caso verdadeiro: Livre • Caso Falso: Mantém o mesmo

Figura 5.5 - Estado Movendo

Fonte: Criada por Luís Filipe Oliveira.

REQUISITANDO ITENS
AÇÕES DE ENTRADA
Nenhuma
AÇÕES DE EXECUÇÃO
Personagem requisita itens ao estoque
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Possui itens necessários: • Caso verdadeiro: Movendo • Caso Falso: Mantém o mesmo

Figura 5.6 - Estado Requisitando Itens

Fonte: Criada por Luís Filipe Oliveira.

TRABALHANDO
AÇÕES DE ENTRADA
Nenhuma
AÇÕES DE EXECUÇÃO
Personagem executa transformação do material
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Transformação concluída: • Caso verdadeiro: Movendo • Caso Falso: Mantém o mesmo

Figura 5.7 - Estado Trabalhando

Fonte: Criada por Luís Filipe Oliveira.

ENTREGANDO ITENS
AÇÕES DE ENTRADA
Personagem entrega item que representa produto final da atividade
AÇÕES DE EXECUÇÃO
Nenhuma
AÇÕES DE SAÍDA
Personagem finaliza atividade
TRANSIÇÕES
Item entregue: • Caso verdadeiro: Movendo • Caso Falso: Mantém o mesmo

Figura 5.8 - Estado Entregando Itens

Fonte: Criada por Luís Filipe Oliveira.

Com base neste modelo inicial, baseado no diagrama com seis estados e no protótipo de cartas, foi possível identificar a sequência de ações de um personagem e prever possíveis falhas na lógica. Em uma leitura inicial, foi identificada a falta de alguns elementos de decisão que são apresentados a seguir:

- **O personagem não verifica para onde deve ir antes de começar a se mover**: Uma possível solução foi a adição de um novo estado que antecede o estado Movendo, o "Decidindo" (Figura 5.11), onde o personagem decide o destino de sua movimentação. O estado Movendo passa a ser responsável apenas por realizar o movimento como mostra a Figura 5.12.
- **O personagem não verifica se chegou ao seu destino**: Complementar à solução anterior, a proposta foi adicionar um novo estado posterior ao estado

Movendo, o “Fim Movimento (Figura 5.13) que verifica onde o personagem parou, mudando o estado de acordo.

- **A chegada ao estoque é ambígua**: Adição de um estado que antecede os estados Requisitando Itens e Entregando Itens, o estado “Em Estoque” (Figura 5.14). Este estado verifica a existência ou não de itens para entrega, alternando entre os estados Requisitando Itens e Entregando Itens conforme necessário.

A adição dos novos estados altera o fluxo apresentado anteriormente, conforme pode ser visto na Figura 5.9. Além disso, foi feita uma atualização do modelo de estados para refletir as mudanças e então dar seguimento à implementação das ações e condições como pode ser visto na Figura 5.10, que representa as mudanças no estado “Alocado”.

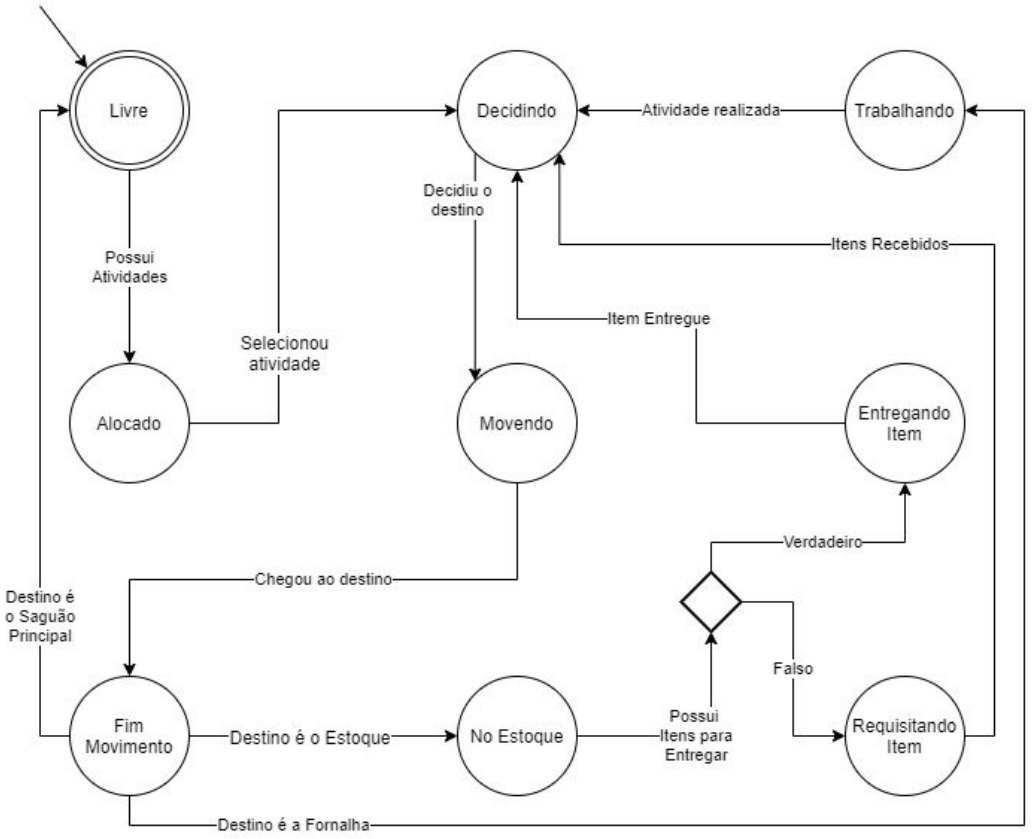

Figura 5.9 - Comportamento dos anões atualizado.

Fonte: Criada por Luís Filipe Oliveira.

ALOCADO v2
AÇÕES DE ENTRADA
Personagem toma a primeira atividade dentre as disponíveis
AÇÕES DE EXECUÇÃO
Nenhuma
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Possui atividades: • Caso verdadeiro: Decidindo • Caso Falso: Mantém o mesmo

Figura 5.10 - Atualização do Estado Alocado

Fonte: Criada por Luís Filipe Oliveira.

DECIDINDO
AÇÕES DE ENTRADA
Nenhuma
AÇÕES DE EXECUÇÃO
Personagem decide para onde ir: - Se possuir atividade e itens > fornalha - Se possuir atividade e não possuir itens ou terminou a atividade > estoque - Se não possuir atividade > saguão
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Possui um destino: • Caso verdadeiro: Movendo • Caso Falso: Mantém o mesmo

Figura 5.11 - Estado Decidindo

Fonte: Criada por Luís Filipe Oliveira.

MOVENDO v2
AÇÕES DE ENTRADA
Nenhuma
AÇÕES DE EXECUÇÃO
Personagem se move até o destino
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Chegou ao destino: • Caso verdadeiro: Fim Movimento • Caso Falso: Mantém o mesmo

Figura 5.12 - Atualização do Estado Movendo

Fonte: Criada por Luís Filipe Oliveira.

FIM MOVIMENTO
AÇÕES DE ENTRADA
Nenhuma
AÇÕES DE EXECUÇÃO
Nenhuma
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Sem atividade atual: • Caso verdadeiro: Livre • Caso Falso: Mantém o mesmo
Possui atividade e itens para executá-la: • Caso verdadeiro: Trabalhando • Caso Falso: Mantém o mesmo
Possui atividade e não possui itens ou possui atividade concluída e item para entregar: • Caso verdadeiro: Em Estoque • Caso Falso: Mantém o mesmo

Figura 5.13 - Estado Fim Movimento

Fonte: Criada por Luís Filipe Oliveira.

NO ESTOQUE
AÇÕES DE ENTRADA
Nenhuma
AÇÕES DE EXECUÇÃO
Nenhuma
AÇÕES DE SAÍDA
Nenhuma
TRANSIÇÕES
Possui atividade e não possui itens: • Caso verdadeiro: Requisitando Itens • Caso Falso: Mantém o mesmo
Possui atividade concluída e item para entregar: • Caso verdadeiro: Entregando Itens • Caso Falso: Mantém o mesmo

Figura 5.14 - Estado No Estoque

Fonte: Criada por Luís Filipe Oliveira.

Tendo definido os estados do personagem, foi possível criar as ações necessárias para seu funcionamento utilizando as descrições das ações presentes no modelo de estados como base para isso. Essas ações são classes que derivam de *Action* do *Scriptable State Framework* e têm sua lógica implementada dentro do método *Act*.

- **Selecionar Atividade**: Seleciona uma atividade da lista de atividades do projeto e adiciona como atividade atual do personagem.
- **Decidir Destino**: Verifica as seguintes condições:
    - Se possuir uma atividade atual e os itens para executá-la, então Destino recebe Fornalha.

    - Se possuir uma atividade e não possuir itens para executá-la ou se atividade já foi realizada, então Destino recebe Estoque.
    - Se não possuir uma atividade atual, então Destino recebe Saguão.

- **Mover**: Chama o método de movimentação do componente de navegação (*Navigator*) do personagem.
- **Requisitar Itens**: Interage com o estoque para solicitar os itens necessários para executar a tarefa. O personagem não recebe os itens até que todos estejam disponíveis em estoque.
- **Entregar Itens**: Interage com o estoque e transfere os itens que possui para o mesmo.
- **Finalizar Atividade**: Marca a atividade como completa e a atividade atual passa a ser vazia.
- **Realizar Trabalho**: Interage, pelo tempo necessário, com a estação de trabalho para transformar os itens no produto final da atividade.

Então, foram criadas as condições, baseadas no texto das transações do modelo de estados. Estas condições são classes derivadas de *Condition* do *Scriptable State Framework* e tem sua lógica implementada do método *Verify*.

- **Possui Atividades**: Verifica se existem atividades na lista de atividades do projeto que têm o personagem como responsável.
- **Selecionou Atividade**: Verifica se possui uma atividade atual.
- **Decidiu Destino**: Verifica se possui um destino definido.
- **Possui itens para entregar**: Verifica se existem itens em posse do personagem.

- **Itens Recebidos**: Verifica se os itens solicitados estão em posse do personagem.
- **Itens Entregues**: Verifica se o personagem não está mais carregando itens.
- **Atividade Realizada**: Verifica se o personagem cumpriu o tempo necessário para execução da atividade.

As ações e condições que representam os comportamentos dos anões poderiam ser criadas como *assets* para o jogo e adicionadas aos seus respectivos estados. Esta configuração pôde ser feita inteiramente pelo *Inspector*, sem a necessidade de criação de mais códigos.

Para que as ações e condições funcionassem, as informações do personagem deveriam estar em algum lugar. Para isso, foram necessárias duas novas classes, uma classe que representasse as informações fixas do personagem, enquanto outra deveria possuir as informações que poderiam ser alteradas em tempo de execução. Para a primeira, foi criada a *CharacterData* que é um *ScriptableObject* e armazena as informações como nome, ícone, animações e habilidades. A segunda foi a *CharacterComponent* que contém uma referência para o *CharacterData***,** de onde tira as informações do personagem, uma referência para sua atividade atual, o destino de sua movimentação, uma lista de itens que carrega consigo e as informações sobre o estado de execução de sua atividade atual. O resultado pode ser observado na Figura 5.15.

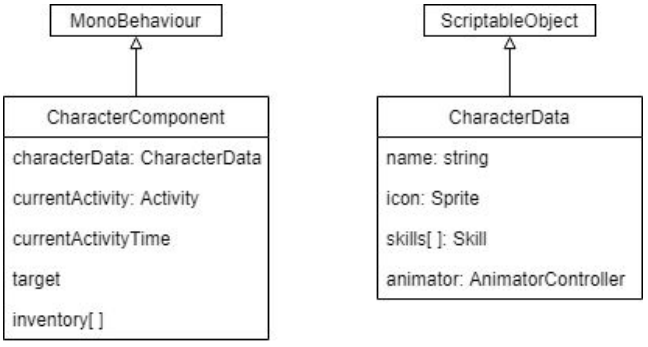

Figura 5.15 - Representação das classes de *Character*

Fonte: Criada por Luís Filipe Oliveira.

Foram criadas também algumas ações para adicionar os elementos da estética do jogo:

- **Ativar Animação**: Dado um nome de animação, busca no componente *Animator* do personagem uma animação de mesmo nome e a ativa.
- **Resetar Animação**: Dado um nome de animação, busca no componente *Animator* do personagem uma animação de mesmo nome e reseta seu estado.

5.3 - As habilidades dos anões

Para criar as habilidades dos anões de forma que pudessem ser facilmente manipuladas pelo editor da *Unity*, sem a necessidade de escrever código adicional, foram criadas duas classes: *Skill* e *SkillType*. A classe *SkillType* é um *ScriptableObject* e possui apenas um nome como atributo, pois desta forma, através do editor seria possível criar objetos do tipo *SkillType* e usá-los para comparação, quando necessário. Já a classe *Skill* possui um *SkillType* e um valor, conforme mostra a Figura 5.16. Como foi apresentado no tópico anterior, um personagem possui uma lista de habilidades e com esta abordagem, seria possível de uma forma mais dinâmica adicionar novas habilidades e remover as que não mais estivessem sendo usadas, pois estas foram construídas de maneira a gerar baixa dependência entre elas e outras classes que possam vir a consumir suas informações.

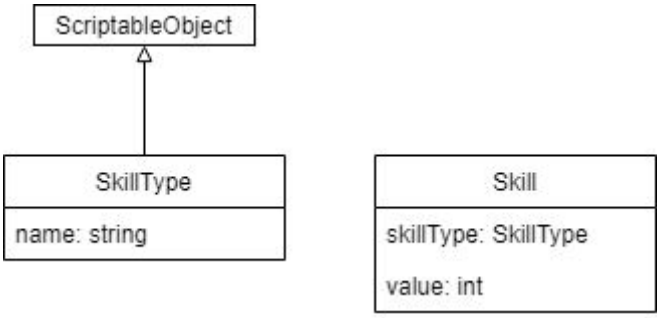

Figura 5.16 - Representação das classes de *Skills*.

Fonte: Criada por Luís Filipe Oliveira.

5.4 - Itens e Receitas

Uma parte central do jogo é a transformação de itens. Para esta transformação acontecer, o personagem deveria seguir uma receita, que é a aplicação de uma habilidade sobre um conjunto de itens. Com base nisso, foram criadas as seguintes classes: *ItemData*, *ItemStack* e *ItemRecipe*, conforme pode ser observado na Figura 5.17. *ItemData* é um *ScriptableObject* que representa um tipo de item no jogo, contendo um nome, descrição e um ícone. *ItemStack* como o nome propõe funciona como uma coleção de itens, contendo um *ItemData* e sua quantidade, sendo utilizado quando se faz necessário contar itens, como por exemplo em um inventário ou na própria receita. *ItemRecipe* é a representação das informações para a transformação de um item, contendo um item como resultado, a habilidade necessária e uma lista de materiais.

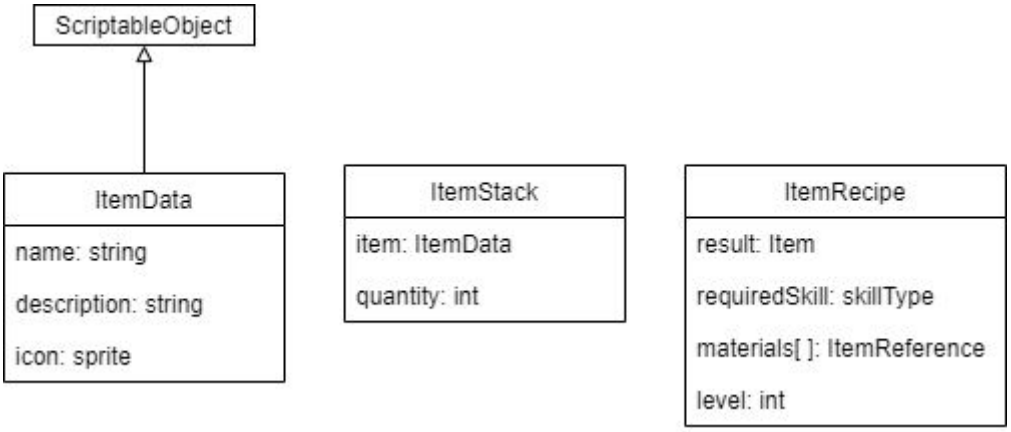

Figura 5.17 - Representação das classes de Item e *Recipe*.

Fonte: Criada por Luís Filipe Oliveira.

5.5 - Os anões realizam atividades

As atividades são uma parte importante do fluxo do jogo. Uma boa parte da execução do jogo é composta de anões realizando atividades. Para este protótipo, o jogador é capaz de criar e atribuir um responsável para esta atividade. Para isso foi criada a classe *Activity*, conforme apresentado na Figura 5.18. A classe *Activity* é composta por um *id*, um item que se deseja criar, uma receita através da qual o personagem deve criar o item, um personagem responsável pela tarefa e seu progresso.

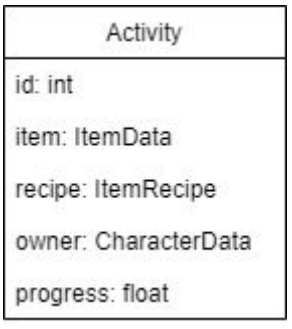

Figura 5.18 - Representação da classe *Activity*.

Fonte: Criada por Luís Filipe Oliveira.

5.6 - O projeto no jogo

Executar o projeto é o objetivo do jogo. Sendo assim, a representação de projeto no jogo foi pensada de forma a facilitar a criação de novos conteúdos, no caso, novos projetos. Pensando nisso, foram criadas as classes *ProjectData* e *ProjectComponent*, conforme ilustra a Figura 5.19. A classe *ProjectData* contém as informações gerais do projeto, como uma configuração, sendo estas o nome do projeto, descrição, personagens disponíveis e uma lista de atividades inicial. *ProjectComponent* é um *Monobehaviour*, e funciona como a instância do projeto. Ela contém uma referência a um *ProjectData*, o time, uma lista de atividades e o tempo de execução do projeto.

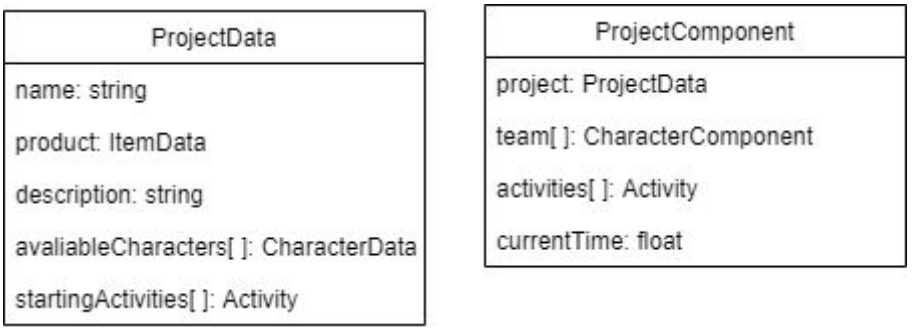

Figura 5.19 - Representação das classes *Project*.

Fonte: Criada por Luís Filipe Oliveira.

5.7 - Criando as artes

A parte visual do jogo começou a ser desenvolvida após ser decidido que a narrativa se passaria em um mundo fantástico e envolveria anões trabalhando em uma forja, pois era importante que o jogador conseguisse distinguir com facilidade os elementos presentes no jogo. Requer muito menos esforço notar a diferença entre um escudo e

uma espada do que notar a diferença, por exemplo, entre dois pedaços de papel representando documentos diferentes, caso o jogo se passasse em um ambiente de "mundo real".

Sendo assim, cada elemento do jogo precisava ser desenhado de forma que fosse facilmente identificável, para que o jogador pudesse prestar mais atenção no processo de gerência de projetos ao invés de gastar suas energias tentando descobrir o significado de cada *asset* visual.

Foi adotado o estilo *pixel art* por sua simplicidade e praticidade: com poucos *pixels* é possível representar e animar todos os elementos necessários para o jogo, incluindo os anões, suas estações de trabalho, o cenário da forja e também a interface, por meio da qual o jogador comandaria todo o jogo.

5.7.1 - Os personagens

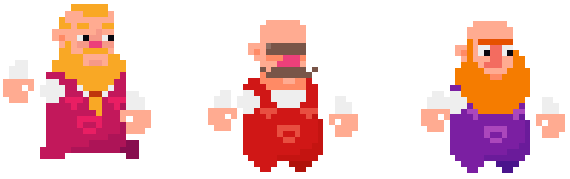

Figura 5.20 - Os personagens anões Gromp, Stork e Indi, respectivamente.

Fonte: Artes criadas por Luís Filipe Oliveira.

Inicialmente foram criados três personagens - Gromp, Stork e Indi. Eles têm em comum o fato de serem todos baixos, gordos e barbudos, pois é assim que anões normalmente são representados em histórias fantásticas. Os três vestem aventais de trabalho para condizer com a narrativa da forja. Embora os personagens criados sejam similares, atentou-se para que não fossem idênticos: seus cabelos e barbas mudam de formato, bem como suas cores. Dessa forma, o jogador pode

reconhecê-los e diferenciá-los entre si, o que é especialmente útil já que cada personagem possui habilidades diferentes.

Os três anões foram animados seguindo a técnica *frame by frame* (quadro a quadro), com a finalidade de deixar mais claro para o jogador que tipo de ação está sendo realizada por cada personagem, a cada momento. Suas animações respondem à máquina de estados e incluem as ações:

- **Coçando a barba:** esta animação representa o estado *idle*, quando os anões estão parados sem realizar nenhuma tarefa.
- **Caminhando:** esta animação indica que os anões estão se movendo entre as estações de trabalho.
- **Martelando:** esta animação indica que os anões estão transformando algum material na fornalha.

Figura 5.21 - *Spritesheets* das animações dos personagens coçando a barba, martelando e caminhando.

Fonte: Artes criadas por Luís Filipe Oliveira.

Futuramente, pretende-se incluir mais animações para deixar o jogo não somente mais divertido, mas também com um *feedback* visual mais claro para o jogador. Por enquanto, foi considerado que estas três animações seriam o suficiente para transmitir as informações necessárias ao jogador.

5.7.2 - O cenário

Visto que a câmera do jogo foi posicionada de forma *Top-Down*, ou seja, observando o cenário de cima, para o cenário foi necessário desenhar somente o chão e as estações de trabalho. O chão foi desenhado em forma de *tiles*, partes menores colocadas lado a lado para que sua repetição forme um desenho maior.

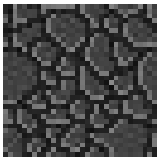

Figura 5.21 - Imagem do *tile* utilizado para construir o chão.

Fonte: Arte criada por Luís Filipe Oliveira.

As estações de trabalho necessárias para esta versão do jogo eram:

- **Quadro de avisos:** aqui, os anões "pegam" novas atividades. Na verdade, foi o jogador quem atribuiu atividades aos anões por meio da interface, mas o quadro de avisos serve para representar essa ação de maneira visual e narrativa.
- **Mesa de estoque:** nela ficam armazenados os materiais que os anões precisam utilizar para fabricar os objetos necessários. Seu desenho possui várias gemas preciosas, pedras, pedaços de ferro etc. colocados sobre e ao redor de uma bancada.
- **Fornalha:** é o local onde os anões criam os itens. É a única estação de trabalho que possui uma animação, pois possui fogo queimando dentro dela e o fole que alimenta o fogo funcionando magicamente.

Figura 5.22 - As estações de trabalho. Respectivamente: o quadro de avisos, a mesa de estoque e a fornalha.

Fonte: Artes criadas por Luís Filipe Oliveira.

5.7.3 - A interface

A aparência da interface do jogo foi criada também seguindo o *pixel art*, para que todo o jogo fique coerente visualmente. Todas as janelas, os botões, as barras de progresso e os ícones foram desenhados com cantos arredondados que, ao serem examinados de perto, revelam que também foram feitos *pixel* a *pixel*. Até mesmo a tipografia utilizada seguiu o estilo *pixel font*.

Figura 5.23 - A interface do jogo criada em *pixel art.*

Fonte: Artes criadas por Luís Filipe Oliveira.

Os ícones criados para este jogo são importantes, visto que o jogador controla tudo pela interface. Assim, por eles é possível ter uma visão mais próxima do rosto dos anões e também dos materiais utilizados / refinados ao longo da jogabilidade. Um detalhe a ser observado é que, no caso dos materiais, o único momento em que aparecem é na interface - os materiais não são mostrados e ficam "escondidos" na estação da mesa de estoque.

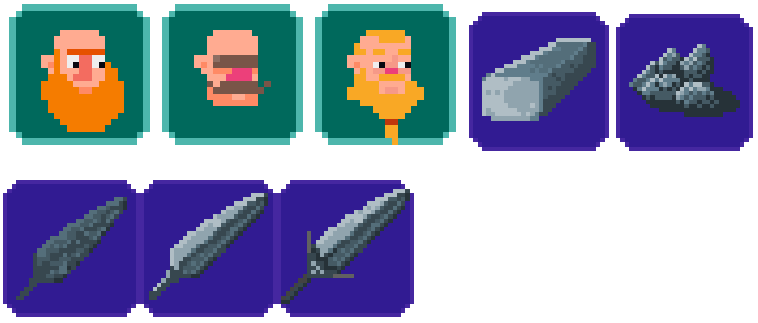

Figura 5.24 - Ícones representando os personagens e os materiais que fazem parte do jogo (*iron ingot, iron ore* e três estágios da fabricação de uma espada).

Fonte: Artes criadas por Luís Filipe Oliveira e João Gabriel Oliveira.

Figura 5.25 - O ícone do personagem inserido na interface.

Fonte: Artes criadas por Luís Filipe Oliveira.

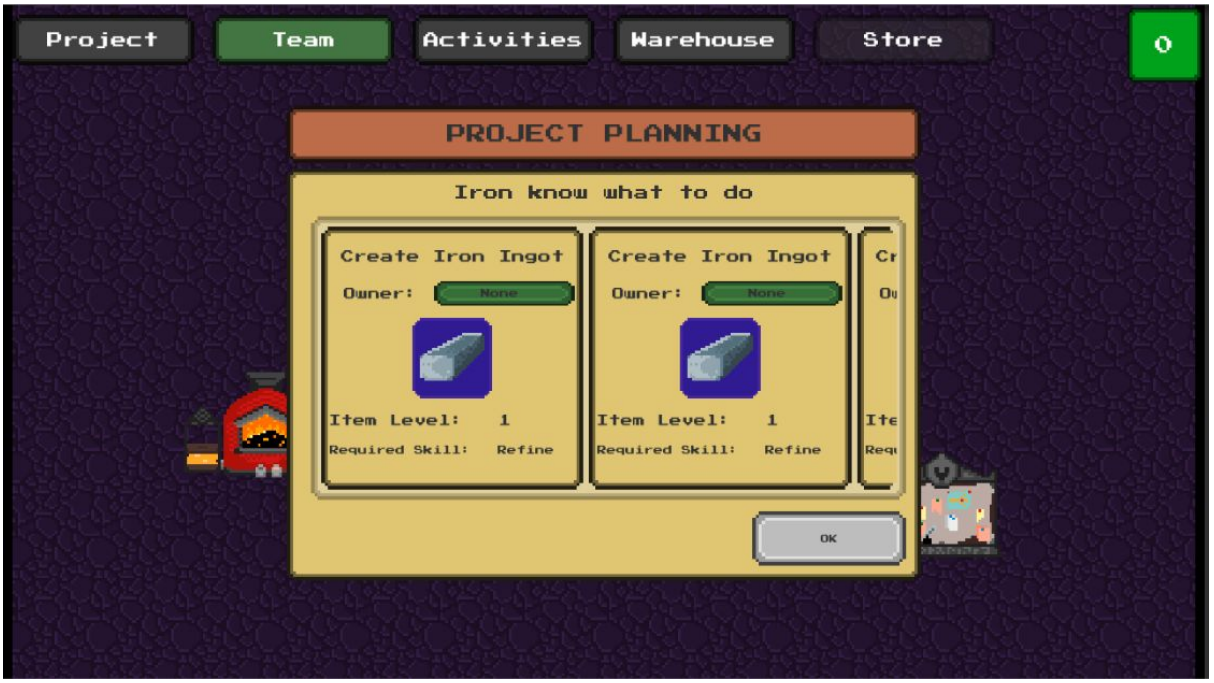

Figura 5.26 - O ícone dos materiais incluídos na interface (*iron ingot*).

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 5.27 - O ícone dos materiais incluídos na interface (*iron ore*).

Fonte: Artes criadas por Luís Filipe Oliveira.

Posteriormente as artes foram atualizadas, como pode ser visto no capítulo 6, na apresentação do manual.

6 - Manual

6.1 - Download e instalação

1. Acessar o link http://bit.ly/TestesDwarfTCC e baixar o arquivo zip ***Dwarven Forge beta 2019v1.1***
2. Descompactar em uma pasta da preferência do usuário.
3. Abrir a pasta e executar o arquivo ***Dwarven Forge*** conforme Figura 6.1.

Figura 6.1 - Conteúdo da pasta zipada.

Fonte: Luís Filipe Oliveira.

6.2 - Como usar:

1. A Figura 6.2 demonstra o estado inicial do jogo quando o aplicativo é aberto. Na área superior ficam os botões de acesso às janelas do jogo; no meio, a área do jogo, com as salas e personagens e por último, no canto inferior direito, uma label com a versão atual.
2. Clicar no botão ***Project*** abre a janela com as informações do projeto, conforme mostra a Figura 6.3. Nela encontram-se o objetivo do projeto, uma descrição do que deve ser feito, quantidade de atividades em cada estado e tamanho da equipe.

3. Clicar no botão ***Activities*** abre a janela de gerenciamento, conforme apresentado nas Figuras 6.4 e 6.7. Nela é possível criar atividades (Figura 6.5), deletá-las e atribuir responsáveis a elas (Figura 6.6).
4. Clicar no botão ***Recipes*** abre a janela de receitas, conforme Figura 6.8. Nela é possível navegar pelas receitas conhecidas, criar uma atividade a partir de uma receita ou vincular a receita a uma atividade existente (Figura 6.9).
5. Clicar no botão ***Planner*** abre a janela de planejamento, conforme Figura 6.10. Nesta janela é possível acompanhar quantas e quais atividades estão em cada fase, reordenar atividades em TO DO, e atribuir um responsável para as tarefas em Unplanned (Figuras 6.11 e 6.12).
6. Clicar no botão ***Team*** abre a janela de visualização da equipe, conforme Figura 6.13. Nesta janela é possível visualizar as habilidades do personagem e atividade atual (Figura 6.14) e os itens que está carregando consigo (Figura 6.15).
7. Os personagens com atividades atribuídas começarão a se mover entre as salas para realizá-las, conforme mostra a Figura 6.16.

Na versão atual, disponibilizada para testes, o projeto pede a criação de um **Lingote de Ferro** (*iron ingot*), porém não foi determinado um tempo limite para execução deste e foram disponibilizados os itens base (*iron ore, oil, coal*) em quantidade suficiente para testar todas as receitas disponíveis no jogo atualmente.

Figura 6.2 - Tela inicial do jogo

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.3 - Janela de projetos

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.4 - Janela de gerenciamento de atividades - Visão geral

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.5 - Janela de gerenciamento de atividades - Criação de atividades

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.6 - Janela de gerenciamento de atividades - Seleção de responsável por atividade

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.7 - Janela de gerenciamento de atividades - Visão completa da atividade

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.8 - Janela de receitas de itens

Fonte: Artes criadas por Luís Filipe Oliveira.

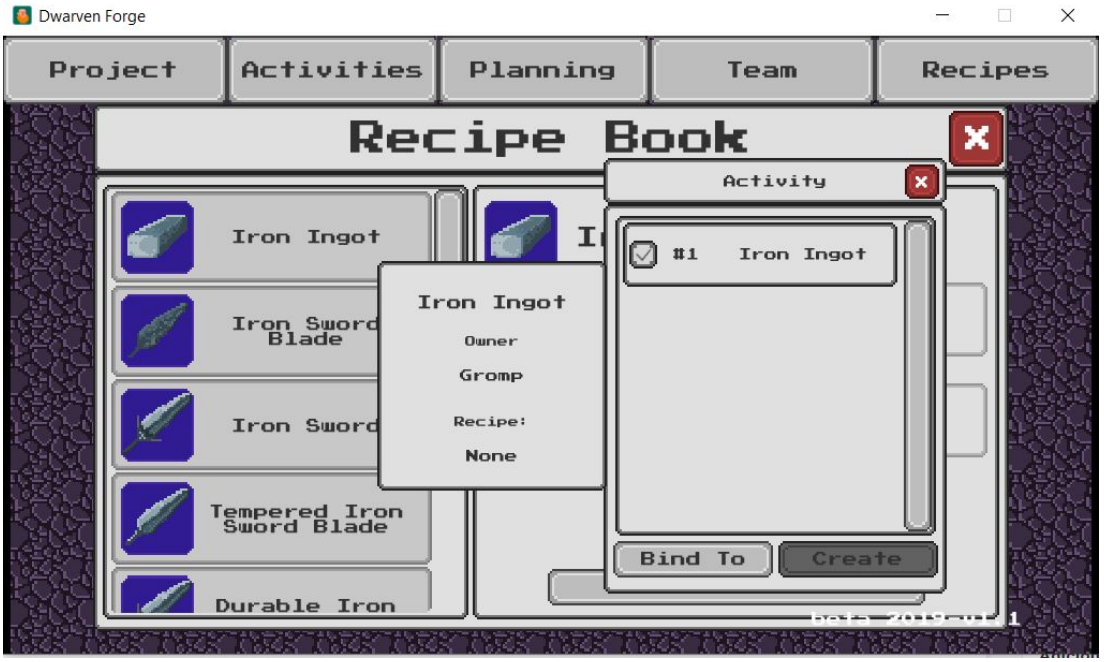

Figura 6.9 - Janela de receitas de itens - Vinculação de receita à atividade

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.10 - Janela de planejamento de atividades

Fonte: Artes criadas por Luís Filipe Oliveira.

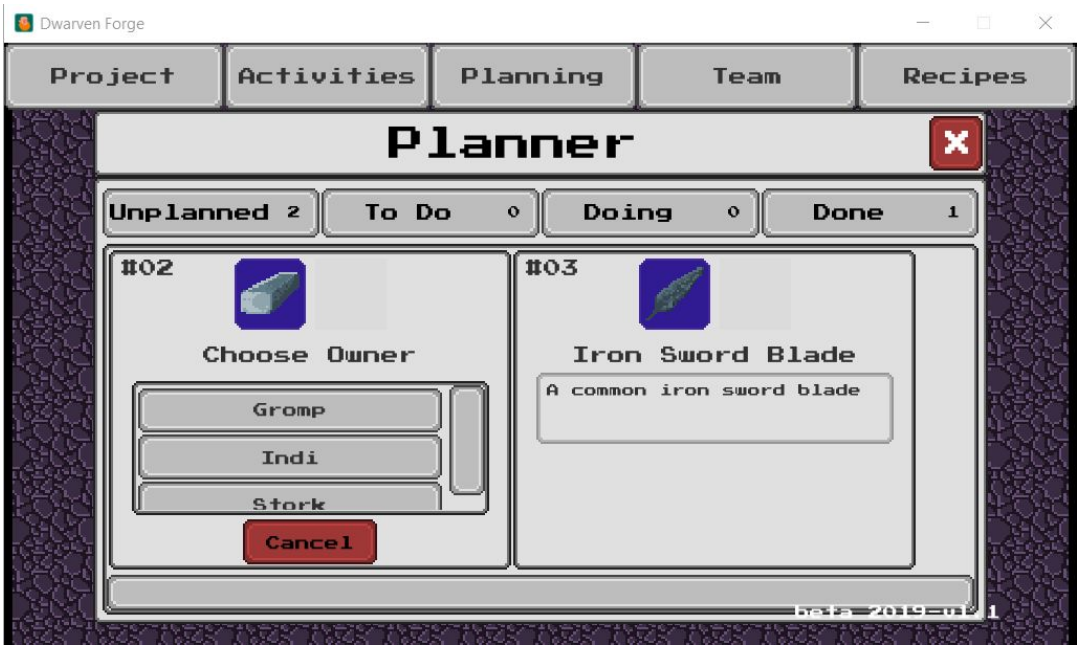

Figura 6.11 - Janela de planejamento - Seleção de responsável por atividade

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.12 - Janela de planejamento de atividades - Responsável selecionado

Fonte: Artes criadas por Luís Filipe Oliveira.

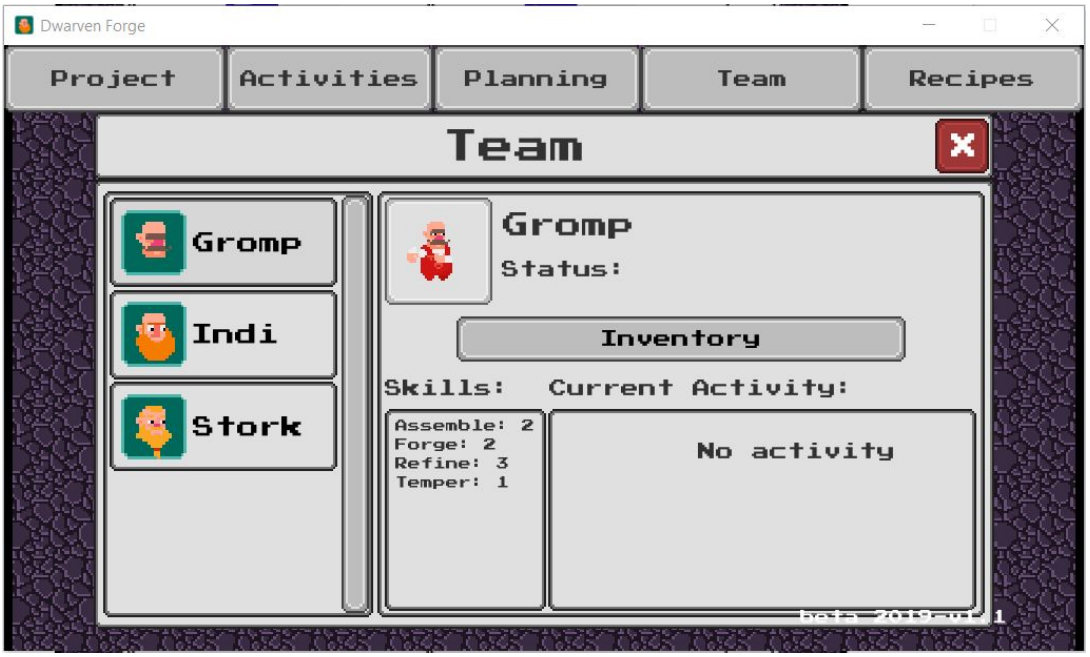

Figura 6.13 - Janela de visualização de equipe.

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.14 - Janela de visualização de equipe - Personagem alocado

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.15 - Janela de visualização de equipe - Visão do inventário

Fonte: Artes criadas por Luís Filipe Oliveira.

Figura 6.16- Personagens se movendo entre salas para executar suas tarefas.

Fonte: Artes criadas por Luís Filipe Oliveira.

7 - Testes e Avaliação

7.1 - Testes com usuários

Para avaliar a aderência do jogo à gerência de projetos, foi criado um formulário que contém questões sobre as áreas de conhecimento de projetos. É importante lembrar que nem todas as áreas foram implementadas no protótipo testado, mas todas estão presentes no teste, pois o objetivo é verificar a percepção destas no jogo e colocar apenas as áreas implementadas poderia induzir os participantes a avaliá-las positivamente. Vale mencionar também que o número de testadores foi bem reduzido, apenas 6 participantes responderam à pesquisa. Por outros meios de comunicação, alguns participantes também contribuíram com importantes feedbacks com relação a bugs e sugestões de melhorias.

O formulário apresentava cada uma das áreas de gerência de projetos e uma escala de 0 a 5, sendo 0 não percebido e 5 presente. E no final, um campo de texto livre para *feedbacks* relacionados a outros aspectos do jogo. Os textos que descrevem as áreas de gerência de projetos apresentados na pesquisa e seus respectivos resultados são:

Gerência de Escopo - Responsável por definir as atividades que devem ser realizadas para entregar o resultado e garantir que somente o trabalho necessário para que o projeto seja finalizado seja realizado. Também define critérios para determinar se o projeto foi completado (Figura 7.1).

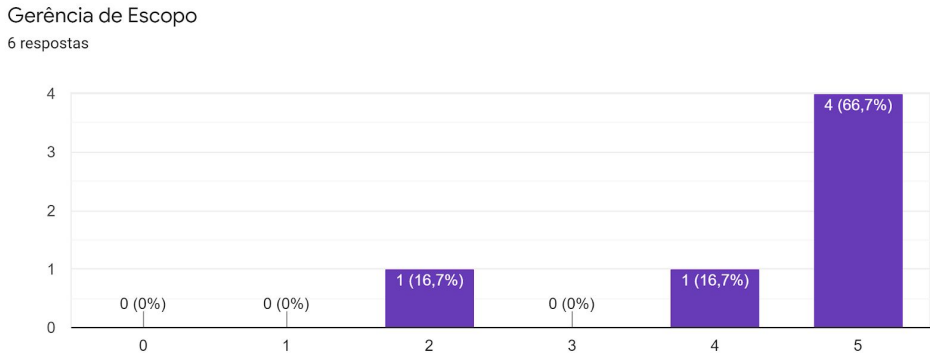

Figura 7.1 - Percepção da Gerência de Escopo pelos usuários

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Custos - Responsável por estimar o custo do projeto, planejar e gerenciar os gastos para garantir que o projeto seja realizado dentro do orçamento definido (Figura 7.2).

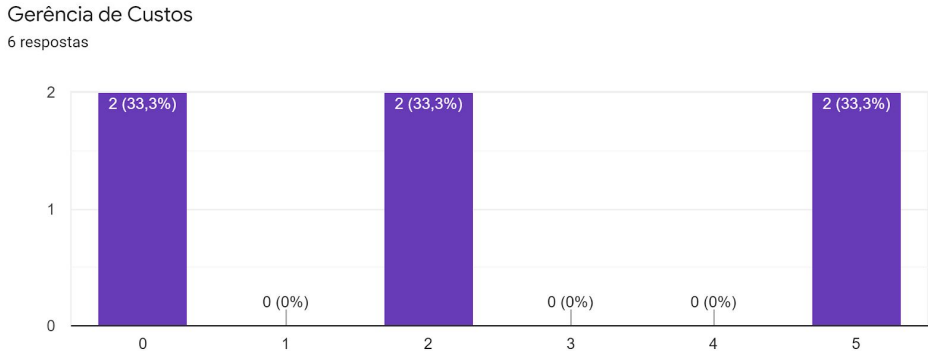

Figura 7.2 - Percepção da Gerência de Custo pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Qualidade - Responsável por garantir que o projeto satisfaça os objetivos e funções para os quais ele foi concebido. Normas e padrões de qualidade

são definidos nos processos dessa área, bem como auditorias, buscando sempre a melhoria contínua (Figura 7.3).

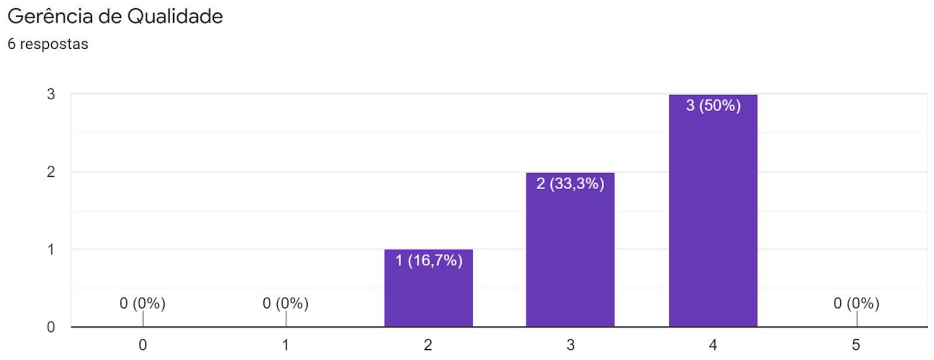

Figura 7.3 - Percepção da Gerência de Qualidade pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Aquisições - Responsável por adquirir bens e serviços externos à organização executora, além de gerenciar as entregas, pagamentos e contratos referentes a estas aquisições (Figura 7.4).

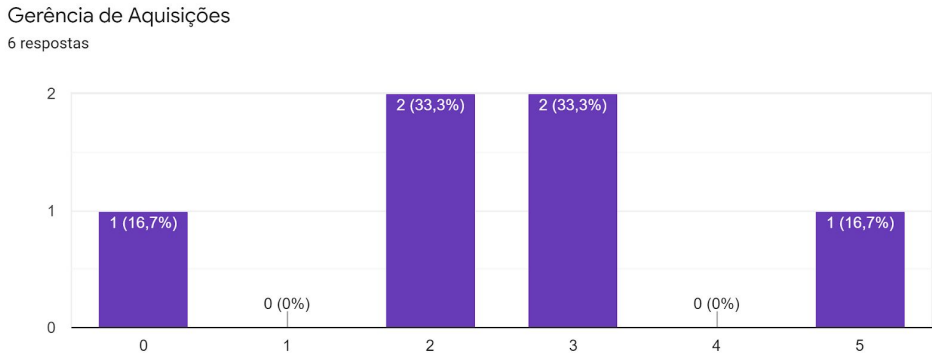

Figura 7.4 - Percepção da Gerência de Aquisições pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Recursos - Responsável por organizar e gerenciar a equipe do projeto, bem como a identificação da necessidade de aquisição de materiais, equipamento ou espaço para realização do trabalho (Figura 7.5).

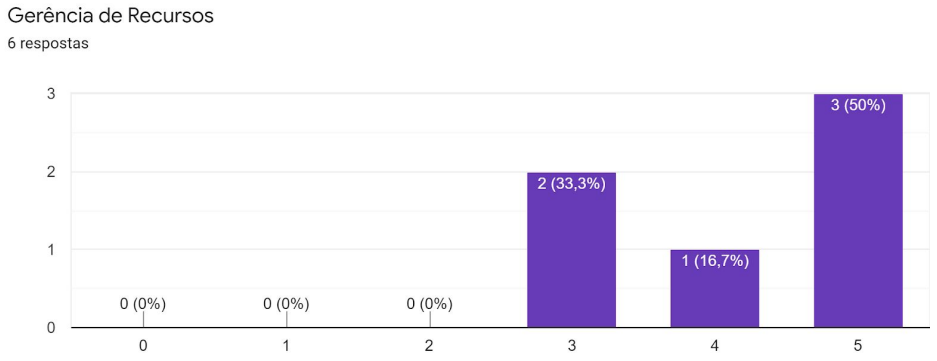

Figura 7.5 - Percepção da Gerência de Recursos pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Comunicação - Responsável por garantir o desenvolvimento, recolhimento, distribuição, armazenamento, recuperação e entrega das informações sobre o projeto de forma oportuna e adequada (Figura 7.6).

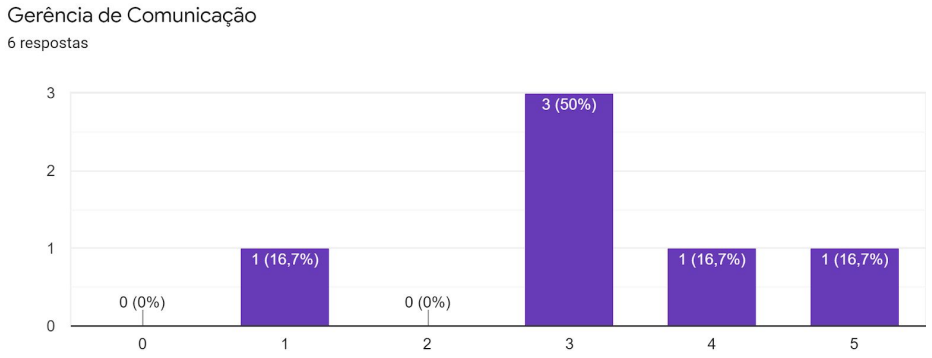

Figura 7.6 - Percepção da Gerência de Comunicação pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Risco - Responsável por monitorar e controlar constantemente riscos que possam aparecer ao longo do projeto. O risco de um projeto é uma incerteza, que pode ter efeitos negativos ou positivos sobre algum dos aspectos do projeto, como escopo, custo, tempo ou qualidade (Figura 7.7).

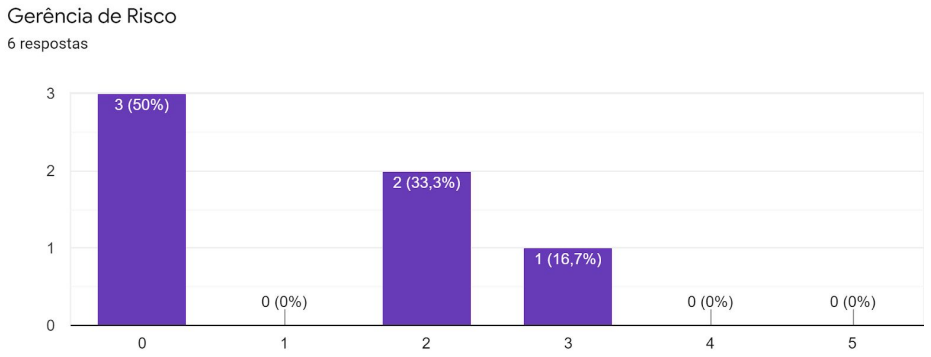

Figura 7.7 - Percepção da Gerência de Risco pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Cronograma - Responsável por garantir que o andamento das etapas e atividades estejam de acordo com o cronograma e que a entrega do projeto ocorra no prazo, da iniciação ao encerramento do projeto. Para isso, as atividades do projeto são sequenciadas de acordo com sua importância, urgência e/ou prioridade, além de receberem uma estimativa do tempo necessário para que sejam concluídas (Figura 7.8).

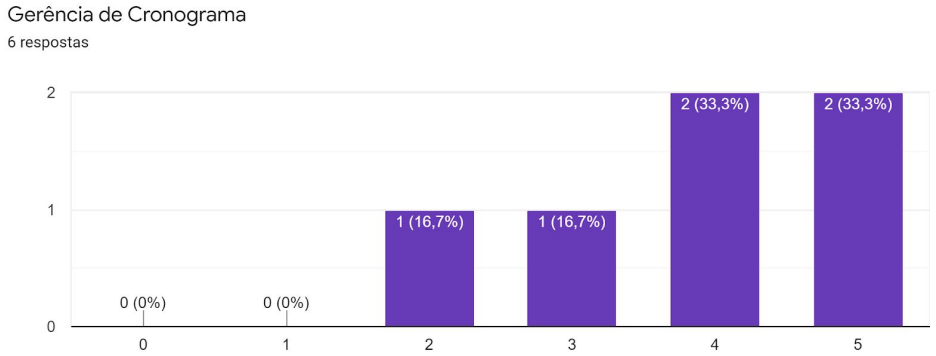

Figura 7.8 - Percepção da Gerência de Cronograma pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Partes Interessadas - Responsável por identificar os grupos, pessoas ou organizações que podem impactar ou ser impactados por uma decisão, atividade ou resultado do projeto e entender suas motivações e necessidades para melhor atendê-las (Figura 7.9).

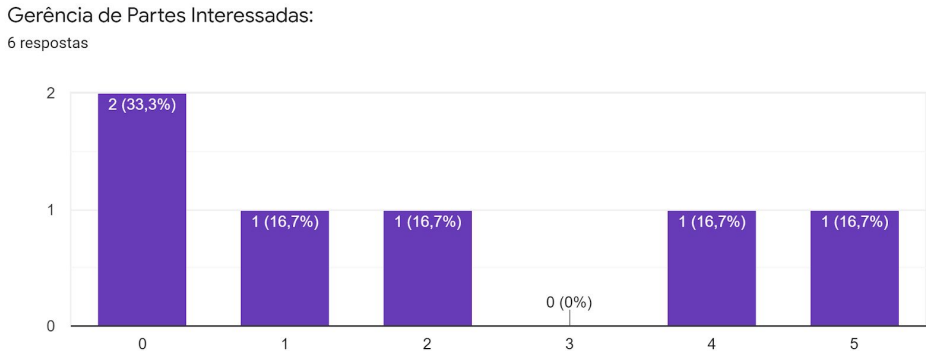

Figura 7.9 - Percepção da Gerência de Partes Interessadas pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Gerência de Integração - Responsável por integrar e manter em sincronia todas as outras. Além disso, também envolve desenvolvimento dos termos de abertura e encerramento, planejamento do projeto, monitoramento e orientação do trabalho (Figura 7.10).

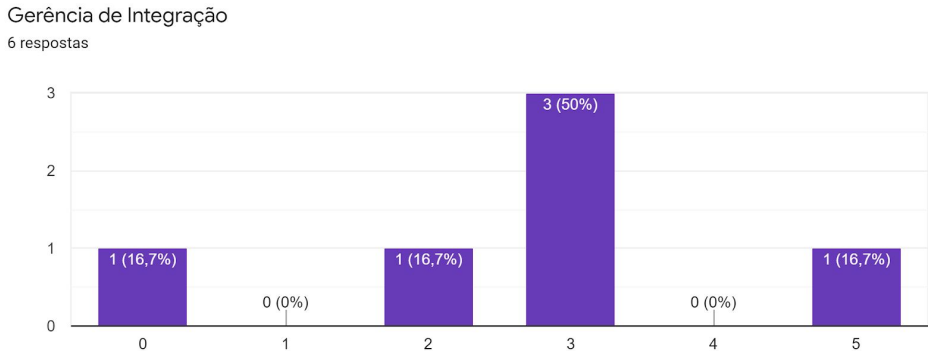

Figura 7.10 - Percepção da Gerência de Integração pelos usuários.

Fonte: *Printscreen* dos resultados da pesquisa.

Das áreas de conhecimento que de fato foram implementadas, que são Gerência de Escopo, Gerência de Recursos, Gerência de Cronograma, Gerência de Comunicação e Gerência de Integração, é possível observar um resultado positivo, com mais da metade das respostas acima dos 3 pontos. Na Gerência de Integração houve uma resposta 0, que provavelmente se deve a uma interpretação do texto apresentado, que se refere às documentações como termo de abertura e encerramento, que não são parte do escopo do jogo.

Das não implementadas, que são a Gerência de Custos, Gerência de Qualidade, Gerência de Aquisições, Gerência de Risco e Gerência de Partes Interessadas, são apresentados resultados que não eram esperados, tendo uma quantidade significativa de respostas em parcialmente presente ou presente. Na Gerência de Custos, são apresentados dois votos em 5 pontos, o que pode ser uma interpretação do uso dos materiais para criar itens como um custo de criação. Para Gerência de Qualidade, houve uma percepção geral de que esta estava implementada, o que provavelmente se deve à uma das etapas do processo de criação do item que que se chama refinamento. Na Gerência de Aquisições existe uma concentração de respostas em 2 e 3 pontos, ou seja, existe uma interpretação de que as aquisições estão parcialmente implementadas no jogo, provavelmente pelo fato de no início da

partida o jogador selecionar os personagens para seu time, quase como uma contratação, o que poderia ser interpretado como a aquisição do mesmo. Por fim, a Gerência de Partes interessadas apresentou uma resposta em 4 e 5 pontos, que apesar de não ter nenhuma menção direta ao cliente, existe uma descrição do pedido, que poderia ser interpretado como uma representação deste.

Com este resultado, foi possível identificar pontos de melhoria nas áreas de conhecimento implementadas, tornando-as mais claras para o jogador, além de insumos para adicionar novas áreas e em que pontos estas se tornariam mais visíveis. Os feedbacks recebidos por outros meios também foram bem interessantes, tratando de questões mais pessoais do que gostariam de ver no jogo como um sistema de missões, uma loja para comprar materiais, mais anões para escolher, mais itens para fazer etc. Dentro do que foi proposto foi uma resposta bem positiva.

7.2 - Análise

Tendo em vista a versão do jogo apresentada é possível fazer uma análise sob a luz das definições apresentadas por Schell e McGonigal sobre o que é um jogo. Começando pelas qualidades apresentadas por Schell (2011):

1. **Jogos são jogados voluntariamente:** A decisão de fazê-lo um jogo de entretenimento e não um jogo puramente educacional está ligada a esta característica. A carga de obrigatoriedade (em aprender ou estudar) que vem junto a este rótulo de "educacional" tende a ser uma barreira para que o jogador consiga aproveitar efetivamente o jogo.
2. **Jogos têm objetivos:** Existe ainda uma necessidade de melhorar a apresentação do objetivo para o jogador, pois para muitos não ficou muito claro o que deveria ser feito. Mas o objetivo está lá: conduzir os anões por todo o processo de construção de um item.

3. **Jogos têm conflitos:** O conflito em Dwarven Forge não está relacionado à existência de um "lado inimigo" com o qual os anões precisam competir, mas sim na dinâmica da própria forja. O conflito existe no fato de que os anões possuem uma série de itens avulsos, mas precisam trabalhar em equipe e utilizá-los para construir os itens das receitas.
4. **Jogos têm regras:** As regras ditam como o jogador deve interagir com o jogo para atingir seu objetivo de construir um item no final. Neste caso, ele deve utilizar as janelas de interface para interagir com os personagens e com o projeto, criando e alterando tarefas, descobrindo receitas, atribuindo tarefas aos personagens e alterando a ordem em que devem ser executadas para que o produto final possa ser entregue como esperado.
5. **Jogos podem levar à vitória ou derrota:** Na versão atual, o jogo não trata vitória ou derrota, pois foi estruturado na forma de sandbox, para execução de testes com usuários.
6. **Jogos são interativos:** Toda ação prevista do jogador no jogo possui uma resposta, direta ou indireta, que vai ser apresentada a ele posteriormente, como atribuir uma atividade à um personagem e ele começar a trabalhar nela, ou alterar a ordem das atividades no quadro de atividades.
7. **Jogos têm desafios:** O desafio no momento é guiar os personagens disponíveis para que realizem a entrega do que foi pedido, com os recursos disponíveis no jogo.
8. **Jogos podem criar valores internos próprios:** Ainda não existem mecânicas neste jogo que permitam a percepção de valor do que é produzido ou recebido dentro do jogo, como dinheiro para melhorar equipamentos, aumentar o espaço de trabalho ou até treinar e recrutar novos personagens.
9. **Jogos envolvem os jogadores:** O jogo teve uma resposta bem positiva das pessoas que testaram, inclusive pedindo adição de novas funcionalidades e conteúdos no jogo.
10. **Jogos são sistemas fechados e formais:** O jogo é composto por um conjunto de elementos interligados que dado uma entrada do usuário, é recebido um

feedback e para que isso aconteça, esse sistema segue um conjunto de regras que limitam como essa interação acontece.

Já pela definição de McGonigal (2012) é possível avaliar o seguinte:

1. **Metas:** A meta, ou objetivo é apresentado ao jogador. Assim como citado no item 2 da lista anterior, o jogo ainda precisa de melhorias para deixar mais claros os objetivos para o jogador.
2. **Regras:** Assim como no item 4 da lista anterior, as regras foram implementadas.
3. **Sistema de Feedback:** Os personagens se movendo e executando animações diferenciadas, mudanças na estrutura e valores das interfaces de usuário em tempo real indicam o que está acontecendo no jogo, para que o jogador possa decidir seus próximos passos.
4. **Participação voluntária:** assim como no item 1 da lista anterior, foi proposto um jogo com fins de entretenimento para facilitar a imersão do jogador no contexto do jogo sem a obrigatoriedade de estar ali.

8 - Conclusão

8.1 - Conclusão

Este projeto teve início a partir de uma proposta: aproveitar o potencial educacional dos jogos para ensinar sobre gerência de projetos. Inicialmente, levou-se em consideração a criação de um jogo educativo - porém, essa terminologia foi logo descartada. Afinal, segundo Costa (2009), jogos com fins pedagógicos são construídos de forma diferente dos jogos com fins de entretenimento. Muitas vezes, não levam em conta aquilo que se está tentando ensinar ao jogador: simplesmente incorporam esse conteúdo em estruturas de jogos de entretenimento já consagrados, falhando dessa maneira tanto como jogos divertidos, tanto quanto jogos educativos.

Assim, o jogo resultante deste trabalho não deveria simplesmente se propor a atuar como jogo educativo, falar sobre gerência de projetos e ter uma mecânica genérica. Para garantir que o potencial educacional dos jogos fosse realmente aproveitado, seria preciso criar um jogo final que utilizasse a gerência de projetos como mecânica principal. Que funcionasse como sugere Schell (2008) com sua Tétrade Elementar: todos os elementos do jogo - mecânica, estética, narrativa e tecnologia - deveriam se apoiar e se complementar de forma a funcionar, dentro do jogo, fundamentalmente da mesma forma como a gerência de projetos funciona na vida real.

Para que isso acontecesse, *Dwarven Forge* foi projetado para que sua mecânica incorporasse vários dos conceitos vistos na gerência de projetos. Os protótipos sofreram grandes mudanças com o passar do tempo até que se chegasse em sua versão atual. De acordo com dados do PMI (2018), um projeto é um conjunto de atividades temporárias realizadas em grupo, destinadas a produzir um produto, serviço ou resultado. *Dwarven Forge*, então, apresenta ao jogador um conjunto de tarefas (executar receitas usando os recursos disponíveis) que serão realizadas por um grupo de personagens (os anões) de forma que sejam criados produtos (itens fantásticos).

O fluxo do jogo funciona de acordo com os cinco grupos de atividades pelas quais a gerência de projetos pode ser dividida, segundo Silva (2015): início, planejamento, execução, monitoramento e controle e encerramento. Além disso, também segue a dinâmica sequencial, assim como o modelo em cascata de Royce (1970), em que para se atingir uma fase, é necessário ter completado todas as atividades da fase anterior - ou seja, para que um item seja criado, o jogador primeiro precisa planejar as atividades e antes disso descobrir o que é necessário para realizá-las.

Por meio da interface, o jogador atua como gerente de projetos e controla tudo o que acontece. Navegando pelos botões e janelas, é o jogador quem determina que tipos de itens serão criados, qual será o personagem designado a cada tarefa, a qual item será dada prioridade na lista de tarefas. O jogador tem acesso à lista de habilidades dos personagens, aos requisitos de cada item para poder ser construído e ao quadro com as listas de tarefas.

As ações que os jogadores podem realizar através destas interfaces estão relacionadas a conceitos como gerenciamento de escopo, de custos, de integração, entre outros, segundo as áreas de conhecimento estabelecidas pelo PMI (2018).

A narrativa fantástica também contribui para o entendimento do jogador sobre gerência de projetos, pois é fácil fazer a diferenciação entre os produtos e personagens que estão sendo gerenciados. A estética simples, em *pixel art*, não só torna o arquivo do jogo leve em termos de *bytes* como também ajuda o jogador a compreender rapidamente a mecânica e a narrativa - o que permite que concentre sua atenção nos elementos da gerência de projetos incorporados ao jogo.

Enquanto isso, a tecnologia utilizada neste projeto, apesar de não ser vista diretamente por quem joga o jogo, também foi peça crucial para fechar a Tétrade Elementar de Schell (2011). O desenvolvimento por meio da game engine *Unity* possibilitou que fossem criadas máquinas de estados para que as mecânicas acontecessem em tempo real de acordo com os comandos do jogador, o que não seria possível em um jogo de tabuleiro, por exemplo.

Assim, é possível afirmar que a proposta inicial foi cumprida: os elementos de *Dwarven Forge* funcionam em sintonia entre si e de acordo com a gerência de projetos, fazendo com que o jogo entretenha o jogador enquanto discretamente ensina a ele habilidades que precisam ser usadas no mundo real durante o gerenciamento de um projeto verdadeiro.

Porém, este trabalho vai além do cumprimento da proposta inicial. Este projeto demonstra ser possível a criação de jogos de entretenimento baseados em atividades reais e que desta forma, o valor educacional deste encontra-se principalmente em suas mecânicas e não necessariamente em sua temática. Esta visão permite uma exploração ainda maior desta conexão entre atividades do mundo real e mecânicas de jogos, ensinando aos jogadores habilidades além das já conhecidas, podendo ser potencializadas se usadas em conjunto com atividades acadêmicas.

O incorporamento de conteúdos acadêmicos de forma tão intrínseca no jogo, participando de sua mecânica principal, possibilita que o jogador não se sinta jogando um jogo educativo, mas sim que aprenda habilidades utilizadas no mundo real enquanto joga um jogo de entretenimento. Este aprendizado é ainda mais bem-aproveitado pois, no jogo, é possível errar quantas vezes forem necessárias até que se aprenda o conteúdo que está sendo abordado - ao contrário do que acontece na vida real, onde errar muitas vezes traz consequências negativas.

8.2 - Trabalhos Futuros

O jogo terá seu desenvolvimento continuado após a entrega deste trabalho, visando a implementação de novas mecânicas correspondentes a áreas do conhecimento de projetos não abordadas nesta versão inicial, melhorias na interface do usuário e usabilidade, ampliação dos desafios do jogo, adição de um sistema bilíngue (pois o mesmo se encontra inteiramente em inglês), além de melhorias e adições na parte visual.

Conforme novas características forem adicionadas, novos testes com usuário serão realizados, com objetivo de avaliar a aderência das mecânicas às áreas do conhecimento da gerência de projetos, assim como garantir que o jogo se mantenha divertido.

Referências

COSTA, L. D. O que os jogos de entretenimento têm que os jogos educativos não têm. VIII Brazilian Symposium on Games and Digital Entertainment, p. 1–20, 2009.

KOSTER, Raph. **A Theory of Fun for Game Design**. 2. ed. Sebastopol: O'Reilly , 2014, 279p.

MCGONIGAL, Jane. (2011). **A realidade em jogo**. Trad. Eduardo Rieche. Rio de Janeiro: BestSeller, 2012. 377p.

SCHELL, Jesse. (2008) **A arte de game design**: o livro original. Trad. Edson Furmankiewicz. Rio de Janeiro: Elsevier, 2011, 489p.

SILVA, Fabiana Bigão; MOURA, Myrian. **Fundamentos da Gestão de Projetos**. Ebook, 2015. 58p.

PMI. O que é Gerenciamento de Projetos? Disponível em: brasil.pmi.org/brazil/AboutUS/WhatIsProjectManagement.aspx. Acesso em: Novembro de 2019

UNITY. Unity para todos. Disponível em: unity.com. Acesso em: Novembro de 2019

Aseprite. Animated Sprite Editor and Pixel Art Tool. Disponível em: https://www.aseprite.org/. Acesso em: Novembro de 2019