Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Sistema para Projeto e Análise de Recursos em Jogos baseados em Lista de Materiais

Estudo de Caso em um Jogo de Sobrevivência

Carolina Ribeiro Oliveira

JUIZ DE FORA
FEVEREIRO, 2022

Sistema para Projeto e Análise de Recursos em Jogos baseados em Lista de Materiais
Estudo de Caso em um Jogo de Sobrevivência

Carolina Ribeiro Oliveira

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Sistemas de Informação

Orientador: Igor de Oliveira Knop

JUIZ DE FORA
FEVEREIRO, 2022

SISTEMA PARA PROJETO E ANÁLISE DE RECURSOS EM JOGOS BASEADOS EM LISTA DE MATERIAIS

Estudo de Caso em um Jogo de Sobrevivência

Carolina Ribeiro Oliveira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Igor de Oliveira Knop
D.Sc. Modelagem Computacional

Luciana Conceição Dias Campos
D.Sc. Engenharia Elétrica

Marcelo Caniato Renhe
D.Sc. Engenharia de Sistemas e Computação

JUIZ DE FORA

15 DE FEVEREIRO, 2022

Resumo

Jogos digitais de sobrevivência são uma forma de entretenimento nos quais o jogador possui um único objetivo: utilizar os recursos disponíveis para aumentar suas chances de prosperar em um ambiente hostil. O desenvolvimento de tal gênero possui como um de seus obstáculos o balanceamento da árvore de manufatura, ou seja, a lista de materiais necessária para construção (*crafting*) de cada um dos itens do jogo. Um jogo bem desenvolvido deve prender a atenção do usuário do início ao fim, e a árvore de manufatura impõe o ritmo do progresso do jogador que interfere diretamente nessa experiência. Diante o exposto, o objetivo deste trabalho é melhorar o processo de desenvolvimento de jogos baseados em construção de itens, mais especificamente, jogos de sobrevivência. Para tanto, é criada uma ferramenta para projeto e manipulação de árvores de manufatura e, através desse ferramental, é analisada sua capacidade de representação em dois jogos de sobrevivência existentes. Adicionalmente, um modelo inicial captura conversões de itens por fora da árvore de manufatura, que permite modelar eventos, missões e trocas de itens com personagens não-jogador. Resultados iniciais indicam que a ferramenta permite experimentar previamente o progresso do jogo através de um simulador de estoques e contratos, que podem ser implementados posteriormente no jogo.

Palavras-chave: desenvolvimento de jogos; lista de materiais; jogos de sobrevivência.

Abstract

Digital survival games are a way of entertainment in which the player has a single goal: to use available resources to increase his chances of thriving in a hostile environment. The development of such a genre has as an obstacle the balancing of the manufacturing tree, that is, the list of materials needed to craft each one of the items in the game. A well-developed game must hold the user's attention from beginning to end, and the manufacturing tree imposes the rhythm of the player's progress which interferes directly in this experience. Given the above, the objective of this work is to improve the process of developing games based on crafting items, more specifically, survival games. Therefore, it was created a tool for designing and manipulating manufacturing trees and, through this tool, it was analyzed the representation capability in two existing survival games. Additionally, an initial model captures item conversions outside the manufacturing tree, which allows you to model events, quests, and exchanges with non-player characters. Initial results indicate that the tool allows you to previously experience the game's progress through a stock and contracts simulator, which can be implemented later in the game.

Keywords: game development, bill of materials, survival games.

Agradecimentos

Ao professor Igor Knop pela orientação, cuidado e pela enorme paciência que teve comigo, sem a qual este trabalho não se realizaria. Não fui sua aluna e orientada mais fácil de lidar, mas por todos os sumiços, desculpas e momentos de esforço você me estendeu a mão e me auxiliou como outros não o fariam.

E ao meu amado Gabriell pelo apoio, motivação e encorajamento, você foi meu maior suporte em todos os momentos de dificuldades, loucuras, noites viradas e comemorações, sua contribuição foi essencial para eu completar essa importante etapa da minha vida.

Aos aqui citados e à minha família, que são minha torcida pessoal, muito obrigada!

“Nosso instinto de sobrevivência é a nossa maior inspiração.”.

Interestelar

Conteúdo

Lista de Figuras

Lista de Abreviações e Siglas

API *Application Programming Interface*. 23, 25

AWS *Amazon Web Services*. 33

BOM *Bill of Materials*. 12, 13, 20–23, 26, 27, 36

CSS *Cascading Style Sheet*. 23, 33

DOM *Document Object Model*. 25

HTML *HyperText Markup Language*. 23

ID Identificador Único. 29, 31, 32

JS *JavaScript*. 7, 23–25, 33

JSON *JavaScript Object Notation*. 32

JSX *JavaScript Sintaxe Extension*. 25

MDA *Mechanics Dynamics Aesthetics*. 7, 18, 20

MVP *Minimum Viable Product*. 33, 43, 44

NPC *Non-Player Character*. 20, 41

SASS *Syntactically Awesome Style Sheets*. 33

SPA *Single Page Application*. 25

TS *TypeScript*. 23, 33

WWW *World Wide Web*. 23

1 Introdução

A definição do conceito de jogos é muito debatida, mas de forma ampla pode-se dizer ser uma atividade voluntária que possui regras e objetivos claros, executados com o propósito de se entreter. O jogo digital tem o mesmo propósito, porém mediado por aparelhos eletrônicos como celulares, computadores e consoles. Um dos primeiros jogos eletrônicos surgiu em 1958, chamado "Tennis for Two", desde então o número de jogos digitais foi aumentando exponencialmente e se popularizando rapidamente, possuindo hoje diversos gêneros diferentes (LEITE, 2006).

Conforme a tecnologia evoluiu, simultaneamente o setor de jogos tem acompanhado e se tornado muito lucrativo e atrativo. Em uma pesquisa feita em 2019 pela Agência Brasil, houve uma projeção de que o mercado de jogos teria um crescimento de 5,3% até o ano de 2022 (GANDRA, 2019).

Apesar de atualmente estarmos vivenciando uma época de quarentena, devido à COVID-19 (MILLéO, 2020), esse mesmo cenário melancólico trouxe benefícios inesperados para as empresas de jogos, principalmente de jogos *mobile*. Nos dois primeiros meses de pandemia, houve um aumento de 17% de usuários diários ativos em jogos *mobile*, de 24% na receita de compras em aplicativos, 14% no número de anúncios assistidos diariamente e jogadores estão baixando 84% mais aplicativos (UNITY, 2020).

1.1 Contextualização

Há diversos gêneros de jogos dentre eles: ação, onde o jogador é constantemente desafiado sendo cobrado ações e reações rápidas; aventura, envolve exploração da história com mínimo combate; e estratégia, exige um pensamento tático (MASTERCLASS, 2020). Há também diversos subgêneros, dentre eles o de sobrevivência que será o foco deste trabalho.

Para comprar jogos, as pessoas utilizam muito serviços de distribuição digital de jogos. A mais utilizada atualmente é a Steam, que vem ganhando cada vez mais popularidade com o passar dos anos, isso pode ser observado na Figura 1.1 que mostra a

quantidade de jogos lançados na plataforma desde o seu lançamento.

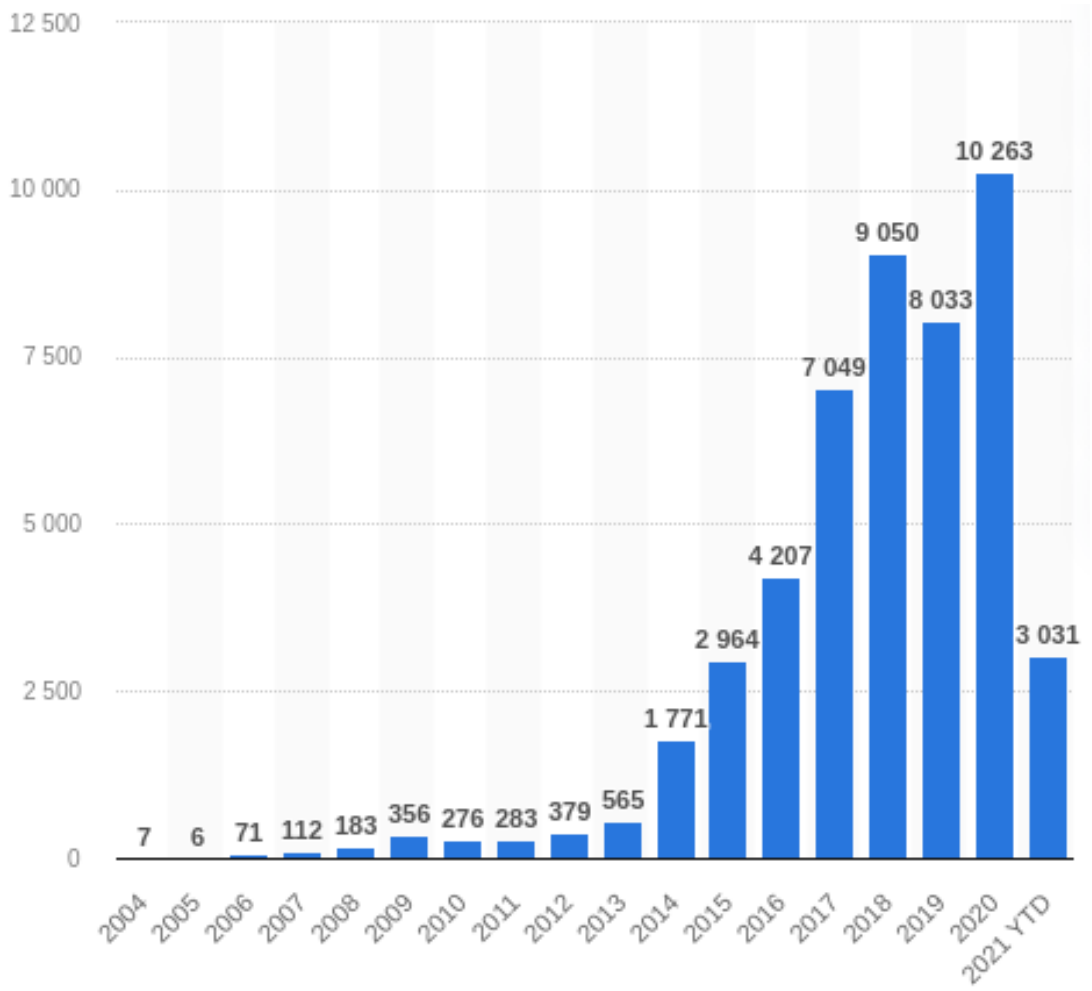

Figura 1.1: Número de jogos lançados na Steam de 2004 a 2021, retirado do site *Statista* (CLEMENT, 2021)

Ao acessar a plataforma Steam (2021b), é possível observar que dentre os dez jogos mais vendidos dois são de sobrevivência, são eles *Terraria* e *Stardew Valley*. Além disso, segundo Jones (2021), sobrevivência é um dos subgêneros que mais se popularizou em 2021, representado na Tabela 1.1 como "Vida e Imersivo".

Nesse subgênero, normalmente o personagem se encontra em mundo desconhecido e com recursos escassos, onde deverá explorar terrenos, juntar materiais, produzir objetos e se alimentar para garantir sua sobrevivência. Muitas vezes esse mundo é uma atmosfera hostil, podendo ocorrer investidas ofensivas de animais, monstros ou até mesmo do próprio terreno.

Uma das partes mais importantes desse subgênero é a construção de objetos a partir de materiais coletados no ambiente. Essa lista de itens e construções possíveis

Tabela 1.1: Gêneros de jogos com maior aumento de popularidade em 2021 (JONES, 2021)

Número	Gênero de jogos	Expectadores Twitch em Abril de 2020	Expectadores Twitch em Abril de 2021	Diferença	Aumento (%)
1	RPG de ação	36.896	336.983	+300.087	+813%
2	Estratégia em tempo real	13.136	76.351	+63.215	+481%
3	Lutas e Artes Marciais	53.006	255.760	+202.754	+383%
4	Quebra-cabeça	88.104	386.571	+298.467	+339%
5	Plataforma e corredor	136.551	596.298	+459.747	+337%
6	Casual	71.814	307.391	+235.577	+328%
7	Vida e Imersivo	52.368	193.171	+140.803	+269%
8	Tiro em terceira pessoa	379.059	1.276.548	+897.489	+237%
9	Esportes de equipe	21.198	62.697	+41.499	+196%
10	Cartas e Tabuleiro	126.406	357.771	+231.365	+183%

de se produzir no jogo é chamada de árvore de manufatura. Ela é decisiva para definir a dificuldade e, consequentemente, diversão do jogo. Por ser uma etapa importante do desenvolvimento de um jogo de sobrevivência, o projetista, ou *game designer*, precisa de bastante atenção em sua criação, balanceamento e testes.

Jogos de sobrevivência atraem seu público ao simular situações de escassez, propondo ao jogador a busca por suas necessidades básicas que se relacionam diretamente com a teoria da motivação humana, a pirâmide de Maslow (ALMEIDA; SCHELSKE; ROVER, 2019). Mapeadas as correlações entre a teoria e o que ocorre nos mundos dos jogos, surgiu a pirâmide mostrada na Figura 1.2. Ela é uma releitura da pirâmide original com a motivação da progressão em jogos, com foco no subgênero de sobrevivência (GRAINER, 2013). Os jogos de sobrevivência, inicialmente se concentram na base da pirâmide e as motivações do jogador progridem gradativamente em direção ao topo.

1.2 Descrição dos problemas

Desenvolver um jogo é um grande desafio dado que é necessária uma equipe interdisciplinar, com habilidades em áreas da programação e até psicologia. O que torna uma tarefa complexa e custosa para pequenas empresas, que saem atrás das grandes quando se trata

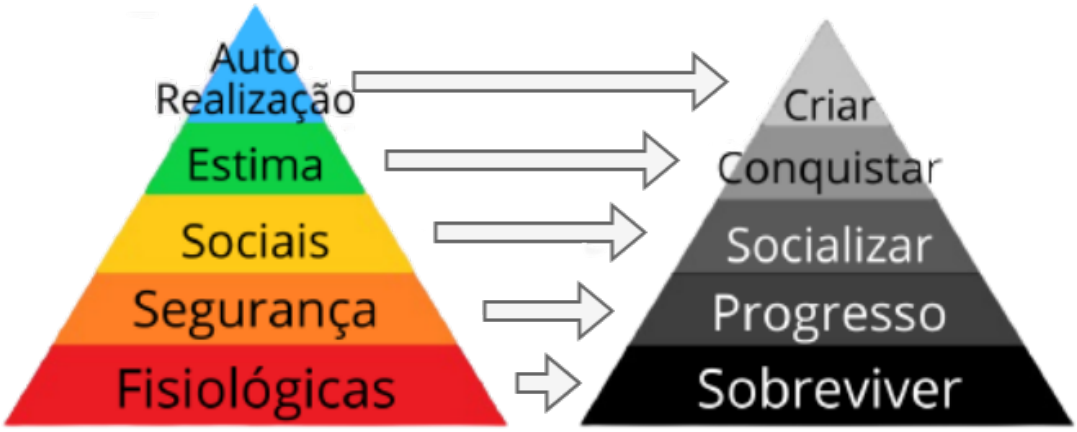

Figura 1.2: Piramide de Maslow e sua releitura para o mundo dos jogos baseado em Grainer (2013)

do custeio e manutenção de equipes desse porte.

Jogos de sobrevivência em específico, têm como principal problema para o jogador o gerenciamento de recursos e estatísticas, como vida, fome e sede, esse equilíbrio é o que mais impacta na sobrevivência do personagem e é o que dita o quão difícil e interessante é a sua jogabilidade. Isso atinge diretamente no maior desafio para o desenvolvedor que é o balanceamento do jogo que, de forma geral, é garantir que a quantidade de conteúdo que o jogo traz seja bem distribuída.

Para balancear jogos do subgênero em questão, é necessário um trabalho minucioso, envolvendo a escolha e evolução da árvore de manufatura. Ela possui um formato de grafo com cada nó sendo um recurso, material ou construção do jogo, e as arestas sendo a relação de manufatura (em uma tradução livre do inglês, *craft*), representando os gastos entre nós. Com isso, deve-se pensar na economia da montagem de itens para fazer sentido no tempo de jogo e que não se torne exaustivo, garantindo uma boa experiência para o usuário.

A definição de um produto a partir de seus componentes é conhecido como *Bill of Materials* (BOM) e diversas aplicações existem em como otimizar as cadeias de produção para tornar o processo mais eficiente e com menores custos (CINELLI et al., 2020; CARMODY et al., 2021; HE et al., 2014). Neste trabalho, os algoritmos e técnicas serão investigados para auxiliar o *game designer* a projetar o processo de criação da árvore de manufatura para um jogo.

1.3 Objetivos

O objetivo geral deste trabalho é aumentar o conhecimento acerca do projeto e uso das árvores de manufatura em jogos utilizando o conceito de lista de materiais, para facilitar e auxiliar o *game designer* a balanceá-la e testá-la antes de sua implementação em um jogo real. Essa investigação utilizará experimentos baseados na literatura relacionado aos principais algoritmos para BOM na idealização e desenvolvimento de uma ferramenta *online*.

O algoritmo se trata de um cálculo recursivo em cima de informações armazenadas localmente ou em um banco de dados, com o intuito de calcular a viabilidade de construção de itens dado uma ou mais unidades de medida e a duração desejada de um dia no jogo e a duração total ou média do jogo completo (KHALAILA; ELIASSEN; BEERI, 1997).

Na ferramenta haverá a visualização dos materiais, o cálculo utilizando uma versão adaptada dos algoritmos BOM, um laboratório digital para simulação de uso e teste de modelos de árvores e, por fim, avaliação do impacto da árvore de manufatura projeta no jogo final.

Como este trabalho tem o propósito de facilitar o desenvolvimento de árvores de manufaturas, serão utilizados dois jogos conhecidos para validar a cobertura da ferramenta em casos reais. Visando testar se o uso do algoritmo auxilia no balanceamento e se a ferramenta cobre todas as necessidades dos modelos inseridos.

Para atingir o objetivo geral, os seguintes objetivos específicos devem ser alcançados:

1. Implementação dos algoritmos para BOM em destaque na literatura;
2. O projeto e implementação de um editor na forma de uma aplicação *web*, chamado *The BOM Builder*, que permita projetar a árvore de manufatura de jogos;
3. Criação de um laboratório de montagem dos itens na aplicação para o projetista simular cenários do jogo;
4. Avaliação da capacidade de representação da ferramenta utilizando modelos reais;

Após a conclusão dos objetivos específicos, será possível analisar os indícios de que a ferramenta desenvolvida neste trabalho pode contribuir ou não para o projeto de jogos de sobrevivência.

1.4 Motivação

O aumento do mercado de jogos implica diretamente na popularidade e na quantidade de jogos desenvolvidos. Os *game designers* precisam criar experiências impactantes e significativas para conseguir que seu jogo se destaque em meio aos demais. Ferramentas de desenvolvimento, *marketing*, ilustração e de efeitos sonoros são as mais comuns. Entretanto, há a oportunidade para novas ferramentas para projeto de mecanismos específicos que comporão a experiência nos jogos.

Dada as dificuldades enfrentadas por empresas de pequeno porte e estúdios *indie*, acreditamos que uma nova ferramenta para facilitar uma das etapas da criação de jogos irá agilizar e baratear o processo de criação da árvore de tecnologia de jogos de sobrevivência.

1.5 Organização do trabalho

Este trabalho foi dividido da seguinte forma: o Capítulo 2 faz uma explicação dos conceitos fundamentais para entendimento deste trabalho; o Capítulo 3 possui o método aplicado para sua realização; o Capítulo 4 detalha o processo de desenvolvimento e o Capítulo 5 analisa os resultados ao representar dois jogos existentes. Por fim, o Capítulo 6 possui as considerações finais, limitações deste trabalho e sugestões de trabalhos futuros.

2 Fundamentação teórica

Este capítulo irá apresentar conceitos, métodos e referências utilizados como base para o desenvolvimento deste trabalho bem como as tecnologias empregadas na ferramenta desenvolvida.

2.1 Jogos durante o isolamento social

Com o surgimento da *internet* e popularização dos computadores pessoais, os jogos começaram a ficar cada vez mais presentes na vida de seus usuários, pois além de proporcionar diversão, também podem ser uma fonte de aprendizado (ARAúJO, 2020). São excelentes ativos para melhorar a coordenação motora, foco e processamento lógico, além de haver também jogos educativos produzidos em parceria com médicos e professores (PINTO; FERREIRA, 2005; CHACON, 2015).

Na pandemia do COVID 19, iniciada no ano de 2019 até o presente, houve uma mudança na busca por entretenimento. A barreira física da quarentena trouxe uma impactante imersão na tecnologia, marcando permanentemente a vida social da população mundial. Como alternativa de diversão, os jogos se tornaram uma grande forma de aliviar o estresse e se conectar com amigos e família. Muitos jogos não só proporcionam diversão, mas também exigem maior trabalho em equipe e trazem ferramentas de comunicação via voz, promovendo maior interação e socialização entre jogadores (BARR; COPELAND-STEWART, 2022).

Eventos e *shows* que previamente eram feitos para grandes multidões em um espaço para festas, agora se encontram em plataformas de jogos, permitindo um evento mundial de maneira mais acessível. Um grande exemplo foi o evento no *Fortnite*, em 24 de abril de 2020, atraindo mais de 12 milhões de jogadores na plateia com o *show* de Travis Scott, *rapper* que ficou na lista de *top* 100 do *Billboard*[1] por mais de 5 anos (BALLHAUS, 2020).

[1] *Billboard* é uma revista americana de música e entretenimento

Essa mudança gerada pelo *lockdown*[2] é possível de ver em números, em 2020 o mercado brasileiro de jogos aumentou exponencialmente. Segundo um estudo feito pela Visa (2021), as transações de cartões Visa realizadas nas principais plataformas e consoles de jogos no último ano cresceram 140% em volume em relação a 2019.

2.2 Gêneros de Jogos e Jogos de Sobrevivência

De acordo com Rogers (2010), jogos são separados em diferentes gêneros e subgêneros. Essa separação é um agrupamento de características das mecânicas dos mesmos, utilizadas para descrever o estilo da jogabilidade. Na Figura 2.1, mostra em laranja alguns exemplos de gêneros como ação, aventura, simulação, estratégia e esportes. E em azul são representados alguns subgêneros, utilizando o gênero de ação como exemplo, ele possui como subgênero jogos de luta, tiro e, em conjunto com aventura, sobrevivência que é o foco deste trabalho.

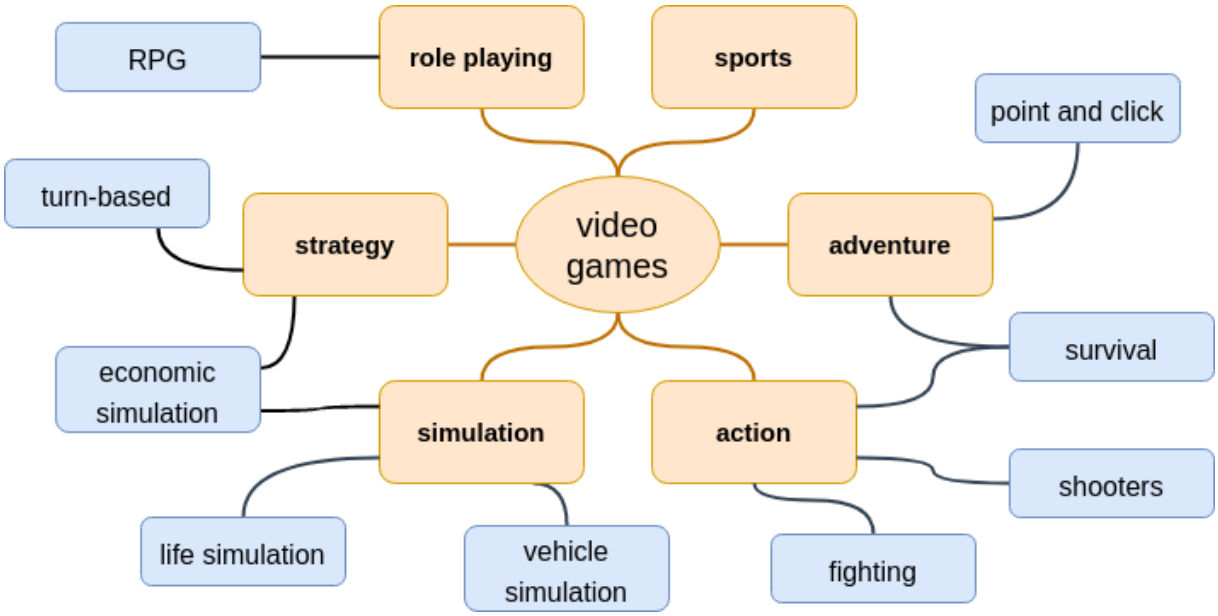

Figura 2.1: Daigrama apresentando alguns gêneros e subgêneros de jogos.

Em um jogo de sobrevivência, normalmente o jogador se encontra em um mundo hostil com recursos escassos devendo fazer escolhas difíceis para manter-se saudável, se alimentar, fazer roupas para ganhar mais armadura ou se blindar contra efeitos de temperatura e criar abrigo.

[2] *Lockdown* é uma medida que consiste em restringir a circulação da população em lugares públicos (DASA, 2021)

Enquanto isso, ele pode se deslumbrar com a liberdade que o jogo traz, podendo expressar sua criatividade através de visuais do personagem ou construções do jogo, escolhendo quais os próximos passos e como ele deverá seguir sua jornada, experimentando o que seria semelhante com a natureza selvagem e havendo a oportunidade de experimentar diversas categorias de trabalho, como jardinagem, construção de casa ou até de uma família (KELLY; NARDI, 2014). Essas características são herdadas do gênero de aventura, cujo foco é a exploração de cenários (FEITOSA; SOUZA, 2018).

Em contrapartida, o ambiente em que o jogador se encontra costuma ser coberto de animais e criaturas perigosas em que, eventualmente, ele deve enfrentá-las, caracterizando a herança de jogos de ação, onde o personagem deve possuir habilidade, velocidade, reflexos e raciocínio rápido.

Utilizando o banco de dados da *Steam* como referência, grande plataforma de jogos que está no mercado há 18 anos e que entrou em um crescimento a partir de 2014 (CLEMENT, 2021), é possível checar a popularidade dos jogos em seu banco de dados (STEAM, 2021a). Para isso, deve-se filtrar para mostrar os jogos de sobrevivência mais jogados de todos os tempos, dessa forma é possível selecioná-los e compará-los, tornando possível observar os jogos do gênero de sobrevivência mais jogados e o crescimento que cada um teve durante os anos, vide Figura 2.2.

Na Figura 2.2, são apresentados os dez jogos mais jogados na Steam atualmente e a popularidade de cada um desde seu lançamento até os dias atuais. Nela é possível visualizar que, como dito na Seção 2.1, houve um grande crescimento de popularidade nos últimos dois anos, sendo eles os anos de isolamento social.

2.3 Projeto e Modelagem de jogos

Segundo Schell (2008), *game design* é o ato de decidir o que um jogo deve ser, começando com uma ideia e produzido para um jogador. O diagrama da Figura 2.3, reproduzida de Schell (2008), permite ver esse processo como um todo. De maneira geral, o *game designer* é a pessoa que projeta a experiência que aparece para o jogador através da interface de um jogo. Para chegar nesse resultado é necessário passar por diversos processos, começando a partir de uma ideia sendo desenvolvida por uma equipe, trazendo os pensamentos para

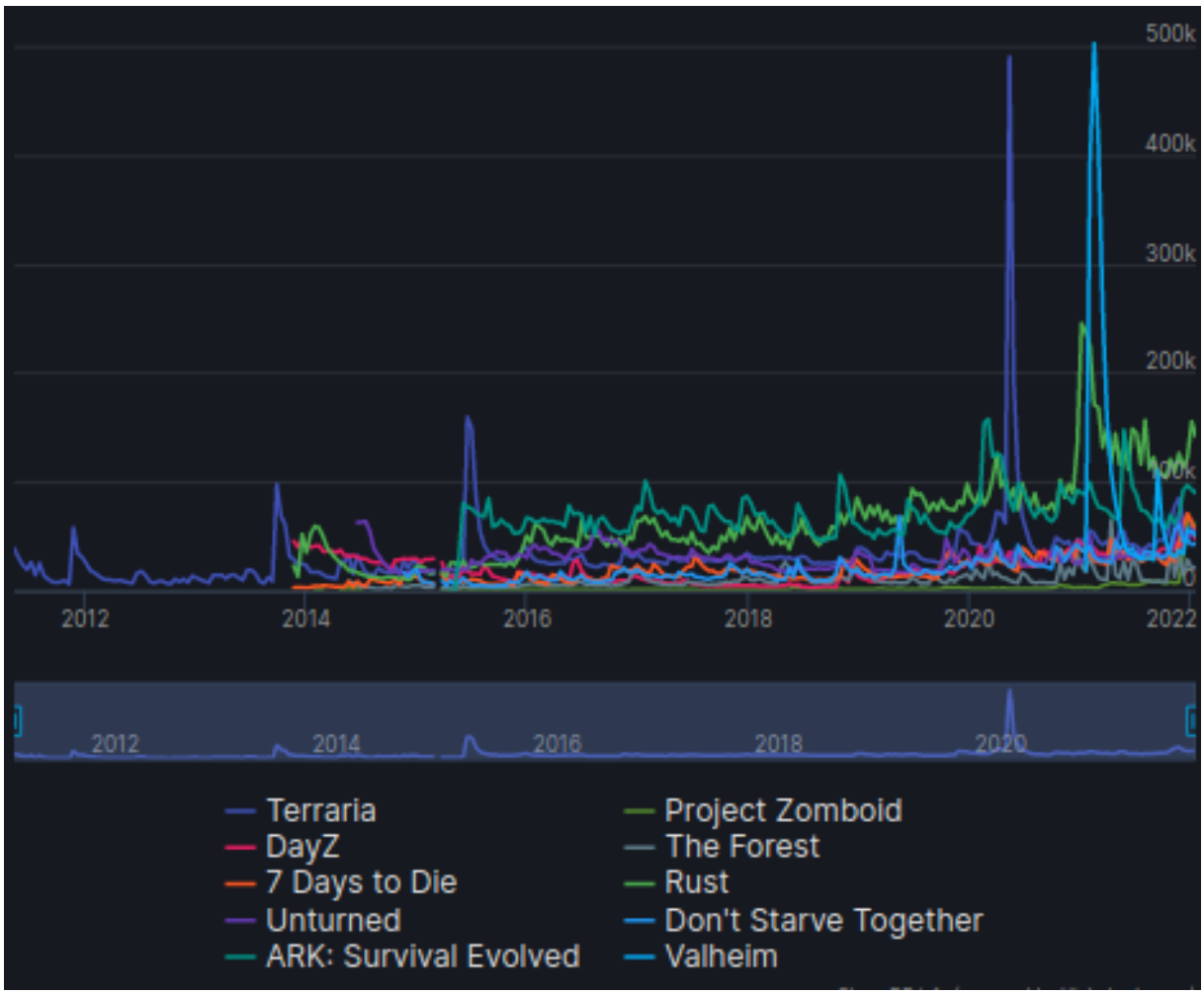

Figura 2.2: Comparação da popularidade dos jogos de sobrevivência mais jogados em todos os tempos na *Steam* (STEAM, 2021a)

realidades concretas através do jogo Schell (2008).

De uma forma mais concisa, e tentando criar uma terminologia para ser usada entre a academia e a indústria, o MDA é um modelo de uso geral que pode ser usado para definir: i) mecânicas, sendo a representação dos dados e algoritmos; ii) dinâmicas, o comportamento das mecânicas em contato com as escolhas do jogador e; iii) estética, sensação e emoção que se deseja passar durante o jogo para o usuário (HUNICKE; LEBLANC; ZUBEK, 2004).

Na Figura 2.4, é mostrado a perspectiva do MDA para o *designer* e para o jogador. Onde o *designer* apenas consegue manipular as mecânicas que a partir dela são geradas as dinâmicas e estética, definindo a jogabilidade e diversão que o jogador terá.

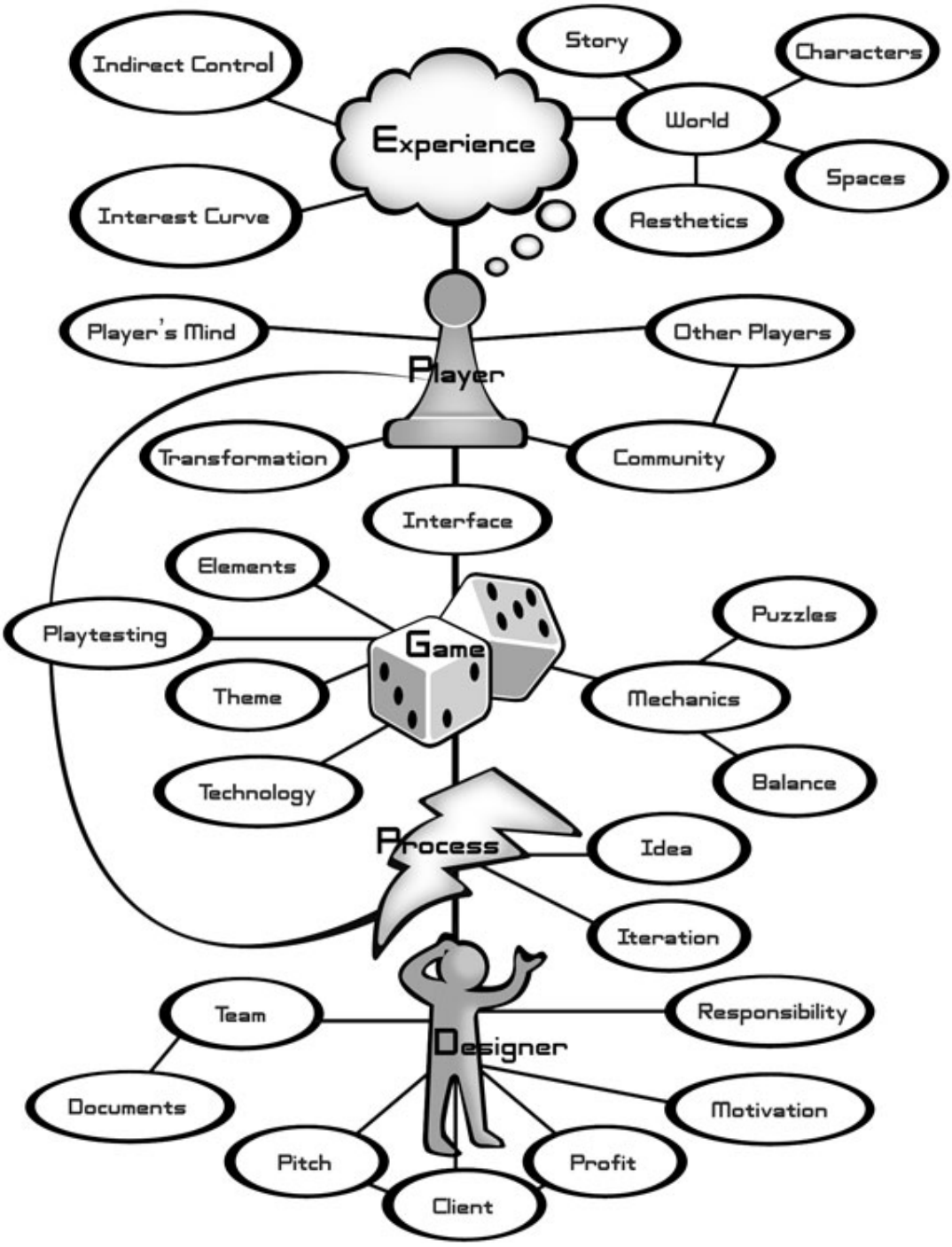

Figura 2.3: Imagem descrevendo o que é *game design* segundo Schell (2008)

2.4 Economia em jogos

Um jogo de sobrevivência não se trata apenas de sobreviver às criaturas e desafios que se encontram pelo mapa, mas apesar disso (KELLY; NARDI, 2014). Deve-se pensar sempre sobre as consequências a longo prazo, permitindo que após uma batalha, consiga voltar para uma base segura e possuir insumos suficientes para sobreviver aos demais fatores do jogo.

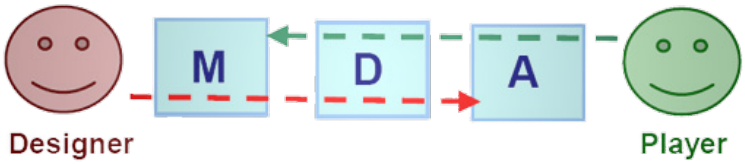

Figura 2.4: Perspectiva do *designer* e do jogador sobre o MDA. Fonte: Hunicke, Leblanc e Zubek (2004).

Para os jogos explorarem a experiência de escassez e a sensação de urgência como motivação e empoderamento, gerado nos jogadores, deve ser criado um modelo complexo de economia de recursos. Esse modelo define quais recursos devem ser limitados ou devem ser repostos, quais eventos naturais impactaram na coleta de materiais ou se haverá *Non-Player Character*s (NPCs)[3], como vendedores, para fazer troca de recursos caso necessário.

Nos jogos do subgênero em questão há dois tipos principais de persona, o conquistador e o emotivo. O primeiro tipo são jogadores que se deleitam em conquistar, são pessoas que jogam visando ganhar poder e acumular itens raros e colecionáveis no jogo. Suas escolhas nos jogos são conduzidas a partir do desejo de completar missões e explorar o desconhecido o máximo possível. Em contrapartida, há jogadores que se apegam ao personagem e suas escolhas são feitas com cautela pensando em mantê-lo e passar pelos desafios com cuidado para manter sua integridade (TOH, 2004).

Ao definir o perfil dos usuários do jogo a ser criado, é possível manipular a árvore de manufatura para atiçar a curiosidade e suprir a necessidade do jogador, definindo também aspectos do jogo e como eles implicam na coleta de recursos e criação de materiais.

2.5 Bill of Materials

O conceito do BOM é a lista de materiais necessários para a construção de um produto. Os algoritmos mais populares são utilizados para o cálculo dos custos e total de materiais necessários para linha de produção, algumas aplicações do BOM são em cadeias de suprimentos de tecnologia médica (CARMODY et al., 2021), aplicações práticas em gestão

[3]NPC é qualquer personagem do jogo que não é controlado por um jogador

Industrial (CINELLI et al., 2020) e projeto de aeronaves comerciais para fabricação (HE et al., 2014).

BOM é um algoritmo recursivo e é aplicado em estruturas que possuem conexões relacionais, como árvores. Ele possui uma estrutura de dados hierárquica de partes necessárias para produzir um item, onde tais partes são convertidas em um novo material (KAZEROONI; KAZEROONI, 2004). Por exemplo, um tampo de madeira, quatro pernas de madeira e alguns parafusos podem ser convertidos em uma mesa.

De acordo com Stapic, Orehovački e Lovrencic (2009), o BOM pode ser representado de diferentes formas. A mais comum é com uma estrutura de árvore exemplificada na Figura 2.5, sendo os nós os materiais a serem produzidos e as arestas representam a conexão entre itens com o valor equivalente à quantia necessária para construção do item pai. A raiz representa o produto final e seus descendentes sendo os componentes necessários para produzi-lo. Enfatizando que, em tal representação, cada material é tratado como um produto separadamente.

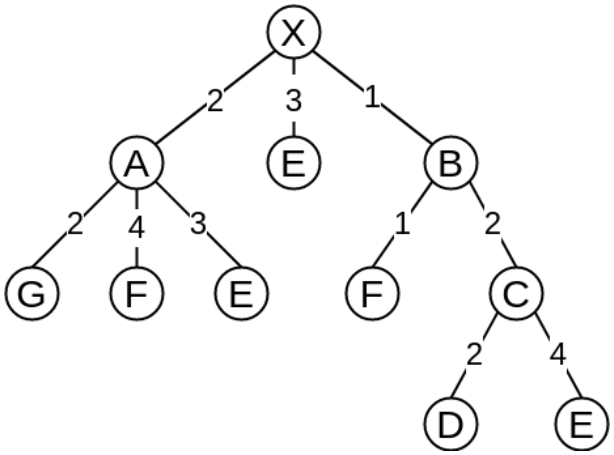

Figura 2.5: Representação em estrutura de árvore (JAVASCRIPT, 2020)

Outro formato de estruturação para o algoritmo é a partir de listas encadeadas, o seu código é mostrado no Algoritmo 1. Comumente as listas encadeadas são implementadas por linguagens que possuem ponteiro ou por referências. Uma opção é implementá-las utilizando de objetos ou tuplas para simular a ligação de encadeamento, caso do Algoritmo 2, forma de representação adotada para desenvolvimento da ferramenta deste trabalho que será apresentada no Capítulo 4.

O Algoritmo 1 é uma visualização simplificada do BOM visto que não possui

um cálculo explicitado. Nele é criado uma lista onde são adicionados os itens requeridos, após sua criação é buscado os descendentes para cada elemento e também os adicionando na listagem. O algoritmo retorna uma listagem contendo uma subárvore do modelo geral.

Algoritmo 1: Releitura traduzida do algoritmo BOM estruturado em lista encadeada de Stapic, Orehovački e Lovrencic (2009)

Entrada: requirements
Saída: Vetor contendo os itens requeridos e todos os seus dependentes

```
1  início
2      list ← inicia um vetor vazio;
3      for para cada element em requirements do
4          list ← element;
5      end for
6      for para cada element em list do
7          for para cada descendant de element do
8              list ← descendant;
9          end for
10     end for
11     return list
12 fim
```

O segundo, e principal para uso deste trabalho, é o Algoritmo 2 que busca partes de uma base, calcula o custo de suas dependências e atualiza o valor da base com o resultado desse cálculo. Ele recebe como parâmetro duas tuplas, a de usos e a de bases, ele passa por cada grupo presente em usos e, para cada unidade desses grupos, ele verifica se é uma subparte da base. Caso seja, o acumulador é atualizado com o valor da unidade e, ao final de todas as unidades do grupo, ele adiciona a parte com o custo atualizado na base.

Ambos algoritmos possuem o mesmo objetivo com formatos ligeiramente diferentes. De maneira geral, neles há uma lista onde nela é passada um vetor com os valores de origem sendo retornado uma segunda listagem onde possui as dependências desses itens. Trazendo a possibilidade dentro de uma grande cadeia de produtos, selecionar uma subárvore para cálculo entre as dependências de um produto.

Algoritmo 2: Releitura traduzida do algoritmo OBOM produzido por Khalaila, Eliassen e Beeri (1997)

Entrada: Uses, Base
Saída: Vetor Base com o valor atualizado dos custos de suas partes

```
1  início
2      cost ← inicia o valor 0;
3      for para cada group em Uses do
4          for para cada unit em group do
5              b ← match(unit.subpart, Base);
6              cost ← cost + b.cost;
7          end for
8          hash_insert((group, cost), Base);
9          cost ← 0;
10     end for
11     return Base
12 fim
```

2.6 Desenvolvimento Web moderno

Aplicações web modernas possuem a expectativa de estarem sempre disponíveis e responsivas, devendo ser seguras, flexíveis e escaláveis. Cada vez mais, cenários complexos pensados para uma melhor interação entre os usuários e a aplicação são construídas usando *HyperText Markup Language* (HTML), *Cascading Style Sheet* (CSS) e *JavaScript* (JS), comunicando-se por meio de *Application Programming Interface*s (APIs) da web (MDN, 2022; MICROSOFT, 2021).

2.6.1 Javascript e Typescript

TypeScript (TS) é um *superset* do JS, uma das principais tecnologias da *World Wide Web* (WWW). Isso significa que ele ainda é o JS, porém com um conjunto de melhorias na linguagem. As principais funções que existem no TypeScript (2012) são: i) conceitos de orientação a objetos, como classes e a adição de *interface* e *abstracts* que não existem no JS ou foram adicionados só mais recentemente; ii) tipagem estática, que diminui a flexibilidade, mas aumenta a consistência do código, reduzindo as chances de enganos por parte dos programadores; iii) análise estática, permitindo verificação do código sem executá-lo, em tempo de desenvolvimento ao invés de execução.

2.6.2 Frameworks para *front-end* modernos

Segundo JavaScript (2020), reconhecido mundialmente por suas pesquisas anuais sobre a linguagem, o ReactJS (2013) é a ferramenta mais utilizada por desenvolvedores de JS a pelo menos cinco anos, seguido diretamente do Angular (2016) e Vue.js (2014), dois frameworks muito populares.

A Figura 2.6 mostra os *frameworks* para *front-end* mais utilizados nos últimos anos, organizados por maior utilização, onde a porcentagem mostrada é o indício de quantos pessoas que utilizam da biblioteca voltariam a utilizá-la novamente.

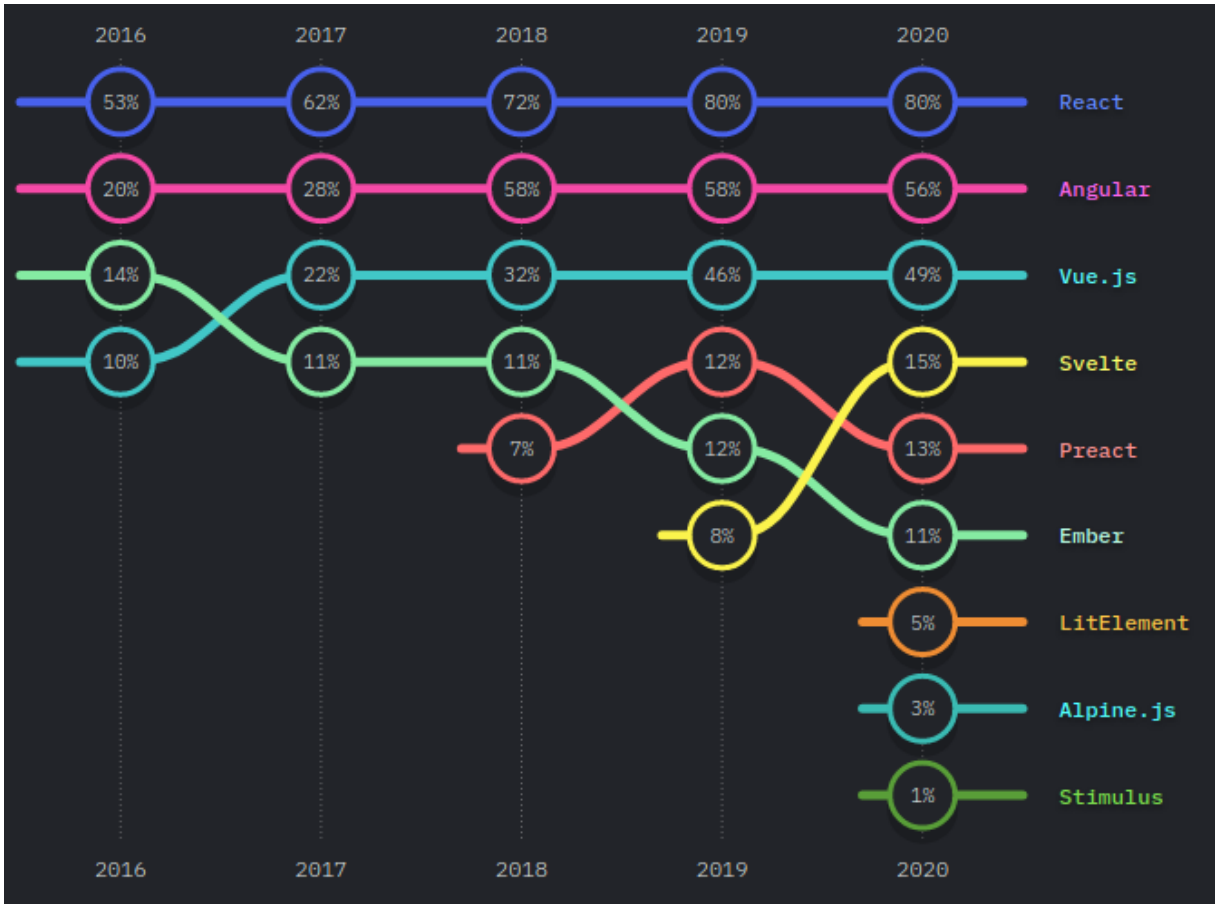

Figura 2.6: JS *front end frameworks* mais utilizados em 2020, segundo a pesquisa levantada pelo site State of JavaScript (2020)

Para entender um pouco do porque o ReactJs possui uma aceitação de 80% das pessoas já utilizaram do framework, são apresentados a seguir suas principais vantagens e funções são:

- *Open source*: é uma ferramenta que foi aberta aos princípios de colaboratividade, transparência e desenvolvimento orientado à comunidade, é uma iniciativa que a

torna acessível a todos e aberta a mudanças.

- *Single Page Application* (SPA): SPAs são sites que trazem uma experiência similar a uma aplicação desktop como, por exemplo, não possuir *reloads* ao trocar de página;
- Componentização: entidades que serão renderizados, modularizando e auxiliando na reutilização de código;
- Ciclo de vida: eventos que ocorrem durante a construção, atualização e desconstrução dos componentes, o que permite modificá-los;
- *Document Object Model* (DOM) virtual: representação ideal da interface do usuário, mantida em memória e sincronizada com o DOM, API para manipular documentos de linguagem de marcação;
- *JavaScript Sintaxe Extension* (JSX): é uma extensão de sintaxe do JS para facilitar a estruturação de componentes;
- Comunidade ativa: dado a sua grande utilização, há diversos fóruns, artigos e bibliotecas disponíveis relacionados ao ReactJs, facilitando seu aprendizado e desenvolvimento.

Produzir uma ferramenta utilizando apenas de JS puro consegue se tornar muito complexo rapidamente, para isso a utilização de *front-end frameworks* auxilia na otimização do tempo e até mesmo na qualidade de código. Para este trabalho o ReactJs foi utilizado para a construção da ferramenta The BOM Builder.

...

Este capítulo apresentou toda a fundamentação utilizada para atingir os objetivos citados anteriormente. No Capítulo 3 serão apresentados os métodos utilizados para o desenvolvimento dos mesmos.

3 Método

Este capítulo descreve o método utilizado neste trabalho de desenvolvimento de uma ferramenta para projeto de uma árvore de manufatura para jogos de sobrevivência.

Este trabalho possui a pesquisa do tipo exploratória, sendo apenas um insumo para auxílio no reconhecimento de melhores práticas, *frameworks* e processos de criação. A caracterização da pesquisa é um reflexo da sua utilidade, buscar conhecimentos já existentes para uma aplicação prática.

Foi buscado se familiarizar e formalizar conceitos de um gênero de jogo em um modelo utilizando de uma aplicação prática de técnicas e algoritmos já existentes. Especificamente, o algoritmo *Bill of Materials* (BOM), explicado no Algoritmo 2 na Seção 2.5, que serão empregados na ferramenta desenvolvida como fruto deste trabalho para projeto de árvores de manufatura em jogos de sobrevivência.

Adicionalmente, este trabalho apresentará o projeto e desenvolvimento de um sistema web, que consiste em uma interface rica para internet que utilizará dos algoritmos BOM para criar um ambiente de modelagem e simulação das árvores de manufatura. Através desse ambiente, jogos existentes serão implementados e simulados para experimentar o grau de representatividade de propostas na ferramenta gerada.

3.1 Etapas

Como fruto deste trabalho, foi feita uma ferramenta para auxiliar o *design* de jogos a balancear e testar a árvore de manufatura antes de sua implementação em um jogo real. Para atingir os objetivos dessa ferramenta, as seguintes etapas foram seguidas:

1. A construção de um corpo de conhecimento sobre o algoritmo BOM;
2. A implementação de algoritmos e modelos para representação dos BOM;
3. O projeto e implementação de um editor na forma de uma aplicação web que permita descrever a árvore de manufatura;

4. A criação de um simulador de estoques na ferramenta, permitindo simular operações de montagem dos itens;

5. Um modelo de contratos, que realizam a conversão direta de itens no estoque;

6. Avaliação da cobertura da ferramenta utilizando de modelos de jogos reais;

3.2 Modelo Conceitual

O sistema é dividido em cinco blocos principais de funcionalidades, como pode-se observar na Figura 3.1. Sendo o central, o bloco de funcionalidades da árvore de manufatura, responsável por modelar as relações entre itens, em quais quantidades de recursos são necessários para se criar uma unidade de um novo item produzido.

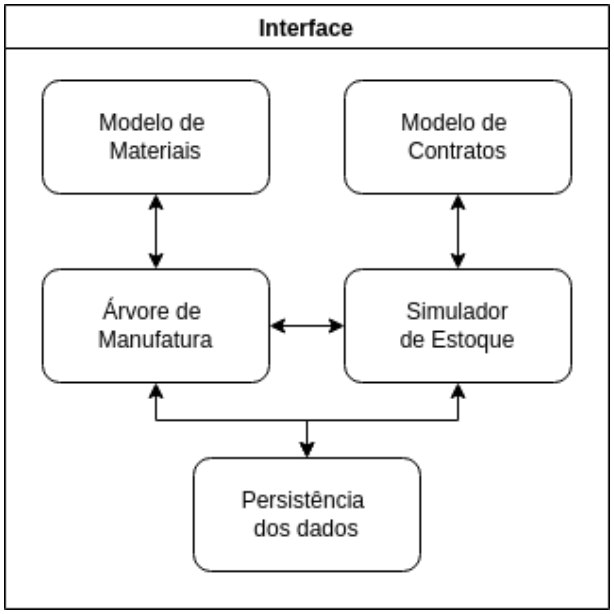

Figura 3.1: Diagrama contendo os módulos do The BOM Builder

O bloco de modelo de materiais é responsável pela implementação de algoritmos BOM necessários para calcular o valor total de recursos para criação de um material, expandindo a árvore das composições por camadas, que permitirá ao *designer* equilibrar o uso dos materiais no jogo.

O bloco simulador é responsável por criar uma instância dos estoques do jogo, representando as quantidades de cada material e permitindo ou não a criação de itens, criando um ambiente para explorar situações que o jogador poderia se encontrar.

O bloco de contratos é responsável por modelar conversões de itens, representando eventos e trocas no jogo. Ele também se conecta ao simulador para uso dos estoques com os itens atualmente disponíveis.

E o bloco persistência dos dados é para salvar e carregar o projeto da árvore do jogo. Todos os blocos foram implementados e podem ser visualizados através da interface da aplicação criada e permite que modelos sejam compartilhados.

O produto final é uma ferramenta com um editor de árvores de manufaturas e um laboratório para projeto e experimentação de jogos de sobrevivência, para se obter indícios de como sua economia e progresso se comportarão antes de o jogo ser de fato implementado.

...

Este capítulo apresentou o método a ser utilizado neste trabalho e a visão geral das funcionalidades a serem implementadas. No Capítulo 4 será apresentado o desenvolvimento detalhado das etapas descritas neste capítulo.

4 Desenvolvimento

Este capítulo descreve como foi desenvolvido o sistema *The BOM Builder*, sua arquitetura e a descrição de cada funcionalidade. Aqui serão apresentadas as interfaces, tecnologias e usabilidade da aplicação criada.

4.1 Modelagem da Árvore de Manufatura

A árvore de manufatura é onde pode-se criar, visualizar, editar e deletar os itens e suas relações. Para desenvolvimento dos algoritmos foram utilizados os dados mostrados na Figura 4.1, nela é possível observar a forma de armazenamento dos dados na ferramenta. Modelado na forma de grafo, os itens seriam equivalentes aos nós sendo representados no desenvolvimento da ferramenta como *nodes*, a conexão entre dependências e itens seriam as arestas representadas como *edges*, e os contratos são um formato diferente de definir conexões representado como *conversionEdges*.

Cada item possui como atributos o *label*, sendo o nome dado ao material, *amount*, sendo a quantidade armazenadas em estoque, e *layer*, chamado na aplicação de camada, sendo o nível ou altura do material na árvore de manufatura.

As conexões entre itens possuem uma estrutura simples quando se comparado aos demais, mas importante, seu Identificador Único (ID) é a junção dos IDs dos itens que se conectam, sendo o primeiro a origem da aresta e o segundo seu destino. O número referido à conexão é a quantidade necessária do primeiro material para fazer o segundo.

A modelagem dos contratos é composta de um *label*, nome opcional, um estado *available* que indica se a conversão pode ou não ocorrer conforme os itens em estoque, e duas listas que indicam os itens de origem (*sources*) e destino (*targets*) e suas respectivas quantidades.

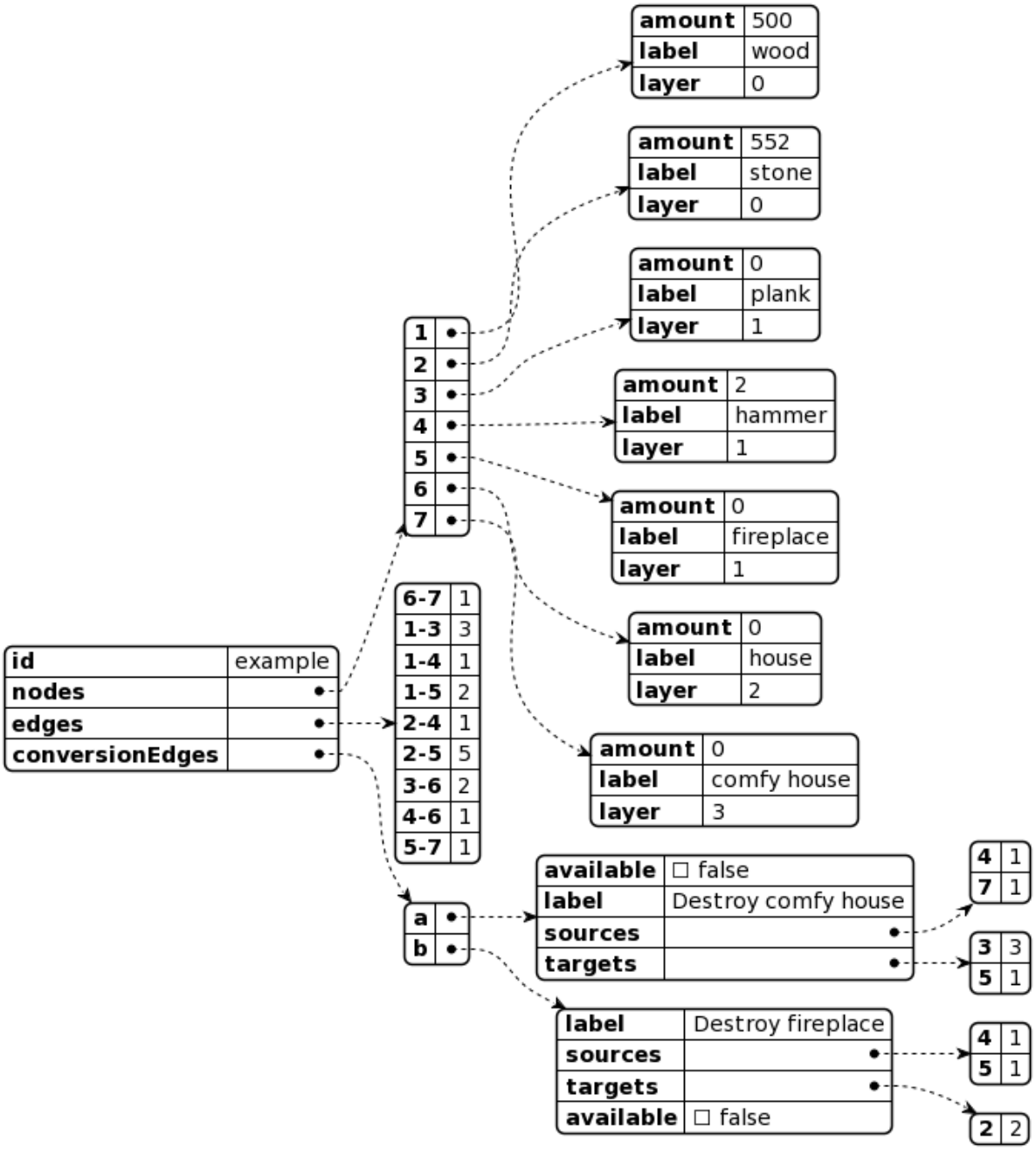

Figura 4.1: Dados utilizados para modelar uma versão simples do problema

4.2 Modelagem dos materiais

Para visualizar cada parte necessária para a construção de um item foi utilizado para seu cálculo uma versão adaptada do Algoritmo 2 apresentado na Seção 2.5, como é mostrado no Algoritmo 3.

O algoritmo recebe como primeiro parâmetro um vetor, *currentNodes*, com os itens utilizados para o cálculo da função. Como segundo recebe o *currentLayer*, equivalente à camada atual dos itens presentes no vetor, controlando a recursão da função, sendo

atualizada a cada iteração até chegar no valor do terceiro parâmetro, a camada desejada, *wantedLayer*. Os valores passados como parâmetro na primeira iteração da função é um vetor contendo o material a ser criado, a camada do mesmo e a camada desejada.

Algoritmo 3: Cálculo de materiais em camadas para um item escolhido

Entrada: currentNodes, currentLayer, wantedLayer
Saída: Vetor contendo os itens da camada desejada e o valor total necessário dos mesmo para criar o material passado na primeira iteração desta função

```
1  início
2      if currentlayer = wantedLayer then
3          return currentNodes
4      end if
5      i ← inicia com o valor 0;
6      nextNodes ← inicia com vetor vazio;
7      while i < currentNodes.length do
8          j ← inicia com o valor 0;
9          sourceEdges ← inicia com vetor contendo as arestas com destino
             igual ao currentNodes[i].id;
10         while j < sourceEdges.length do
11             amount ← sourceEdges[j].label * currentNodes[i].amount;
12             nextNodes ← nextNodes + o item de ID sourceEdges[j].source
                 com o amount atualizado;
13             j = j + 1;
14         end while
15         i = i + 1;
16     end while
17     return currentNodes + chamada recursiva a este algoritmo passando
         como parâmetro nextNodes, currentLayer - 1 e wantedLayer
18 fim
```

No Algoritmo 3, para cada nó passado no vetor *currentNodes*, é feito uma busca para captar as origens de suas conexões e elas são adicionadas ao vetor *nextNodes* com sua quantidade atualizada. A atualização da quantidade é feita pela multiplicação da quantidade do nó destino pela quantidade necessária do mesmo para criar o item destino. O resultado gerado pelo Algoritmo 3 é um vetor contendo todos os itens da camada desejada com sua quantidade total calculada. Um exemplo dessa aplicação será mostrado na Subseção 4.5.4.

4.3 Modelagem dos Contratos

Contratos são conexões realizadas diretamente no simulador entre os itens presentes na árvore de manufatura, eles permitem maior flexibilidade, normalmente relacionado a eventos no jogo. A diferença entre os contratos e conexões que ocorrem no grafo é a liberdade de, ao invés de gerar um único material, gerar um novo conjunto de itens.

Os contratos também permitem representar itens que atuam como catalisador: é necessário para produzir um material, mas ele não é consumido no processo (FRITZ, 2015). Um exemplo de catalisador, em um jogo de sobrevivência, é o uso de uma serra em um tronco de madeira bruta, ao final a serra ainda existe e o tronco foi refinado e transformado em uma tábua de madeira.

O estado *available*, mostrado na Figura 4.1, é definido pelo Algoritmo 4. Ele averigua se a quantidade dos materiais citados como pré-requisitos em *sources* são equivalentes aos valores necessários para permitir a conversão, trazendo como resultado o valor de *available*.

Algoritmo 4: Cálculo que identifica se o contrato está disponível para ser aplicado no simulador presente na ferramenta

Entrada: currentConversion, nodes
Saída: Booleano que define se o item está ou não disponível para construção

```
1 início
2     available ← inicia com o valor true;
3     for para cada source em currentConversion.sources do
4         if nodes[source.nodeId].amount for igual a source.value then
5             available ← false
6         end if
7     end for
8     return available
9 fim
```

4.4 Persistência dos dados

Criar uma árvore de manufatura pode levar um tempo considerável, pensando nisso foi criado a opção de salvar e carregar um projeto da árvore de manufatura. O usuário pode utilizar da importação através de arquivos no formato *JavaScript Object Notation* (JSON) ou através da inserção de um ID que busca os dados do projeto no banco de dados.

Os dados da árvore são armazenados em um banco de dados *DynamoDB* da *Amazon Web Services* (AWS), ele é um banco orientado a documento, escolhido devido a sua simplicidade e performance, além da aplicação ser de baixa escala. Para a manipulação do banco foi criado uma aplicação *back end* com o principal intuito de manter a segurança de dados sensíveis.

4.5 The BOM Builder

Este trabalho desenvolveu um *Minimum Viable Product* (MVP), na forma de uma aplicação web *Open Source*, utilizando a *MIT License* e disponível no GitHub [4], onde também se encontra um guia rápido de como contribuir. Além disso, a aplicação foi feita com suporte a diferentes línguas utilizando a biblioteca de internacionalização i18next (2011), já implementando uma tradução para inglês e português.

The BOM Builder é uma aplicação web desenvolvida utilizando como linguagem o TS, superconjunto tipado de JS, e utilizando a biblioteca *open source* ReactJS, que auxilia na criação de interfaces baseadas em componentização. Para facilitar a evolução do código, também foi aplicada uma configuração restrita do ESlint (2013), ferramenta de análise estática de código que auxilia na implementação de boas práticas para projetos JS procurando e resolvendo problemas no código-fonte.

4.5.1 Design da interface

Sites não são pensados apenas com atenção ao código, é necessário pensar na interação humana para que a aplicação seja visualmente atrativa e prática. Para isso, foi utilizado uma análise do conceito Gestalt, que implica em como a percepção da forma se aplica no site a ser desenvolvido, considerando a necessidade de simetria, facilidade, percepção de cores, entre outros (YıLMAZ; MUMCU, 2018). Esse estudo gerou o visual da aplicação deste trabalho, aplicado na ferramenta utilizando *Syntactically Awesome Style Sheets* (SASS), pré processador da linguagem de estilização CSS (SASS, 2006).

[4] O código da ferramenta The BOM Builder, fruto desse trabalho, está disponível no GitHub ⟨https://github.com/Caroliveira/bom-demonstration⟩

4.5.2 Árvore de Manufatura

A árvore de manufatura da ferramenta por ser visualizada na Figura 4.2. A representação gráfica foi possível pelo uso do conjunto de componentes ReactFlow[5]. Essa visualização é a que permite ao *designer* ter uma visão global das conexões e realizar suas alterações para alcançar a árvore desejada em seu jogo.

Outras bibliotecas foram utilizadas inicialmente para esse fim, como a D3.js (2011), Dagre-d3 (2013) e Mermaid (2015), mas o ReactFlow foi a que apresentou melhores resultados para criação e manipulação de grafos, devido à sua facilidade, boa documentação e funções necessárias. Porém, devido sua interação ser extremamente visual e não haver atalhos de teclado que funcionem para sua utilização, a biblioteca não é acessível para pessoas com deficiências visuais e isso é discutido na seção trabalhos futuros.

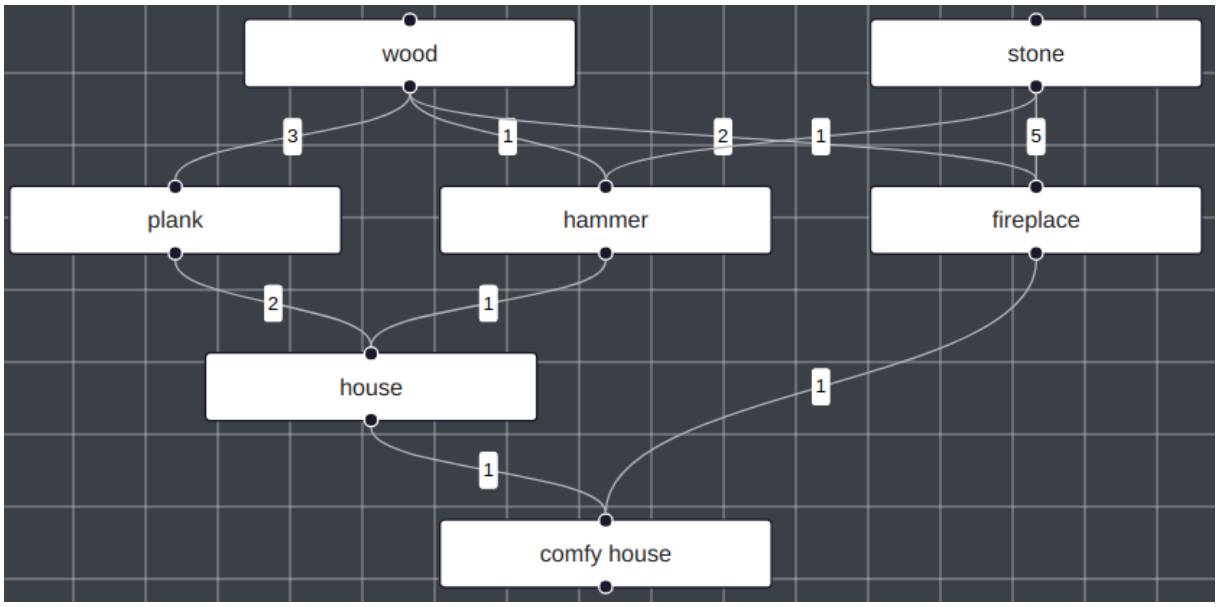

Figura 4.2: Visualização de uma árvore de manufatura simples utilizando The BOM Builder

4.5.3 Interface

Na página de criação do grafo, o usuário consegue criar novos itens referentes ao seu jogo ao clicar no botão com o sinal de mais. Após um item ser criado, é possível alterar seu nome ao clicar duas vezes na visualização do mesmo, direcionando para uma tela com mais

[5]ReactFlow é um *toolkit* para ReactJS para construção de redes. Disponível em ⟨https://reactflow.dev/⟩

informações do nó, entre elas a opção de editá-lo e deletá-lo. Ao adicionar um material, só é feita uma simples verificação relacionada ao nome do mesmo, o que garante a não existência de duplicatas ou nomes não preenchidos.

Os nós são apresentados no grafo com dois círculos para conexão, uma para receber a conexão e outro para iniciá-la. Quando dois ou mais itens são criados, é possível adicionar conexão entre eles, representada na Figura 4.3 a conexão onde é necessário um cipó para construir uma corda. Para criar essa conexão é necessário clicar no círculo de iniciação do item origem e segurar o clique até chegar no círculo de recebimento do item destino. Isso cria uma linha, representando a aresta de conexão, com o valor necessário para a construção.

(a) Visualização da conexão entre itens com orientação horizontal

(b) Visualização da conexão entre itens com orientação vertical

Figura 4.3: Capturas de tela da ferramenta para visualização de conexões em duas orientações: a) horizontal e b) vertical.

4.5.4 Visualização detalhada de materiais

Na interface da aplicação há a visualização da árvore de manufatura por inteiro, apresentada na subseção Árvore de Manufatura, e há a visualização de itens individualmente. Na Figura 4.2, ao dar dois cliques em um de seus itens, o projetista é direcionado para a página da Figura 4.4.

Essa interface permite navegar entre os dependentes do item, visualizar a informação de seus pré-requisitos por camadas, as conversões e eventos que constam o item

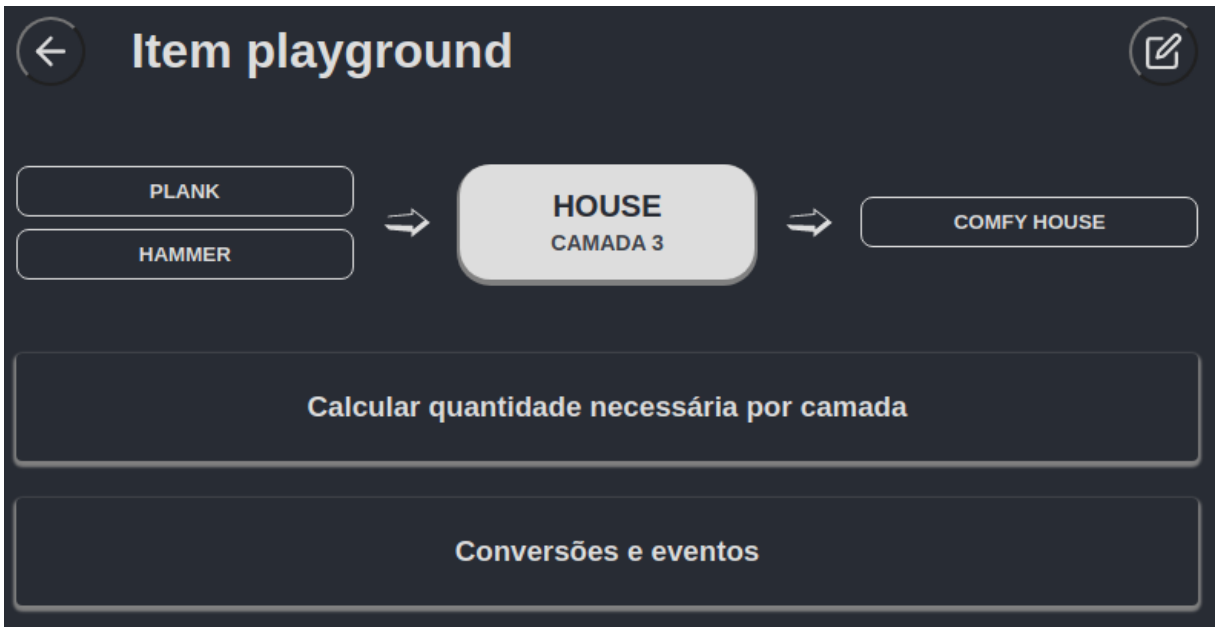

Figura 4.4: Captura da tela de um item após, no grafo, dar dois cliques no mesmo

em questão. A aba *Calcular quantidade necessária por camada*, é a representação visual da aplicação do algoritmo BOM, é possível selecionar a camada desejada e, à direita, é mostrado o resultado do cálculo de quantas unidades são necessárias de cada pré-requisito para criar o item desejado, como mostra a Figura 4.5.

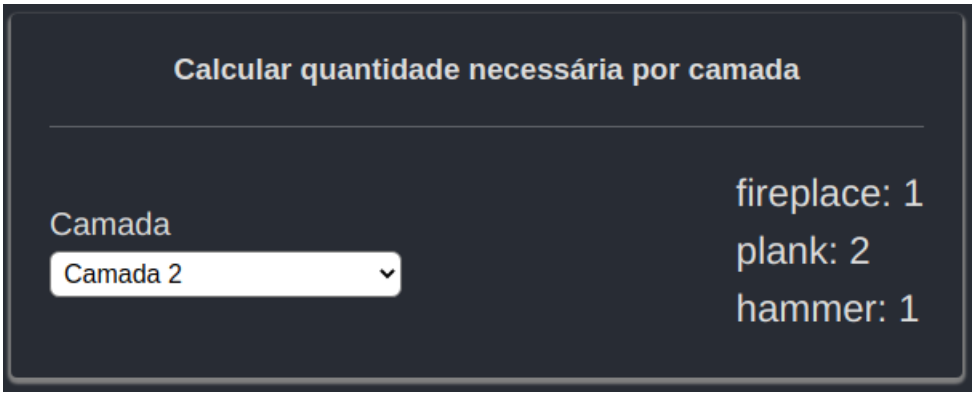

Figura 4.5: Resultado visual do cálculo feito pelo algoritmo BOM implementado na aplicação

4.5.5 Contratos

Na interface, apresentada na Figura 4.6, é possível observar a esquerda os itens de pré-requisito e a quantidade necessária de cada, a direita os itens que serão gerados pela conversão e em baixo mostra qual será o resultado da conversão e a diferenciação entre o estado anterior e posterior, apontando quantos itens foram gastos e quantos foram ganhos.

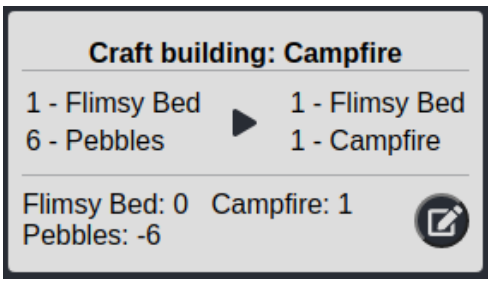

Figura 4.6: Captura da tela da visualização de uma conversão

Na figura, é mostrado um exemplo de bloqueio de itens, onde para produzir uma *Campfire* é necessário existir uma *Flimsy Bed*. No resultado é feito uma operação em cima de um histograma item a item, mostrando que a quantia de *Flimsy Bed* não será alterada, enquanto seis *Pebbles* seriam gastos para a produção de uma *Campfire*. Essa operação auxilia a ter uma melhor visão do resultado da conversão, principalmente em jogos com conversões complexas e números de maior escala.

4.5.6 Simulador

Na aplicação há uma página dedicada ao simulador, onde o usuário consegue visualizar os mesmos dados já observados no grafo, também sendo organizado por camadas, e consegue visualizar as conversões, ordenadas por disponibilidade. O importante do simulador é a interação que ele oferece ao *designer*, permitindo simular o jogo ao ir clicando nos itens e conversões disponíveis.

Na Figura 4.7, é possível observar o funcionamento do simulador. Nele os materiais e contratos disponíveis são apresentados na cor branca e os indisponíveis em cinza sem a possibilidade de clique. A disponibilidade é definida a partir dos pré-requisitos do item e se eles possuem a quantidade necessária para formá-lo.

O simulador é a forma de validar a árvore de tecnologia criada anteriormente, onde o *game designer* consegue testar diferentes cenários e averiguar se o balanceamento do jogo foi feito da forma desejada.

...

Este capítulo apresentou o que e como foi desenvolvido o fruto deste trabalho.

Figura 4.7: Captura da tela do simulador

No Capítulo 5, será feita uma avaliação da capacidade de representação da ferramenta criada.

5 Avaliação da Capacidade de Representação

Para observar a capacidade de representação da aplicação The BOM Builder, dois jogos de sobrevivência foram modelados. Os jogos escolhidos possuem uma jogabilidade diversa e trazem diferentes perspectivas em suas árvores de tecnologia. São eles, Don't Starve e The Survivalists.

5.1 Don't Starve

Don't Starve, desenvolvido e publicado pela Klei Entertainment Inc (KLEI, 2013), é um jogo para um único jogador, onde o personagem chega em um mundo criado proceduralmente cheio de animais e monstros, onde o único objetivo é sobreviver. O personagem começa coletando recursos simples como gravetos e pedras, e precisa criar itens e construções para manter a alimentação, vida e sanidade altas.

Duas características marcantes do jogo são: a existência de Charlie, uma criatura que se esconde nas sombras da noite, adicionando uma restrição de sempre manter uma fonte de luz próxima e; ser *permadeath*, ou seja, se seu personagem morrer é definitivo e um novo personagem deve ser criado. Isso adiciona um impacto maior nas decisões do jogador e muda completamente a experiência, deixando-a mais próxima da base da pirâmide de Maslow.

A Figura 5.1 apresenta duas capturas de tela do jogo, nas quais é possível observar na Figura5.1a, a interface do jogo para criação dos itens da árvore de manufatura, nela é visível que para construir uma *campfire* são necessários três *cut grass* e duas *logs*. E na Figura 5.1b, o personagem em seu acampamento onde todos os itens foram feitos a partir da coleta e construção de materiais.

A modelagem de itens do jogo é mais direta, não havendo trocas ou eventos, todos os itens são criados diretamente, como mostrado no modelo parcial na Figura 5.2.

(a) Captura de tela com foco nas abas de construção de itens do Don't Starve

(b) Captura da tela da base inicial de um personagem no jogo

Figura 5.1: Telas do jogo Don't Starve de: a) abas de construção e b) base.

Dessa forma a árvore de tecnologias consegue ser representada por inteiro no diagrama.

Foi percebido que a ferramenta carece em captar a durabilidade dos itens, pois o modelo deste trabalho é representado por estoque e não há uma representação de itens individualmente para apontar se estão ou não gastos. No Don't Starve, itens, construções e alimentos possuem tempo de vida que podem ser alterados dado as circunstâncias do jogo, como, por exemplo, uma comida leva mais tempo para estragar se estiver na *Ice Box* (geladeira), e isso não é mapeado pela ferramenta em questão.

Outro fator não representado pela ferramenta, é a opção de randomização na quantidade de itens a serem gerados por conversões. No jogo há situações onde itens possuem uma porcentagem de chance de serem gerados como, por exemplo, na Figura 5.3, possui um contrato chamado "Destroy tent" para destruir uma tenda, porém a tenda possui a possibilidade de usos e a quantidade de materiais gerados ao destruí-la podem variar.

O jogo permite recuperar parte dos materiais utilizados ao desmontar uma construção com um martelo. Para toda construção do jogo que possui quantidade máxima de usos, a quantia de itens retornados ao desmontá-la varia conforme a Equação 5.1 [6].

[6]Resultado ao martelar estruturas que possuem quantidade máxima de usos, retirado da wiki do jogo ⟨https://dontstarve.fandom.com/wiki/Hammer⟩

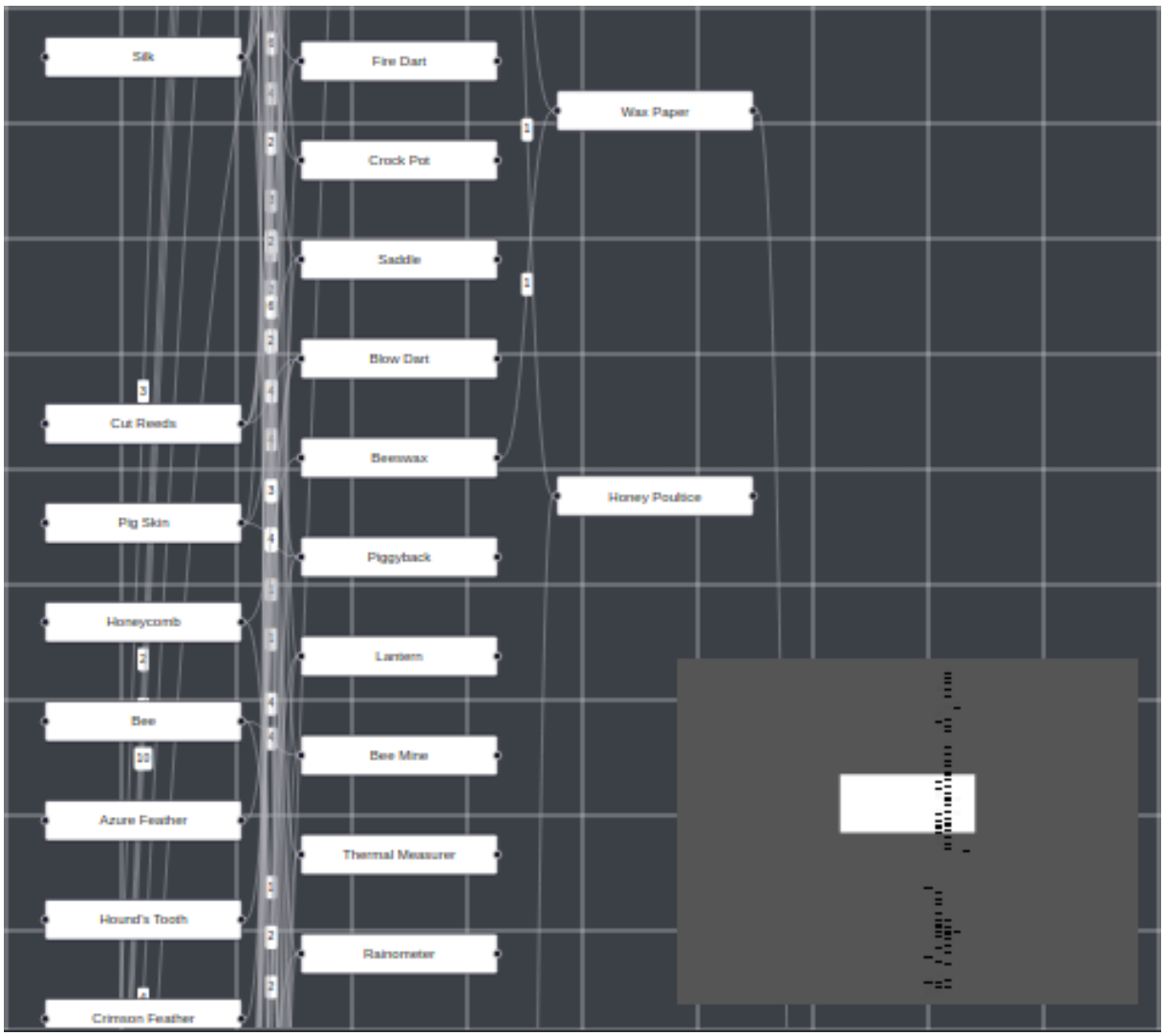

Figura 5.2: Diagrama parcial do jogo Don't Starve, o tamanho completo do diagrama pode ser observado no canto inferior direito

$$amount\_returned = original\_amount * percent\_uses\_left * 0.5 \tag{5.1}$$

5.2 Survivalists

The Survivalists, desenvolvido e publicado pela Team17 (2020), apesar de também ser um jogo de sobrevivência, tem uma estrutura muito diferente de Don't Starve, visto que possui objetivos além da sobrevivência. Ele é um jogo que pode ser jogado só ou com amigos, nele o personagem pode renascer e há interação com NPCs.

O diferencial do jogo é a utilização de macacos que podem ser encontrados

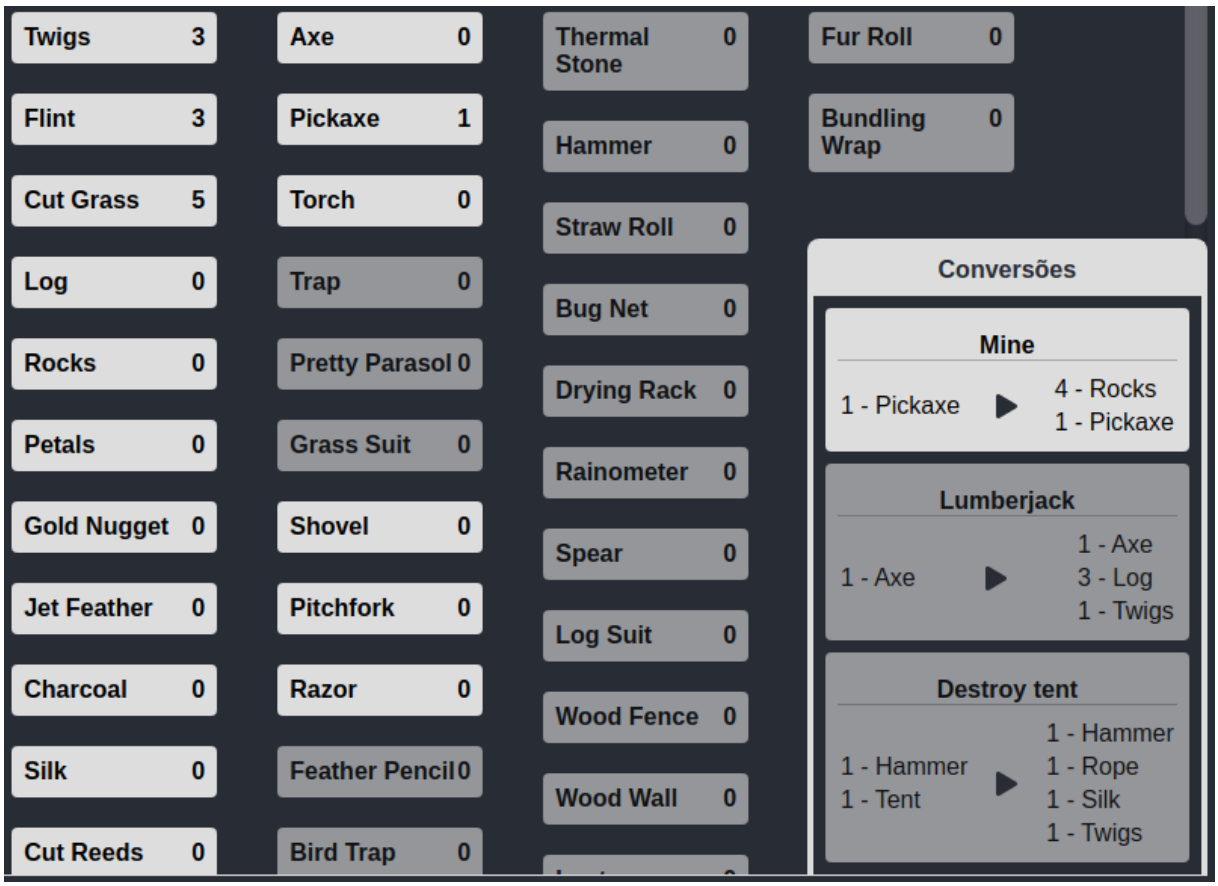

Figura 5.3: Simulador apresentando árvore parcial do jogo Don't Starve e algumas conversões possíveis

durante o jogo para auxiliar em tarefas, como cortar madeira, coletar materiais, construir, carregar baús ou até mesmo lutar ao seu lado. Essa característica permite ao jogador automatizar processos de extração e coleta, podendo focar em outros aspectos do jogo. Isso permite ao *game designer* criar uma árvore de manufatura com custos maiores, inclusive forçando o jogador a usar esses mecanismos, diferentemente de um jogo de sobrevivência tradicional. A Figura 5.4 apresenta duas capturas de tela do jogo em que é possível observar o personagem em seu acampamento e a interface de manufatura dos itens.

Esse jogo possui uma árvore de manufatura mais complexa e que demanda uma maior quantidade de conversões, como pode ser observado na Figura 5.5. A ferramenta em questão permite a demonstração da árvore no diagrama, mostrada na Figura 5.6, porém não possui um temporizador ou algo que simule a existência de automação, que deveria acontecer através dos macacos do jogo.

(a) Captura da tela da aba de *crafting* da mesa de construções

(b) Captura da tela do acampamento

Figura 5.4: Telas do jogo The Survivalists: a) abas de itens construídos através da bancada de criação e b) acampamento.

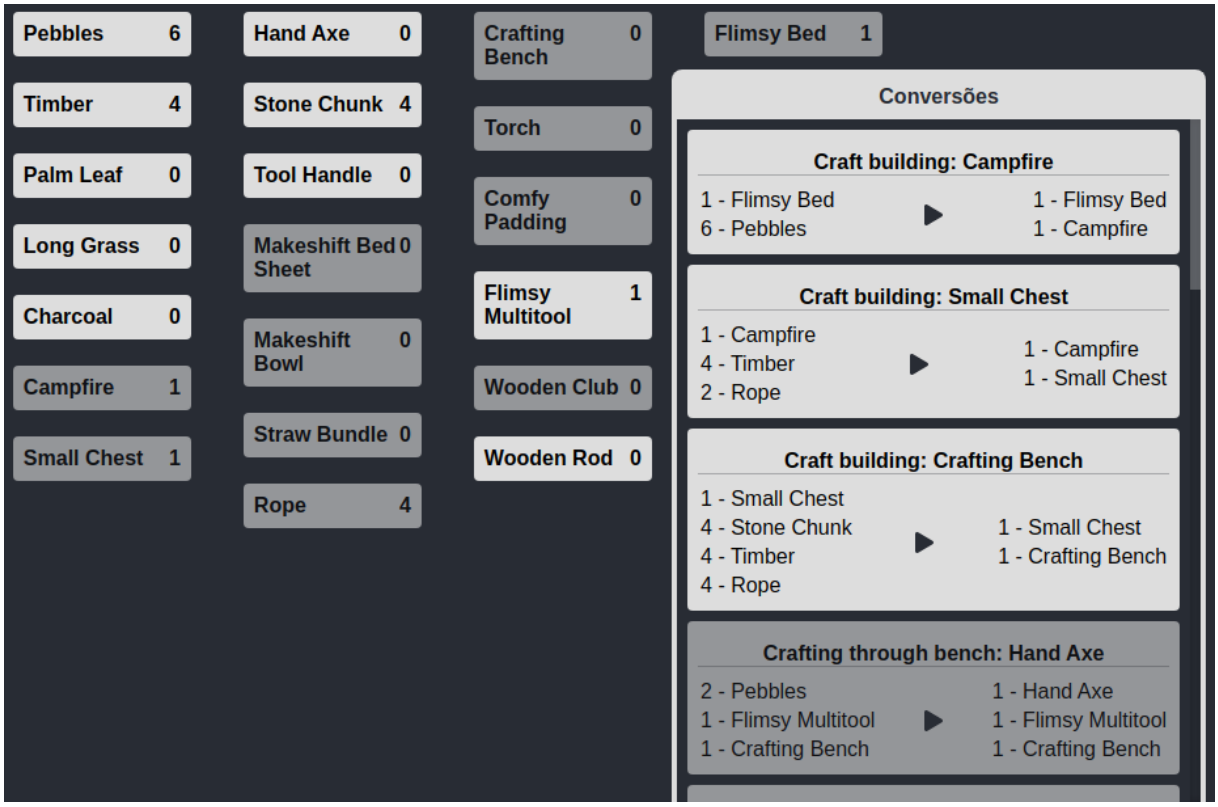

Figura 5.5: Simulador apresentando árvore parcial do jogo The Survivalists e todas as conversões possíveis envolvendo a *craft bench*

5.3 Avaliação

A ferramenta The BOM Builder é um MVP que permite a projeção, representação e simulação de árvores de manufaturas. A representação é feita através de um grafo que

Figura 5.6: Diagrama do jogo The Survivalists contendo a árvore manual e itens relacionados à bancada de construções

possui uma boa estrutura e funcionalidades desejadas, mas não é escalável. É possível perceber na Figura 5.2 que, devido à quantidade de itens e conexões, o grafo se torna confuso (e até ilegível em alguns momentos), com isso a ferramenta precisaria de novas funcionalidades para apresentação parcial da árvore de manufatura.

Além disso, o simulador presente na ferramenta não possui a individualização de itens, não permitindo a representação de tempo de vida dos materiais ou construções. Apesar disso, junto dos contratos, o simulador cumpre seu papel de conseguir experimentar cenários do jogo antes de sua implementação.

Por fim, a ferramenta cumpre seu papel base. Como um MVP, ela precisa ser melhorada para representar porções maiores das árvores de manufatura e possui algumas limitações. A interface do sistema segue parcialmente os conceitos de Gestalt, sendo

necessário melhorar principalmente a parte de percepção de cores utilizadas no site.

...

Este capítulo apresentou uma avaliação da capacidade de representação da ferramenta através da inserção de dois jogos existentes. No Capítulo 6, serão apresentados as conclusões do desenvolvimento deste trabalho.

6 Considerações Finais

Esse trabalho de conclusão de curso consistiu no desenvolvimento do sistema chamado The BOM Builder, sendo uma aplicação web com a funcionalidade de criar, editar e visualizar árvores de manufatura para jogos. Nessa aplicação web, foi criado um simulador de estoques e operações de montagem, onde o projetista consegue simular situações diversas que poderiam acontecer no jogo.

Nessa simulação, foi implementada a possibilidade de criar contratos, que servem para mapear eventos, catalisadores, bloqueio de itens ou representar a criação de um mesmo item de formas diferentes.

Contratos realizam a conversão direta de itens em estoque e não são representados pelo diagrama. Com eles, a possibilidade para simulações é estendida, trazendo mais flexibilidade para representações e simulações. Para validar a proposta foram utilizadas duas árvores de jogos já existentes, onde foi possível observar que a ferramenta atingiu o objetivo de representação das árvores de jogos.

Ao observar o jogo Don't Starve em contato com a ferramenta, foi possível avaliar que a ferramenta carece em captar dados que precisam variar a quantidade de itens gerados de acordo com algumas equações ou probabilidade e carece em representar durabilidade de materiais, visto que é trabalhada a partir de estoques não existindo a individualização dos itens.

O jogo The Survivalists, possui a possibilidade de automação da produção, permitindo uma árvore de manufatura mais complexa. Em contato com a ferramenta, não foi possível representar essa automação, falhando em captar a vantagem e velocidade de produção de itens através dos macacos.

6.1 Limitações e Trabalhos Futuros

Temos indícios iniciais que o The BOM Builder pode permitir a desenvolvedores a criarem e balancearem as árvores de tecnologia dos seus jogos, porém ainda é um trabalho em

progresso, sendo necessário criar novas funções e melhorar funções já existentes.

Um dos trabalhos futuros é permitir mais possibilidades de visualização no grafo. Como o grafo é uma parte importante para a visualização macro da árvore de manufaturas, também é interessante adicionar a visualização de subárvores e de todas as conexões entre itens, inclusive os contratos.

Também é a intenção acrescentar outras características aos itens, como unidades de tempo e a duração, visto que em um jogo um item pode ser utilizado diversas vezes antes que acabe, por exemplo, um machado poderia ser utilizado cinco vezes para cortar madeira antes de quebrar.

6.1.1 Simulador

Uma funcionalidade inicialmente planejada, mas que não foi implementada, é adaptar o simulador para que a simulação possa ser automatizada, utilizando a mesma lógica de definição da quantidade dos itens, para torná-lo similar a uma instância de um *idle game*, jogo incremental, trazendo maior visibilidade de como seria o funcionamento da árvore no jogo real.

Portanto, como um possível futuro trabalho, será necessário adicionar ao programa informações como a duração de itens e cálculo da usabilidade dos mesmos. Será necessário também trazer, não só uma refatoração dos algoritmos, como também da tela em questão, pois essa aplicação impacta visualmente e são muitos detalhes para serem mostrados, devendo haver maior tato para não torná-la uma página confusa para o usuário.

6.1.2 Acessibilidade

Tornar a aplicação acessível é de extrema importância, apesar de algumas diretrizes terem sido seguidas, ainda há muito a adicionar na ferramenta em questão. Sendo necessário melhoras no grafo, visto que as linhas de conexão tem a mesma cor das linhas de fundo e que muitas das alterações são feitas por cliques. Para isso, seria necessária a troca de cores utilizadas e a possibilidades de busca dos nós individualmente por nome.

Além disso, os modais existentes na aplicação não funcionam corretamente com o leitor de tela, pois é possível navegar na página abaixo sem fechá-lo. Para isso é

necessário adicionar uma configuração para que a tela abaixo seja ignorada.

Por fim, a ferramenta atualmente só possui capacidade de representação da interface correta por computadores e, como possível trabalho futuro, seria interessante adaptar a interface da ferramenta para visualização *mobile*. Tornando a navegação na ferramenta possível de qualquer dispositivo.

Bibliografia

ALMEIDA, I. X. de; SCHELSKE, F. L.; ROVER, A. Percepção dos fatores motivacionais de maslow no contexto organizacional. *Unoesc amp; Ciência - ACSA*, v. 10, n. 1, p. 37–44, jun. 2019. Disponível em: ⟨https://portalperiodicos.unoesc.edu.br/acsa/article/view/15915⟩.

ANGULAR. *Documentação do Angular*. 2016. Disponível em: ⟨hhttps://angular.io/⟩.

ARAúJO, J. M. F. JOGOS ELETRÔNICOS: INFLUÊNCIAS POSITIVAS E NEGATIVAS DOS GAMES EM MEIO A SOCIEDADE. *Brasil Escola*, 2020. Disponível em: ⟨https://meuartigo.brasilescola.uol.com.br/atualidades/jogos-eletronicos-influencias-positivas-e-negativas-dos-games-em-meio-a-sociedade.htm⟩.

BALLHAUS, W. C. W. Pulling the future forward: The entertainment and media industry reconfigures amid recovery. *PWC*, 2020. Disponível em: ⟨https://www.pwc.com/gx/en/entertainment-media/outlook-2020/perspectives.pdf⟩.

BARR, M.; COPELAND-STEWART, A. Playing video games during the covid-19 pandemic and effects on players' well-being. *Games and Culture*, v. 17, n. 1, p. 122–139, 2022. Disponível em: ⟨https://doi.org/10.1177/15554120211017036⟩.

CARMODY, S. et al. Building resilient medical technology supply chains with a software bill of materials. *npj Digital Medicine*, 2021.

CHACON, A. Artigo: A influência comportamental dos jogos eletrônicos. *Fábrica de jogos*, 2015. Disponível em: ⟨https://www.fabricadejogos.net/posts/artigo-influencia-comportamental-dos-jogos-eletronicos/⟩.

CINELLI, M. et al. A network perspective for the analysis of bill of material. *Procedia CIRP*, v. 88, p. 19–24, 01 2020.

CLEMENT, J. Number of games released on Steam worldwide from 2004 to 2021. *Statista*, 2021. Disponível em: ⟨https://www.statista.com/statistics/552623/number-games-released-steam/⟩.

D3.JS. *Documentação da biblioteca Data-Driven Documents*. 2011. Disponível em: ⟨https://d3js.org/⟩.

DAGRE-D3. *Documentação da biblioteca Dagre-d3*. 2013. Disponível em: ⟨https://github.com/dagrejs/dagre-d3/wiki⟩.

DASA. *Lockdown durante a pandemia do Coronavírus: o que é e quais países adotaram*. 2021. Disponível em: ⟨https://dasa.com.br/blog/coronavirus/lockdown-coronavirus-significado/⟩.

ESLINT. *Documentação do ESlint*. 2013. Disponível em: ⟨https://eslint.org/⟩.

FEITOSA, M.; SOUZA, S. Características qualitativas para desenvolvimento de jogos de sobrevivência. *Research, Society and Development*, v. 7, p. 579383, 05 2018.

FRITZ, T. Resource Convertibility (Part 2). 2015. Disponível em: ⟨https://johncarlosbaez.wordpress.com/2015/04/10/resource-convertibility-part-2/⟩.

GANDRA, A. Mercado de games no Brasil deve crescer 5,3% até 2022. *Agência Brasil*, 2019. Disponível em: ⟨https://agenciabrasil.ebc.com.br/geral/noticia/2019-08/mercado-de-games-no-brasil-deve-crescer-53-ate-2022-diz-estudo⟩.

GRAINER, S. *Maslow's Hierarchy of Game Design*. 2013. Disponível em: ⟨https://www.gameskinny.com/xflt8/maslows-hierarchy-of-game-design⟩.

HE, L. et al. Integration of bill of materials with unified bill of materials model for commercial aircraft design to manufacturing. *Concurrent Engineering*, v. 22, p. 206–217, 08 2014.

HUNICKE, R.; LEBLANC, M.; ZUBEK, R. Mda: A formal approach to game design and game research. *AAAI Workshop - Technical Report*, v. 1, 01 2004.

I18NEXT. *Documentação da biblioteca de internacionalização i18next*. 2011. Disponível em: ⟨https://www.i18next.com/⟩.

JAVASCRIPT, S. of. Front-end Frameworks. 2020. Disponível em: ⟨https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/⟩.

JONES, A. This gaming genre is gaining the most popularity in 2021. *Invision Game Community*, 2021. Disponível em: ⟨https://invisioncommunity.co.uk/this-gaming-genre-is-gaining-the-most-popularity-in-2021/⟩.

KAZEROONI, M.; KAZEROONI, A. An activity based bill of materials, a new concept for production planning. 2004. Disponível em: ⟨https://www.academia.edu/5575417/An_Activity_Based_Bill_of_Materials_a_New_Concept_for_Production_Planning⟩.

KELLY, S.; NARDI, B. Playing with sustainability: Using video games to simulate futures of scarcity. *First Monday*, v. 19, n. 5, Apr. 2014. Disponível em: ⟨https://firstmonday.org/ojs/index.php/fm/article/view/5259⟩.

KHALAILA, A.; ELIASSEN, F.; BEERI, C. Efficient bill-of-materials algorithms. 01 1997.

KLEI. *Site oficial do jogo Don't Starve*. 2013. Disponível em: ⟨https://www.klei.com/games/dont-starve⟩.

LEITE, L. C. *JOGOS ELETRÔNICOS MULTI-PLATAFORMA*. Tese (Doutorado) — PUC-Rio, 2006.

MASTERCLASS. *Guide to Video Game Genres: 10 Popular Video Game Types*. 2020. Disponível em: ⟨https://www.masterclass.com/articles/guide-to-video-game-genres#10-popular-video-game-genres⟩.

MDN. *Web technology for developers*. 2022. Disponível em: ⟨https://developer.mozilla.org/en-US/docs/Web⟩.

MERMAID. *Documentação da biblioteca Mermaid*. 2015. Disponível em: ⟨https://mermaid-js.github.io/mermaid/#/⟩.

MICROSOFT. *Characteristics of Modern Web Applications*. 2021. Disponível em: ⟨https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/modern-web-applications-characteristics⟩.

MILLéO, A. *COVID-19: tudo sobre o novo coronavírus. Exames, sintomas, tratamentos, transmissão etc.* 2020. Disponível em: ⟨https://vidasaudavel.einstein.br/coronavirus/covid-19-faq/⟩.

PINTO, R. D.; FERREIRA, L. F. Ciência do comportamento e aprendizado através de jogos eletrônicos. *Seminário Jogos Eletrônicos, Educação e Comunicação*, v. 1, p. 1–14, 2005.

REACTJS. *Documentação do ReactJS*. 2013. Disponível em: ⟨https://reactjs.org/⟩.

ROGERS, S. *Level Up! The Guide to Great Video Game Design*. [S.l.]: John Wiley & Sons, 2010.

SASS. *Documentação do SASS*. 2006. Disponível em: ⟨https://sass-lang.com/⟩.

SCHELL, J. *The Art of Game Design: A book of lenses*. [S.l.]: Morgan Kaufmann, 2008.

STAPIC, Z.; OREHOVAčKI, T.; LOVRENCIC, A. In search of an improved bom and mrp algorithm. In: . [S.l.: s.n.], 2009.

STEAM. Comparing online charts for 10 apps. *SteamDB*, 2021. Disponível em: ⟨https://steamdb.info/graph/?compare=105600,108600,221100,242760,251570,252490,304930,322330,346110,892970⟩.

STEAM. TOP SELLERS. *Steam*, 2021. Disponível em: ⟨https://store.steampowered.com/search/?sort_by=Reviews_DESC&filter=topsellers⟩.

TEAM17. *Site oficial do jogo The Survivalists*. 2020. Disponível em: ⟨https://www.team17.com/games/the-survivalists-2/⟩.

TOH, W. The economics of decision-making in video games. 2004. Disponível em: ⟨http://gamestudies.org/2103/articles/toh⟩.

TYPESCRIPT. *Documentação oficial do TypeScript*. 2012. Disponível em: ⟨https://www.typescriptlang.org/⟩.

UNITY. COVID-19's Impact on the Gaming Industry: 19 Takeaways. *Unity*, 2020. Disponível em: ⟨https://create.unity3d.com/COVID-19s-impact-on-the-gaming-industry⟩.

VISA. Dados da Visa mostram crescimento de quase 140% no faturamento do mercado de games no Brasil. *Visa*, 2021. Disponível em: ⟨https://www.visa.com.br/sobre-a-visa/noticias-visa/nova-sala-de-imprensa/mercado-de-games-brasil.html⟩.

VUE.JS. *Documentação do Vue.js*. 2014. Disponível em: ⟨https://vuejs.org/⟩.

YıLMAZ, S.; MUMCU, S. Application of gestalt principles in planting design. In: _____. [S.l.: s.n.], 2018. ISBN 1-5275-1087-5.