Interface de Usuário para Controle de Geração Procedural com Wave Function Collapse

Wellington B. Almeida[1], **Geber L. Ramalho** [1]

[1]Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50.732-970 – Recife – PE – Brasil

{wba,glr}@cin.ufpe.br

Abstract. *This work describes a User Interface (UI) enabling the exploration of image generators through tile combinations using the Wave Function Collapse (WFC) algorithm. The aim is to enhance Procedural Content Generation (PCG) intelligibility and controllability for a broad audience, addressing the scarcity of graphical interfaces providing control over PCG techniques. To achieve this, a high-level interface was developed, implemented through a client-server architecture. The client, a web page, manages dynamic tile registration, while the server executes the algorithm, returning procedurally generated images. User experiments suggest that WFC can be controlled via the UI, allowing users to explore PCG in tilemaps. However, the graphical interface faces challenges concerning user experience, primarily in terms of clarity and attractiveness.*

Resumo. *Este trabalho descreve uma Interface de Usuário (UI) que permite explorar geradores de mapas de tiles utilizando o Wave Function Collapse (WFC). O objetivo por trás desse sistema é tornar o uso de Geração Procedural de Conteúdo (GPC) mais controlável e inteligível para um público amplo, contribuindo para resolver o problema da escassez de interfaces gráficas para GPC. Para fazer isso, foi desenvolvida uma interface de alto nível que interage com o algoritmo. Ela foi implementada por meio de uma arquitetura cliente-servidor, em que o cliente é uma página web responsável pelo cadastro dinâmico de tiles, enquanto o servidor executa o algoritmo e retorna os resultados. Os experimentos realizados com usuários sugerem que o WFC pode ser controlado por meio da UI, permitindo a exploração da GPC de mapas de tiles. No entanto, a interface gráfica proposta apresenta desafios de experiência de usuário, especialmente no que diz respeito à clareza e atratividade.*

1. Introdução

A produção de conteúdo para jogos contemporâneos representa um desafio significativo em termos de custos para os estúdios, especialmente no contexto dos jogos triplo A. Esses projetos exigem uma equipe substancial de profissionais e vários anos de desenvolvimento. Por exemplo, o desenvolvimento do jogo *Cyberpunk 2077*, lançado no final de 2020, implicou um investimento de cerca de 181 milhões de dólares (sem considerar os gastos com marketing) e envolveu uma equipe de cinco mil pessoas durante quatro anos de trabalho [Kiciński et al. 2020].

Na maioria dos casos, os elementos de produção mais custosos são os criados pelas equipes de design e arte, como terrenos, mapas, estórias, diálogos, missões e itens,

conhecidos como *game content*. Isso se deve principalmente ao fato de que esse tipo de conteúdo precisa ser desenvolvido individualmente para cada projeto. Além disso, em comparação com as equipes técnicas, não há uma quantidade significativa de *middlewares* disponíveis para facilitar o trabalho dessas equipes [Togelius et al. 2011].

A Geração Procedural de Conteúdo (GPC) é definida como a criação algorítmica de conteúdo com entrada limitada ou indireta do usuário [Hendrikx et al. 2013]. Nesse contexto, a GPC pode ser utilizada como alternativa para atenuar o custo de produção de *game content* [Shaker et al. 2016]. Mas não só isso, GPC pode apresentar outras vantagens. O valor de *replay*, ou rejogabilidade, de um jogo é aumentado drasticamente quando utiliza-se GPC [Compton and Mateas 2006]. Além do mais, ela permite o desenvolvimento de jogos praticamente impossíveis de serem feitos manualmente [Short and Adams 2017], como é o caso de No Man's Sky [Games 2016], onde a escala de mundo e variedade de detalhes do jogo extrapolam a capacidade de criação humana. Visto isso, GPC é amplamente utilizada na indústria de jogos, possuindo por volta de 3400 jogos marcados com a *tag* "Geração Procedural" na *Steam*[1] na data deste artigo.

Contudo, exitem barreiras que dificultam o acesso a GPC. [Togelius et al. 2013] apontam que a maioria dos sistemas de GPC não é de fácil interação e controle, o que se configura como um dos principais desafios no campo de GPC. Este obstáculo precisa ser superado para que ocorram avanços significativos tanto na comunidade acadêmica quanto na indústria de jogos. Esse problema acontece porque há escassez de algoritmos geradores com opções de parâmetros que permitam aos usuários alterar qualquer aspecto do conteúdo criado durante o processo de geração. Mesmo quando o algoritmo possibilita algum nível de controle, a falta de interfaces gráficas adequadas para interagir com ele é notável.

Mas como deve ser uma interface gráfica para a criação automática de conteúdo? A resposta a esta pergunta transcende o campo da GPC, sendo, na verdade, um amplo desafio de interface humano-computador. Esse desafio é consequência das tecnologias emergentes de criação automática de conteúdo. Por exemplo, Mello et al. destacam a carência de um controle detalhado sobre as imagens geradas por meio de Inteligência Artificial (IA) através de uma interface textual [Mello et al. 2023].

A co-criação (ou autoria mista) de conteúdo procedural é outra vantagem que uma interface de GPC controlável pode proporcionar. Pois, como apontado por [Yannakakis et al. 2014], quando tanto o humano como o computador fazem contribuições proativas para solução de um problema, a co-criação possui a capacidade de fomentar a criatividade humana. Além disso, possibilitaria aos criadores de *game content* (artistas e designers) acesso aos benefícios de produção oriundos do uso de GPC, principalmente para os usuários sem conhecimentos técnicos em desenvolvimento de software.

Em ótica das limitações de controle existentes no campo de GPC, este trabalho propõem uma Interface de Usuário (UI) de co-criação de *game content*, permitindo o controle de alto nível sobre um algoritmo gerador. Especificamente, este projeto concentrou-se na criação de mapas de tiles usando um algoritmo baseado em busca, o *Wave Function Collapse* (WFC). Com isso, o objetivo é tornar o WFC inteligível para um público mais amplo através da UI proposta.

[1] `https://store.steampowered.com/tag/browse/#global_5125`

Na seção subsequente, abordaremos outros trabalhos que também propuseram alguma forma de controle sobre conteúdo procedural e destacaremos as diferenças em relação à proposta deste artigo. Em seguida, na seção dedicada ao algoritmo WFC, apresentaremos os fundamentos teóricos que sustentam a lógica subjacente à UI implementada. Posteriormente, na seção de descrição da ferramenta desenvolvida, exibiremos suas funcionalidades, arquitetura e interface. Por fim, relataremos e analisaremos os resultados dos testes com os usuários da UI, avaliando aspectos de usabilidade e experiência deles.

2. Trabalhos Relacionados

A busca por abordagens que combinem a expertise humana com a potência dos algoritmos de GPC tem sido uma área de crescente interesse. No que diz respeito à geração automática de mapas de tiles, o uso de cadeias de Markov é bastante popular. Por exemplo, [Snodgrass and Ontañón 2014] propuseram um método de geração procedural de mapas baseados em tiles utilizando cadeias de Markov hierárquicas. Em seus experimentos, eles utilizaram mapas similares aos do Super Mario Bros e apresentaram uma série de estratégias tanto para o treinamento das cadeias de Markov quanto para a geração de mapas com base nelas. No entanto, a proposta deles oferecia nenhum grau de controle direto sobre o conteúdo gerado. O usuário se limitava a controlar quais mapas seriam utilizados pelo método para aprender os padrões estatísticos.

Existem tentativas de criação de interfaces humano-computador na área de aprendizado de máquina aplicado à GPC. [Halina and Guzdial 2021] desenvolveram um editor de autoria mista com o objetivo de criar diagramas musicais para partidas no *Taiko no Tatsujin*, um jogo de ritmo que simula tambores. Neste editor, o usuário posiciona as batidas rítmicas no diagrama, e um agente de IA complementa partes do diagrama através de geração procedural. A proposta é que a IA sirva como guia e fonte de inspiração durante o processo de criação da partida. Em contraste com a GPC convencional, essa abordagem não faz uso de um algoritmo gerador, mas sim da execução de um modelo de aprendizado de máquina. No caso de [Halina and Guzdial 2021], eles empregaram uma Rede Neural Recorrente (RNN) *Long Short-Term Memory* (LSTM) treinada para prever séries temporais arbitrárias, baseando-se em exemplos de outros diagramas do jogo.

Em contrapartida, há outros trabalhos que focam em um público alvo mais amplo, além dos desenvolvedores de jogos, como usuários para as ferramentas de controle de geração automática de conteúdo. [Lopes 2023] demonstrou que é possível que narradores de *Role-Playing Game* (RPG) de mesa usufruam de mapas de cidades medievais fictícias proceduralmente gerados com Diagramas de Voronoi.

Recentemente, temos observado o surgimento de algoritmos geradores que apresentam resultados promissores em termos de controlabilidade, como demonstrado nos resultados do trabalho de [Kim et al. 2020]. Nesse estudo, eles propuseram uma adaptação do WFC que aceita grafos como entrada em vez de uma grade de *bitmaps*. Nessa abordagem, os game designers conseguiram definir regras simples de associação por meio do grafo de entrada, as quais eram posteriormente utilizadas para gerar de forma procedural elementos-chave de conteúdo para as fases de um jogo. No entanto, é importante notar que a solução deles está limitada à edição dos vértices e arestas de um grafo como método de interação com o WFC, o que poderia dificultar o uso por parte de usuários não especializados.

Diferentemente do trabalho de [Kim et al. 2020], nossa pesquisa concentra-se em explorar como as características de controlabilidade do WFC podem ser traduzidas para uma UI de alto nível. Dessa forma, buscamos contribuir para o avanço com o desafio relacionado à carência de interfaces de usuário que permitam o controle preciso sobre algoritmos de GPC, conforme discutido na seção anterior.

O WFC foi escolhido como objeto de estudo neste trabalho devido à sua completa dependência da entrada do usuário durante o processo de geração [Gumin 2016]. Portanto, propomos uma maneira de tornar o WFC mais inteligível para um perfil diverso de usuários, incluindo leigos e desenvolvedores de jogos, com o objetivo de avaliar o potencial desse algoritmo como insumo para uma interface de usuário de controle de GPC. A próxima seção fornecerá detalhes sobre o que é o WFC, como ele funciona e qual versão do algoritmo foi empregada na solução desenvolvida.

3. O Algoritmo WFC

De acordo com os trabalhos de [Karth and Smith 2017], o algoritmo *Wave Function Collapse* (WFC) é definido como:

> Um algoritmo gerador de imagem baseado em exemplos, que surgiu a partir de necessidades práticas de Geração Procedural de Conteúdo (GPC). No WFC, novas imagens são geradas no estilo dos exemplos fornecidos, garantindo que cada janela local (NxN) na saída ocorra em algum lugar na entrada. Operacionalmente, WFC implementa um método de busca gulosa, *greedy search*, sem *backtracking*.

As primeiras aplicações de WFC advieram da área de computação gráfica como resposta ao desafio da síntese de textura. O desafio consiste em gerar uma imagem de maior escala, como saída, que apresenta uma textura semelhante à de uma imagem menor que é utilizada como entrada. Normalmente, as imagens de entrada e saída são caracterizadas com base nos padrões locais que elas contêm, onde tais padrões costumam ser sub-imagens de pequena dimensão, muitas vezes, apenas poucos pixels (como pode-se ver na Figura 1). [Gumin 2016], um desenvolvedor independente de jogos, propôs o WFC como uma técnica eficaz para síntese de textura.

O WFC recebe como entrada um *bitmap*, uma representação de uma imagem numa matriz de pixels, e retorna um outro *bitmap* de maior dimensão que possui semelhança local com o de entrada. Nesse contexto, entende-se semelhança local como o fato que a saída deve conter apenas os padrões de pixels NxN que estão presentes na entrada, onde N é o número de pixels por dimensão. Na Figura 2 pode-se visualizar como funciona a semelhança local. Primeiro o algoritmo dividi o *bitmap* de entrada em padrões NxN únicos. Então cada padrão é associado a um rótulo. Em sequencia, ele inicia todos os pixels do *bitmap* de saída em seu estado não observado, i.e., cada valor de pixel é a sobreposição das cores do bitmap de entrada, e.g., se o *bitmap* de entrada for preto e branco, os estados não observados seriam cinza. Então o algoritmo entra no ciclo observação-propagação que é definido como:

- Etapa de observação: Um padrão NxN não observado com a entropia mais baixa é colapsado em um estado fixo, recebendo, assim, algum dos rótulos pré-definidos.

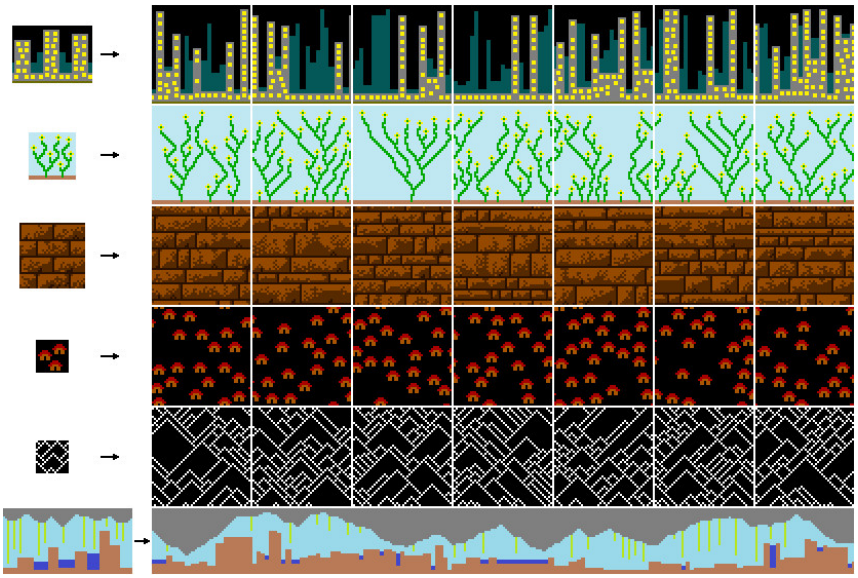


Figura 1. Exemplos de entradas e saídas do WFC [Gumin 2016].

- Etapa de propagação: As restrições de rótulo (impostas pela etapa anterior) para os vizinhos em estado não observado são propagadas por todo o bitmap de saída.

Por fim, o algoritmo termina quando a saída está completamente em um estado observado, ou seja, todos os pixels tem exatamente um rótulo associado a cada um deles. Todavia, o algoritmo pode chegar a um estado contraditório e não concluir a geração, isso ocorre caso um pixel não possa ser atribuído a um rótulo após a propagação das restrições. Da mesma forma, determinar se um dado bitmap permite uma solução sem contradições é um problema NP-difícil. Portanto, é impossível ter uma solução que nunca chegue a uma contradição, a menos que P = NP.

Figura 2. Demonstração da semelhança local para N=3. Nessa imagem pode-se visualizar alguns padrões 3x3 replicados do *bitmap* de entrada para o de saída [Gumin 2016].

3.1. Modelo de Tiles Simples

Uma variação do WFC, também proposta por [Gumin 2016], é o modelo de tiles simples. Nessa versão o algoritmo recebe como entrada uma lista de tiles e uma outra lista com as regras de adjacências entre eles. Como pode ser visto na Figura 3. Um tile é uma pequena imagem usada para compor elementos visuais em jogos, como por exemplo mapas de terrenos. Em relação ao WFC básico, o modelo de tiles simples muda a fase de propagação para apenas uma propagação de restrições de adjacências.

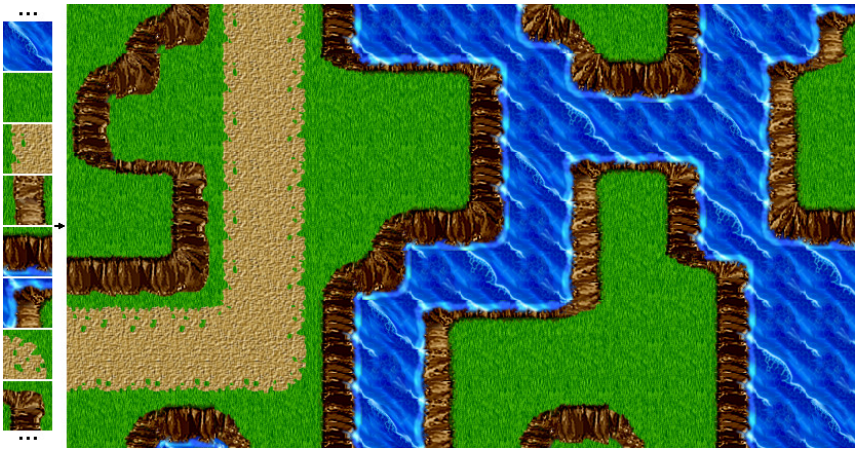


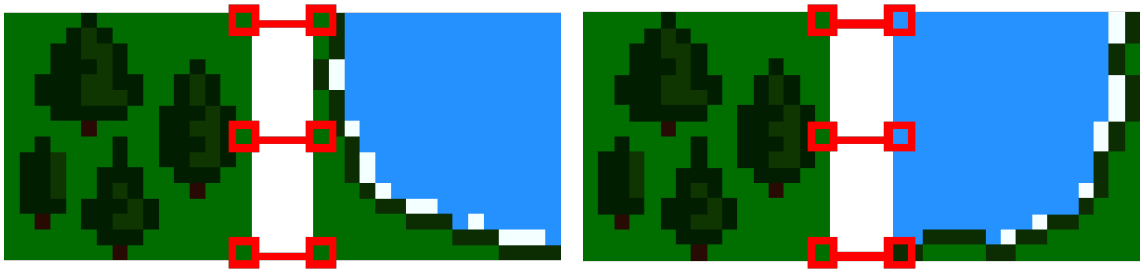
Figura 3. Exemplo de entrada e saída do WFC com o modelo de tiles simples [Gumin 2016].

É importante destacar que o modelo de tiles simples potencialmente requer uma lista extensa de todas as combinações de adjacências possíveis entre os tiles. Visto isso, [Gumin 2016] implementou um sistema de simetria para os tiles com a finalidade de encurtar a enumeração das adjacências. Nesse sistema cada tile pode ter um tipo de simetria definido pelos caracteres: “X”, “T”, “I”, “L” e “\”. Tais caracteres são isomórficos as ações do grupo diédrico D4. Dessa maneira, através de um par de tiles adjacentes o sistema consegue inferir as demais adjacências provenientes da simetria dos mesmos.

3.2. Geração de Adjacências

Apesar da implementação do sistema de simetria de [Gumin 2016], a lista de adjacências ainda pode ser considerada extensa, visto que elas devem ser definidas uma a uma. Para ilustrar isso, consideremos um dos mapas de tiles utilizados nos experimentos deste estudo (seção 5.1). Nesse cenário, para um conjunto de 7 tiles foram necessário definir no mínimo cerca de 30 regras de adjacências para obter resultados satisfatórios, ou seja, uma quantidade menor de regras não permitiu gerar mapas com variações significativas. Com o objetivo de simplificar o processo de configuração das adjacências, propomos um algoritmo que preenche automaticamente a lista com um conjunto inicial. Posteriormente, esse conjunto pode ser ajustado, seja reduzindo-o ou expandindo-o, de acordo com os objetivos do usuário.

A lógica subjacente a essa funcionalidade envolve a comparação de pontos fixos, denominados pontos de junção, nas bordas dos tiles, a fim de determinar se eles são vizinhos, como ilustrado na Figura 4a. Para cada adjacência identificada dessa maneira, são excluídas aquelas que podem ser inferidas pelo sistema de simetria. É importante destacar que a redução do número de pontos de junção aumenta a probabilidade de detecção de uma vizinhança, mas também amplia a possibilidade de falsos positivos (ver Figura 4b). Por outro lado, um aumento no número de pontos de junção reduz a probabilidade de falsos positivos, mas potencialmente aumenta a ocorrência de falsos negativos. Além disso, observamos que tiles com poucos detalhes (i.e. dimensão de até 16 pixels) obtiveram melhores resultados, ou seja, geraram uma quantidade significativa de adjacências com praticamente nenhum falso positivo; nesses casos, em geral, apenas 3 pontos de junção foram suficientes.



(a) Vizinhança detectada. **(b)** Vizinhança não detectada.

Figura 4. Funcionamento da detecção automática de vizinhança. Nesse exemplo utilizou a quantidade de pontos de junção igual a 3 e o conjunto de tiles *Tiny Islands* [Majadroid 2021].

Integrando tal funcionalidade de geração automática de adjacências, a ferramenta proposta neste trabalho é uma interface gráfica que permite que seu usuário crie, customize e experimente um gerador procedural de mapas de tiles que utiliza o algoritmo WFC com o modelo de tiles simples.

4. A Ferramenta

A arquitetura escolhida para o sistema implementado foi o modelo cliente-servidor. Nesse modelo, o cliente é responsável por enviar solicitações de serviços e/ou recursos ao servidor, que, por sua vez, recebe tais solicitações, processa-as e as envia de volta ao cliente solicitante. No contexto específico da ferramenta proposta neste trabalho, optou-se por tal arquitetura para centralizar o processamento do WFC no servidor, enquanto os clientes disponibilizam a interface gráfica na qual múltiplos usuários podem interagir simultaneamente.

4.1. Servidor

O servidor é uma aplicação NodeJs com ExpressJs que disponibiliza uma API REST que trata requisições HTTP [Nodejs 2022]. A API disponibiliza quatro pontos de acesso: salvar e/ou atualizar gerador, criar adjacências para gerador, listar geradores salvos e executar um gerador especificado.

O primeiro ponto de acesso trata requisições HTTP do tipo POST e PUT. Ele recebe um JSON com as informações do *tilemap*. O mesmo é processado e armazenado em um banco de dados Mongo [Inc. 2023] com a seguinte estrutura:

- *path*: Campo textual único que identifica o *tilemap*. Também é usado para definir o caminho onde as imagens dos tiles são armazenadas;
- *tilesize*: Campo numérico que define qual a dimensão de cada tile. O WFC com o modelo de tiles simples requer que todos os tiles utilizados na geração tenham a mesma altura e largura.
- *unique*: Campo *boolean* (verdadeiro ou falso) que determina os tile do tilemap são "únicos". Quando verdadeiro indica que cada tile possui uma imagem diferente para cada variação de acordo com a simetria (campo *symmetry*) que o mesmo possui. Já quando falso, apenas uma imagem é necessária e as demais variações do tile são obtidas automaticamente aplicando as ações do grupo diédrico D4, i.e, rotações e inversões da imagem original.

- *tiles*: Campo de lista. Cada item contém informações individuais de cada tile. Cada item possui os seguintes campos:
  - *name*: Campo textual com o nome do tile. E.g. “ponte”;
  - *symmetry*: Campo textural com um único caractere com a sigla da simetria correspondente ao tile, como definido na seção 3.1;
  - *weight*: Campo número que representa o peso do tile durante a etapa de propagação do WFC. Por padrão esse valor é 1 e pode variar entre 0 e 2;
  - *assets*: Campo de lista de cadeias de caracteres ASCII. Cada item é uma String Base64 que representa uma imagem codificada. Caso o campo *unique* seja falso, o tamanho da lista sempre será um, caso contrario, o tamanho da lista é definido pela simetria do tile (campo *symmetry*). Os índices da lista começão em 0 e terminam em N-1, onde N é o tamanho da mesma.
- *neighbors*: Campo de lista com as informações de adjacência (ou vizinhança) entre os tiles.
  - *left*: Vizinho da esquerda. É um campo textural que sempre deve apresentar o seguinte padrão: nome do tile, espaço em branco e índice da variação, nessa ordem. E.g., “ponte 1” significa que o vizinho da esquerda é a segunda variação do tile “ponte”;
  - *right*: Vizinho da direita. É um campo textural que apresenta o mesmo padrão do campo *left*.

O segundo ponto de acesso aceita solicitações do tipo GET e gera um conjunto de adjacências para um determinado conjunto de tiles. Em resumo, ele implementa a lógica descrita na seção 3.2. Essa funcionalidade foi implementada para simplificar o processo de cadastro de um gerador, uma vez que, como explicado na seção 3.1, a lista de adjacências pode ser potencialmente extensa e isso pode tornar o processo de cadastro longo e repetitivo para os usuários.

O terceiro ponto de acesso permite a listagem dos geradores cadastrados. Ele aceita requisições do tipo GET e simplesmente realizar uma consulta do modelo no banco de dados, retornando ao cliente solicitante uma lista com todos os geradores cadastrados. Vale a pena ressaltar que não há restrição de acesso entre os clientes. Qualquer usuário do sistema pode ter acesso a todas os geradores cadastrados, seja por ele ou por outros. Por fim, o último ponto de acesso é o que de fato executa o WFC e retorna uma imagem gerada proceduralmente. Esse ponto de acesso aceita requisições do tipo GET e ler o parâmetro de url “generator-path”. Em sequência, é feito uma consulta no banco de dados por um gerador com atributo “path” igual ao parâmetro de url lido. Então, o objeto encontrado é passado como entrada ao algoritmo gerador. Para tal, utilizou-se o porte em JavaScript [Chapelier 2021] do WFC. Lembrando que devido a natureza do WFC, há a chance que a execução termine em contradição, nesse caso ao invés de retornar a imagem gerada o ponto de acesso retorna uma mensagem com código de erro 500.

O servidor implementado pode ser acessado em `https://wfc-pcg.herokuapp.com` e seu código fonte encontrado em `https://github.com/wba25/wfc-pcg`.

4.2. Cliente

O cliente dessa aplicação é uma página web criada com o *framework* Vue.Js [Vue.js 2021]. Ela é composta por duas telas: *home* e cadastro (ou edição). A *home* é

a tela inicial do sistema, onde é possível gerir os geradores cadastrados por meio de uma tabela, como pode ser visto na Figura 3. Os dados desta tabela são obtidos por meio de uma requisição ao ponto de acesso de listagem, descrito na seção 4.1. Nas últimas colunas de cada linha o usuário tem acesso a duas ações: editar e gerar. A opção "editar" navega para a tela de cadastro, mas com o formulário já preenchido com as informações do gerador selecionado. Por outro lado, a opção "gerar" faz uma solicitação ao servidor por uma nova geração. No caso de sucesso, a página realiza o download da imagem procedural, permitindo, assim, que o usuário visualize o resultado. Porém, caso o servidor informe que a geração terminou em contradição, uma mensagem aparece para orientar o usuário.

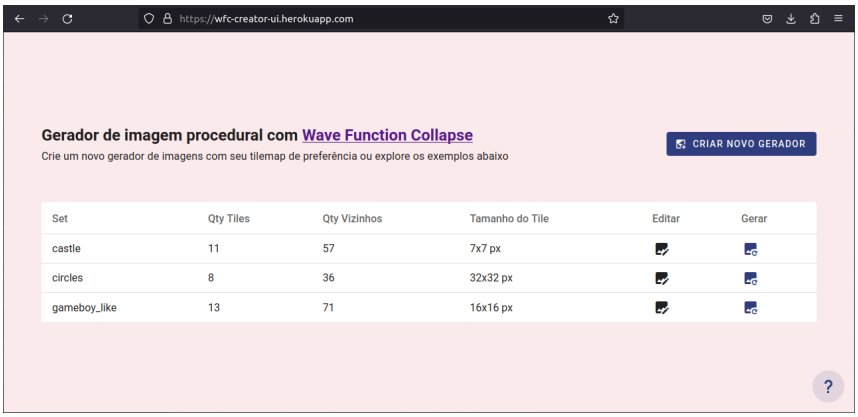


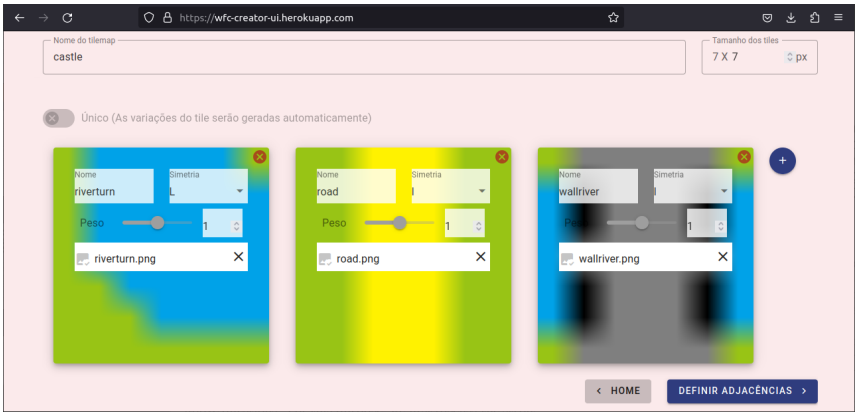
Figura 5. Tela Home do cliente da ferramenta desenvolvida. Nela temos uma tabela com a lista dos geradores cadastrados, onde é possível gerar imagens procedurais. Também temos um botão que leva para a tela de cadastro.

A tela de cadastro/edição é composta por duas etapas: tiles (ver na Figura 6a) e adjacências (ver na Figura 6b), o usuário navega entre as telas nessa ordem. O objetivo de ambas as etapas é coletar os dados necessários para executar o algoritmo WFC na versão de tiles simplificados, dessa maneira permite que o usuário do sistema tenha controle dos parâmetros do algoritmo gerador, podendo, assim, calibrar os padrões das imagens geradas. Os dados coletados são tratados e enviados para o ponto de acesso de cadastro descrito na seção 4.1. Ao término do processo, um novo gerador é cadastrado no banco de dados do servidor, mas já no caso de uma edição, os dados do gerador são apenas atualizados. Finalmente, de volta para a tela *home*, o usuário pode visualizar o gerador recém cadastrado na tabela de listagem, assim, podendo utilizá-lo para gerar mapas de tiles ou editá-lo caso os resultados não estejam de acordo com o esperado.

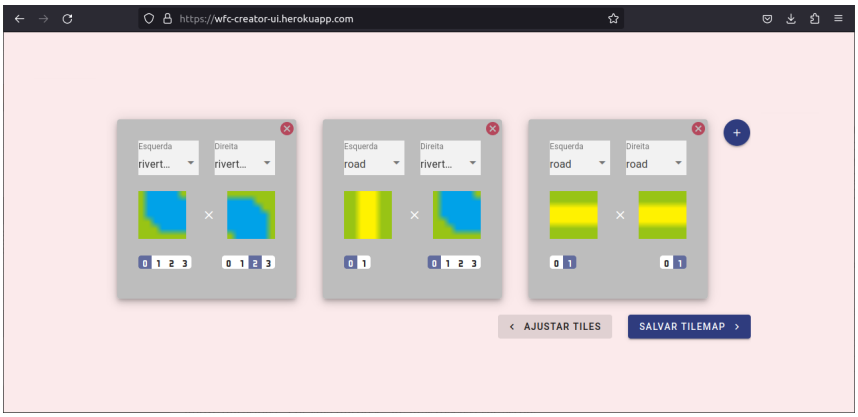
O interface gráfica está acessível para uso na página web `https://wfc-creator-ui.herokuapp.com` e seu respectivo código fonte em `https://github.com/wba25/wfc-pcg-ui`.

5. Avaliação

Com a finalidade de tornar o algoritmo WFC controlável por meio de uma UI e, assim, tornar a geração de mapas de tiles mais acessível, foi avaliada a experiência de uso da solução implementada. Durante os testes, foi requisitado aos participantes que adicionassem dois novos geradores. Em um deles era indicado utilizar a funcionalidade de "geração automática das adjacências" e no outro o conjunto de tiles foi recomendado o



(a) Etapa de configuração dos tiles.



(b) Etapa de configuração das adjacências.

Figura 6. Captura de tela: cadastro/edição de um gerador.

cadastro totalmente manual. Ao fim foi feito uma entrevista com o intuito de coletar *feedback* dos participantes, além disso, foram aplicados alguns questionários com o intuito de metrificar os resultados obtidos em relação a aspectos de usabilidade e experiência de usuário.

Todos os participantes, independentemente de sua experiência, enquadravam-se na faixa etária compreendida entre 18 e 30 anos e estavam matriculados em cursos de ensino superior ou já haviam concluído sua graduação. Nesse experimento, um total de 5 participantes foi envolvido, sendo esse grupo segmentado entre os perfis:

- P1: trabalha em um estúdio de desenvolvimento de jogos como artista. Confecciona *assets* para jogos 2D. Um participante.
- P2: desenvolvedor profissional de software. Não trabalha com desenvolvimento de jogos. Um participante.
- P3: estudante de graduação em ciências da computação. Um participante.
- P4: estudante de graduação em engenharia química. Dois participantes.

5.1. Metodologia

A ferramenta descrita na seção 4.2 foi disponibilizada para avaliação dos testadores por meio de uma página web[2], isso ofereceu a vantagem de possibilitar a realização dos expe-

[2]Acessível para uso em `https://wfc-creator-ui.herokuapp.com`

rimentos de forma remota. Visto que permitiu que os experimentos fossem conduzidos de maneira mais conveniente, eliminando a necessidade de deslocamentos físicos dos participantes. Além disso, a implementação via arquitetura cliente-servidor (seção 4) garantiu que todos os usuários compartilhassem um ambiente homogêneo durante os experimentos realizados, principalmente do lado do servidor, onde é mais suscetível à demanda de recursos computacionais mais substanciais, já que é nesse ambiente onde ocorre a execução do algoritmo WFC e, consequentemente, a geração das imagens. Além disso, com a autorização dos participantes, foi sugerido eles gravassem a tela do computador durante os testes.

Para assegurar constância nos resultados, todos os usuários seguiram o mesmo roteiro durante os testes. Inicialmente foi apresentado ao participante uma breve introdução sobre a proposta deste projeto. Em seguida, foi fornecido os insumos para cadastro dos geradores: um conjunto de imagens para os tiles e sugestões sobre simetria, peso e adjacências relacionadas aos mesmos. Feito isso, o usuário era instruído a cadastrar dois geradores com os seguintes conjunto de tiles: *Tiny Islands* [Majadroid 2021], com 16 pixels de dimensão de tiles, e *Summer* [Gumin 2016], com 48 pixels. A primeira etapa do cadastro de ambos era similar, em síntese, foi solicitado o preenchimento do formulário de cada tile. Porém, a distinção entre eles aconteceu no cadastro das adjacências. Nessa etapa do cadastro, foi sugerido a utilização da funcionalidade de preenchimento automático para o *Tiny Islands* e o cadastro manual para o *Summer*. Após isso, era o fim da etapa de cadastrado dos geradores. *Tiny islands* ficou com sete (7) tiles e quarenta e uma (41) adjacências; já *Summer* com treze (13) e quarenta e nove (49).

Um vez concluído o cadastro do gerador, o usuário foi encorajado a explorar o quanto de controlabilidade sobre os resultados a ferramenta possibilitava. Ele gerou imagens (usando uma tabela como na Figura 5) com os geradores que ele mesmo cadastrou, em seguida os editou e, posteriormente, regerou mais imagens. Isso possibilitou uma comparação entre as geradas antes e após a edição, assim, permitindo que os usuários explorassem como suas alterações influenciaram nas imagens resultantes. Eles eram livres para escolher o que editar, entretanto, em alguns casos os dados fornecidos faziam com que o WFC terminasse em contradição e não conseguisse gerar imagens, quando isso acontecia a edição tinha que ser refeita.

Como a proposta do presente trabalho é uma UI que permite aos usuários controle sobre a geração de mapas de tiles com WFC, logo é importante que a ferramenta proposta apresente bons resultados em aspectos de usabilidade e experiência de usuário. Entretanto, tais aspectos são subjetivos e difíceis de mensurar. Visto isso, foram utilizados duas escalas psicométricas de proposito geral. Para estimar a usabilidade utilizou-se o formulário System Usability Scale (SUS) [Brooke et al. 1996]. Esse formulário utiliza perguntas de escala Likert, ou seja, os participantes informaram o grau de concordância ou discordância por meio de uma escala de categorias: “Discordo totalmente”, “Discordo”, “Neutro”, “Concordo” e “Concordo totalmente”. Por fim, ele permite calcular a pontuação SUS, um valor de zero a cem que indica o quão boa é a usabilidade de um sistema. Já para experiência de usuário, utilizou-se o User Experience Questionnaire (UEQ) [Laugwitz et al. 2008], um formulário que consegue uma análise mais detalhada da experiência, separando seus resultados em seis métricas: atratividade, clareza, eficiência, confiabilidade, estimulação e inovação.

Dado que as escalas utilizadas são de proposito geral, também conduzimos uma entrevista semi-estruturada na qual consultamos os participantes sobre aspectos específicos da ferramenta. O objetivo era obter uma análise qualitativa mais aprofundada desses pontos particulares. Em resumo, o roteiro da entrevista consistiu nas seguintes questões:

- As imagens geradas corresponderam à sua intenção ao utilizar a ferramenta?
- Você consideraria usar a ferramenta em projetos futuros?
- Você conseguiria utilizar a ferramenta por conta própria e sem auxílio do tutorial?
- Você encontrou algum bug ou teve dificuldades ao usar a ferramenta?

5.2. Resultados e Discussões

A análise dos vídeos dos testes com os usuários revelou algumas tendências comuns em todos os casos estudados. Os experimentos duraram em média 25 minutos. A maior parte desse tempo foi gasta na tela de cadastro. Em relação a etapa de preenchimento dos tiles, todos os participantes demoraram mais tempo na primeira vez, já no segundo teste o cadastro era mais rápido e direto ao ponto, independente da ordem do conjuntos de tiles escolhida. Na etapa de preenchimento das adjacências, a abordagem por meio do preenchimento automático, conjunto *Tiny Islands*, foi mais rápida em todos os casos. Embora o cadastro manual das adjacências do *Summer* foi o processo mais demorado de todo o teste.

Durante a fase de exploração dos geradores cadastrados, observou-se que a maior parte das edições realizadas pelos usuários estava relacionada à modificação dos valores de peso de determinados tiles ou à remoção de adjacências específicas. Um exemplo notável foi a ação do usuário P1, que alterou as informações iniciais fornecidas para o conjunto de tiles *Tiny Islands* da seguinte maneira: quando ele diminuiu o peso dos tiles relacionados ao mar, as imagens resultantes apresentaram mais ilhas e áreas de terra maiores em comparação com a configuração anterior à edição. Por outro lado, ao remover adjacências que representavam conexões entre áreas de terra pontiagudas, as novas imagens não incluíam mais penínsulas.

As respostas obtidas a partir da entrevista conduzida após a conclusão dos testes proporcionaram a agregação de pontos de análise qualitativos sobre o sistema. De modo geral, os participantes expressaram que a ferramenta possibilitava ajustes posteriores ao cadastro, tornando os resultados mais interessantes do seu ponto de vista. No entanto, também observou-se, tanto por meio dos relatos dos participantes quanto nos vídeos, que ocorreram contradições mais frequentemente quando as edições se distanciaram do cadastro inicial, principalmente no caso dos usuários P1, P2 e P3. Isso resultou em restrições à liberdade de exploração da ferramenta, já que os usuários precisavam desfazer as mudanças. Infelizmente, é desafiador propor uma solução para esse problema, considerando que a detecção de contradições no algoritmo WFC é um problema NP-difícil. Apesar dessa limitação, esses resultados indicam que, mesmo com algumas restrições, os usuários conseguiram transmitir suas intenções nas imagens geradas pelo algoritmo por meio da edição dos dados de cadastro.

No que diz respeito aos problemas enfrentados durante o uso da ferramenta, todos os participantes, com exceção de P1, expressaram que a ferramenta parecia ser confusa e demandava esforço, mas não apresentou bugs. Em contraste, P1, P2 e P3 afirmaram que

adquiriram autonomia para utilizar a ferramenta por conta própria após os testes realizados.

Além disso, P1 e P4 demonstraram interesse em utilizar a ferramenta em projetos futuros. No entanto, P3 e P2 enfatizaram que a ferramenta precisa de melhorias para torná-la mais prática caso decidam utilizá-la novamente. No caso de um dos participantes do perfil P4, ele manifestou o desejo de empregar a ferramenta em seu estado atual para criar mapas para jogos de tabuleiro. Enquanto isso, P1 expressou interesse em usá-la para gerar proceduralmente mapas de tiles a partir de conjuntos de mapas que ele produziu no passado, dentre eles, o usuário destacou que gostaria de testar com tiles de um projeto pessoal de um jogo de RPG. Essas perspectivas destacam as diferentes visões dos participantes em relação à utilidade e à usabilidade da ferramenta em suas respectivas aplicações futuras.

Sobre os dados coletados pelas escalas psicométricas. A pontuação SUS foi de 65 pontos. Esse valor é considerado abaixo do limite inferior para uma confiabilidade interna aceitável, que é de 68 pontos segundo [Sauro 2011]. Isso significa que, apesar do sistema ser utilizável, ainda há o que melhorar em relação a usabilidade. Isso ficou mais evidente com as respostas do UEQ e da entrevista.

Com a utilização do UEQ, conseguimos obter informações mais detalhadas sobre aspectos específicos que influenciaram a experiência do usuário. Conforme evidenciado na Figura 7, os resultados do UEQ indicam que a maior parte dos participantes teve uma percepção que variou de neutra a negativa em relação à atratividade e clareza de uso do sistema. Uma hipótese plausível é que as etapas de cadastro tenham contribuído para essa experiência negativa, pois P2, P3 e P4 mencionaram durante a entrevista que o cadastro de tiles foi confuso na primeira vez e a fase de cadastro manual das adjacências foi considerada trabalhosa e repetitiva.

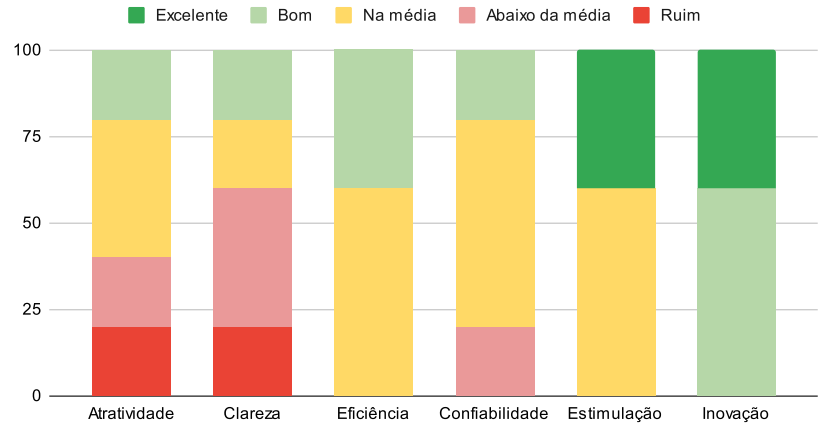


Figura 7. Resultados obtidos pelo User Experience Questionnaire. A escala vai de 0 a 100%, e.g., 20% dos participantes tiveram a percepção que a atratividade do sistema é "boa".

Adicionalmente, é importante notar que 60% dos usuários classificaram a percepção de confiabilidade da ferramenta como neutra. Isso pode estar relacionado ao fato de que, ocasionalmente, como observado nos vídeos e nas entrevistas, o processo de

geração resultava em contradições. Apesar desses pontos negativos, é encorajador observar que a maioria dos participantes teve uma impressão positiva em relação à eficiência, estímulo e, sobretudo, inovação proporcionados pela ferramenta.

6. Conclusão

Este artigo explorou uma UI como maneira de controlar e interagir com a GPC de mapas de tiles criados via WFC. Em ótica dos *feedbacks* das entrevistas, resultados das escalas psicométricas e análise dos experimentos, pode-se afirmar que o algoritmo WFC foi inteligível via uma interface gráfica de usuário, visto que, um grupo diversos de usuários, incluindo não familiarizados com desenvolvimento de jogos, conseguiu utilizar a ferramenta proposta.

Apesar dos participantes expressaram interesse em usar a ferramenta em projetos futuros, foram identificados pontos de melhoria em relação à usabilidade e à experiência de usuário. As etapas de cadastro, em particular o cadastro manual de adjacências, foram consideradas confusas e trabalhosas, possivelmente, afetando a atratividade e a clareza de uso do sistema. Com o propósito de aprimorar o grau de controle e a facilidade de interação, possíveis melhorias futuras podem incluir a introdução de maior personalização por parte do usuário, e.g., permitir que ele defina a posição inicial de alguns tiles, para que o algoritmo possa então preencher o restante do mapa. Além disso, a inclusão de recursos que facilitaram a execução de processos manuais foi bem recebida, como a capacidade de gerar automaticamente adjacências entre tiles.

Podemos concluir que o modelo de tiles simples do WFC demonstrou ter considerável grau de controlabilidade através de uma interface de alto nível, pois um grupo diversos de usuários foi capaz de manipular geradores procedurais de mapas de tiles. Contudo, vale ressaltar que a ferramenta proposta não conseguiu lidar de maneira eficaz com as contradições do WFC durante o processo de geração, o se configurou como uma limitação que dificultou a interação dos usuários, pois tornou a UI menos flexível a edições. Adicionalmente, este projeto poderia ser expandido para abranger a criação de mapas tridimensionais ou outros tipos de *game content*, como missões e trilhas sonoras. Isso demonstra o potencial de aprimoramento e expansão da ferramenta desenvolvida neste estudo.

Além do mais, é altamente encorajada a exploração e adaptação de outras técnicas de geração procedural, como a Inteligência Artificial Generativa (IA Generativa), para serem integradas em uma interface gráfica de usuário. Essa abordagem pode representar avanços substanciais na superação do desafio da falta de controle de alto nível na área de GPC. Ao combinar técnicas de generativas com uma interface de usuário amigável, podemos abrir portas para a criação de conteúdo procedural de forma mais intuitiva e personalizada. Isso tem o potencial de revolucionar a maneira como os jogos são desenvolvidos e, ao mesmo tempo, impulsionar a inovação em toda a indústria de jogos.

Referências

Brooke, J. et al. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7.

Chapelier, K. (2021). Javascript port of Wave Function Collapse. https://github.com/kchapelier/wavefunctioncollapse.

Compton, K. and Mateas, M. (2006). Procedural level design for platform games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 2, pages 109–111.

Games, H. (2016). No man's sky. https://www.nomanssky.com/. Último acesso em: 21 de agosto de 2023.

Gumin, M. (2016). Wave Function Collapse Algorithm. https://github.com/mxgmn/WaveFunctionCollapse.

Halina, E. and Guzdial, M. (2021). Taikonation: Patterning-focused chart generation for rhythm action games. In *Proceedings of the 16th International Conference on the Foundations of Digital Games*, pages 1–10.

Hendrikx, M., Meijer, S., Van Der Velden, J., and Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1–22.

Inc., M. (2023). Mongodb. https://www.mongodb.com/. Último acesso em: 19 de junho de 2023.

Karth, I. and Smith, A. M. (2017). Wavefunctioncollapse is constraint solving in the wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, pages 1–10.

Kiciński, A., Nielubowicz, P., and Nowakowski, M. (2020). Transcript of the teleconference dedicated to the cd projekt group annual results for 2020. https://www.cdprojekt.com/en/wp-content/uploads-en/2021/04/transcript-2020-results.pdf. Último acesso em: 20 de agosto de 2023.

Kim, H., Hahn, T., Kim, S., and Kang, S. (2020). Graph based wave function collapse algorithm for procedural content generation in games. *IEICE TRANSACTIONS on Information and Systems*, 103(8):1901–1910.

Laugwitz, B., Held, T., and Schrepp, M. (2008). Construction and evaluation of a user experience questionnaire. In *HCI and Usability for Education and Work: 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2008, Graz, Austria, November 20-21, 2008. Proceedings 4*, pages 63–76. Springer.

Lopes, J. H. (2023). Criação de mapas de cidades para rpgs de mesa utilizando algoritmos de geração procedural.

Majadroid (2021). Tiny islands - 16x16 tilemap. https://opengameart.org/content/tiny-islands-16x16-tilemap. Último acesso em: 26 de agosto de 2023.

Mello, R., Calegario, F., and Ramalho, G. (2023). Elodin: Naming concepts in embedding spaces. *arXiv preprint arXiv:2303.04001*.

Nodejs (2022). Node.js is an open-source, cross-platform javascript runtime environment. https://github.com/nodejs/node. Último acesso em: 19 de junho de 2023.

Sauro, J. (2011). *A practical guide to the system usability scale: Background, benchmarks & best practices*. Measuring Usability LLC.

Shaker, N., Togelius, J., and Nelson, M. J. (2016). *Procedural content generation in games*. Springer.

Short, T. and Adams, T. (2017). *Procedural generation in game design*. CRC Press.

Snodgrass, S. and Ontañón, S. (2014). Experiments in map generation using markov chains. In *FDG*.

Togelius, J., Champandard, A. J., Lanzi, P. L., Mateas, M., Paiva, A., Preuss, M., and Stanley, K. O. (2013). Procedural content generation: Goals, challenges and actionable steps. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Togelius, J., Yannakakis, G. N., Stanley, K. O., and Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186.

Vue.js (2021). Vue.js - the progressive javascript framework. https://vuejs.org/. Último acesso em: 19 de junho de 2023.

Yannakakis, G. N., Liapis, A., and Alexopoulos, C. (2014). Mixed-initiative co-creativity.