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

Avaliação da geração de conteúdo por Wave Function Collapse na experiência do jogador

Rodrigo Coelho Villa Verde

JUIZ DE FORA
SETEMBRO, 2021

Avaliação da geração de conteúdo por Wave Function Collapse na experiência do jogador

Rodrigo Coelho Villa Verde

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Igor de Oliveira Knop

JUIZ DE FORA

SETEMBRO, 2021

Avaliação da geração de conteúdo por Wave Function Collapse na experiência do jogador

Rodrigo Coelho Villa Verde

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 CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Igor de Oliveira Knop
D.Sc. Modelagem Computacional

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

Luiz Maurílio da Silva Maciel
D.Sc. Engenharia de Sistemas e Computação

JUIZ DE FORA

8 DE SETEMBRO, 2021

Resumo

A criação de um jogo bem sucedido, digital ou não, consiste em fornecer ao jogador uma narrativa envolvente, composta de roteiros, cenários, ilustrações, animações e progresso de personagens. Essa indústria bilionária envolve profissionais das mais diversas áreas como artistas, programadores, analistas de sistemas, engenheiros, matemáticos, roteiristas, psicólogos e outros. Para pequenos estúdios, o uso de algoritmos especializados para criar parte do conteúdo tem sido uma alternativa para baratear os custos em pessoal e tempo de projeto. Entretanto, o resultado da geração deve ainda ser coerente e gerar uma experiência prazerosa para o jogador. Este trabalho investiga uma das técnicas mais recentes de geração procedural, a *Wave Function Collapse*, e observa o resultado da geração na experiência do jogador em relação a topologia, desafios e recompensas geradas. Uma implementação do algoritmo foi realizada e adaptada a um ambiente de testes, que serve como laboratório para estudos em geração procedural. Uma discussão é realizada com base nas observações do resultado da geração, que servem de base para trabalhos subsequentes.

Palavras-chave: *Wave Function Collapse*, desenvolvimento de Jogos, geração procedural de conteúdo.

Abstract

Developing a successful game is to provide the player with an engaging narrative, made up of scripts, scenarios, illustrations, animations, and character progress. This billion-dollar industry involves professionals from the most diverse areas such as artists, programmers, engineers, mathematicians, screenwriters, psychologists, and others. For small studios, the use of specialized algorithms to create part of the content has been an alternative to reduce costs in staff and project time. However, the generation result must still be coherent and generate a pleasurable experience for the player. This work investigates one of the most recent procedural generation techniques, the Wave Function Collapse, and evaluates the generation result in the player's experience. An implementation of the algorithm is carried out and adapted to a testing environment, which serves as a laboratory for studies in the area. Results are discussed, providing insight to future works.

Keywords: Wave Function Collapse, Game development, procedural content generation.

Agradecimentos

A todos os meus parentes, por confiar em mim.

Ao professor Igor pela orientação, paciência e oportunidade de trabalhar em um projeto tão interessante.

Aos professores do Departamento de Ciência da Computação pelas boas aulas e disponibilidade para explicações extra classe.

Aos meus amigos de curso que enfrentaram todos os desafios comigo

Conteúdo

Lista de Figuras

Siglas

IA *inteligência artificial*. 16

PCG *Procedural Content Generation*. 12

PCGLab *Procedural Content Generation Laboratory*. 4, 16, 19, 20, 22, 23, 33, 35, 36

WFC *Wave Function Collapse*. 4, 5, 8–10, 13–16, 19–36

1 Introdução

O mercado de jogos digitais está em constante crescimento e é muito diversificado: possui desde empresas multinacionais fazendo os chamados jogos "AAA" (do inglês, *triple A*, que são jogos que se espera grande sucesso de crítica e público, e grande retorno financeiro) a empresas independentes, algumas vezes compostas por menos de três pessoas, criando jogos praticamente sem nenhum orçamento inicial (conhecidos como *indies*). Essa indústria emprega um grande número de pessoas e é um sonho para um número muito maior. Com demanda para atuação de profissionais de diversas áreas, como artistas, programadores, analistas de sistemas, engenheiros, matemáticos, roteiristas, psicólogos, jornalistas, profissionais de marketing e até jogadores profissionais, muito esforço é despendido a cada minuto em busca de profissionalização.

Com esse conjunto diverso de demanda por proficiências, uma equipe de desenvolvimento de um jogo digital normalmente é formada por diferentes profissionais ou uma única pessoa assume diferentes papéis. O processo de criação envolvendo tantos profissionais ou acúmulo de tarefas é custoso em tempo e dinheiro. Dessa forma, fica cada vez mais difícil se fazer um jogo lucrativo e menos vezes consegue-se fazer jogos com sucesso comercial (SHAKER; TOGELIUS; NELSON, 2016). Para contornar essa situação, os estúdios buscam desenvolver métodos para inovar na criação do jogo, seja através de novos mecanismos inovadores, ou de conteúdo gerado artificialmente por programas de computador, diminuindo assim o trabalho humano necessário e podendo fazer mais e melhor e em bem menos tempo.

Estúdios de jogos de grande porte podem se beneficiar muito de gerar conteúdo proceduralmente: diferentes jogos usam o método de gerar um grande e vasto mapa para depois colocar seus artistas para inserirem os detalhes. Mas os grandes beneficiados, e mais conhecidos por usar tais métodos, são os estúdios independentes que fazem jogos inteiros ao redor de geração automática de conteúdo, substituindo o trabalho de toda uma equipe por algoritmos de geração. Com isso, a geração procedural é vista como uma importante fonte de corte de gastos e pesquisas na área podem levar a um mercado mais

lucrativo.

Geração procedural é um método algorítmico de criar conteúdo dada uma entrada limitada ou indireta do usuário (SHAKER; TOGELIUS; NELSON, 2016). O conteúdo gerado pode variar muito, podendo ser arte, música, terreno, histórias, objetos, objetivos e todo tipo de elementos para uso em um jogo.

Existem diferentes métodos de geração procedural, como autômatos celulares, gramáticas gerativas, geração por ruído de *Perlin*, *Wang Tiles* e *Wave Function Collapse* (WFC). Cada um deles apresenta suas peculiaridades que serão melhor explicadas na fundamentação teórica.

Usar conteúdo gerado apresenta alguns prejuízos: um delicado balanço entre controle e variabilidade deve ser levado em consideração. Faça uma geração com muito pouco controle e ela poderá gerar artefatos não funcionais ou que entregam uma experiência ruim ao jogador. Por outro lado, com uma geração muito controlada, todos os resultados podem ser muito parecidos e dar o sentimento de repetição cansativa para o jogador. O que conduz à principal questão deste trabalho: "Como a geração procedural por WFC afeta a experiência do jogador?".

A experiência do jogador pode ser avaliada pelo conceito de fluxo (CSIKSZENTMIHALYI et al., 1990). O conceito é baseado em um fluxo psicológico com efeitos para retenção de atenção, de forma a nublar a percepção de tempo e de si mesmo. Com isso o jogador ficaria focado no jogo, tendo um grande sentimento de controle, mesclando ações conscientes com a perda da consciência de si. Conseguir um estado de fluxo para o jogador é fundamental para que ele queira continuar jogando. Em tal estado o jogador é compelido pelos desafios do jogo e a própria experiência de jogar é a justificativa do ato. Com isso espera-se que a geração procedural alcance um resultado que mantenha o fluxo e, usando esse conceito, este trabalho busca discutir como a geração afeta a experiência do jogador.

Este trabalho tem como objetivo principal investigar se o conteúdo gerado por uma das técnicas a, WFC, gera uma experiência satisfatória ao jogador, e se ajustes nos ladrilhos da geração influenciam diretamente nessa experiência. A WFC foi escolhida como objeto de estudo por ser uma das técnicas mais recentes, tendo ficado bem po-

pular atualmente (KARTH; SMITH, 2017) e por permitir um certo controle na direção da geração, usando diferentes entradas. Para atingir o objetivo principal, uma série de objetivos específicos devem ser atingidos:

1. Implementar a WFC seguindo a literatura;
2. Adaptar a implementação para funcionar junto ao ambiente para estudo de geração procedural;
3. Verificar a variabilidade, ou seja, se a repetição dos padrões se torna monótona;
4. Analisar como os diferentes mapas gerados podem influenciar no modo de jogar;
5. Analisar a influência das características dos ladrilhos de entrada no resultado final.

O método para investigação é baseado em uma pesquisa qualitativa, na qual foi implementado o WFC e o mesmo foi acoplado a um ambiente de testes. Usando essa estrutura, foi avaliado como as características dos diferentes mapas gerados influenciam na forma de jogar. Com esses dados, foi estimado a qualidade da geração, a influência de seus parâmetros e quais alterações pós-geração seriam necessárias para uso.

Este documento foi organizado em cinco capítulos. Além desta introdução, o Capítulo 2 traz um histórico e explicações dos conceitos fundamentais aqui utilizados. O Capítulo 3 apresenta o método usado no desenvolvimento deste trabalho. O Capítulo 4 contempla o processo de desenvolvimento e analisa os resultados. Por fim, o Capítulo 5 possui as considerações finais, seguido pelas referências bibliográficas utilizadas.

2 Fundamentação Teórica

Este trabalho analisa a experiência do jogador dada a geração por WFC. Portanto, é necessário revisitar os conceitos aqui utilizados, desde a cultura de jogos até os principais métodos de geração procedural.

2.1 Jogos e Jogos Sérios

Jogos acompanham a cultura de diversos povos por toda a história da humanidade (HUIZINGA, 1971). Definir os mesmos é uma tarefa em aberto tendo diferentes interpretações dependendo da área (LEWIS, 1968). No contexto desse trabalho será usada a definição de jogo como ambiente no qual os jogadores têm objetivos artificiais controlados por uma série de restrições, tendo um final quantificável (SALEN; TEKINBAŞ; ZIMMERMAN, 2004).

Jogos possuem objetivos próprios e independentes do ambiente externo (HUIZINGA, 1971). Enquanto *jogos sérios* é um termo referente a jogos em que se tem um objetivo além do divertimento interno (ABT, 1970; MICHAEL; CHEN, 2005). Esses são usados para treinamentos, educação ou informação (DJAOUTI et al., 2011).

2.2 Mapas de influência

Segundo Tozour (2001), um agente para ser inteligente deve conseguir tomar decisões dado seu ambiente. Um mapa de influência é uma representação de toda informação que alcança o agente, permitindo um entendimento dos arredores para tomar uma decisão considerada estratégica. Esse mapa marca pontos positivos ou negativos, atraindo e repelindo agentes. Os mapas carregam informações sobre o que estão representando, permitindo ao agente tomar decisões estratégicas, como as observadas em jogadores humanos (TOZOUR, 2001).

O mapa é um espaço geográfico dividido em setores onde cada setor pode representar múltiplos estados em referência a diferentes influências. Desse setor, a influência

desses estados é propagada para os setores vizinhos sucessivamente (MARK, 2015).

O mapa de influência não comanda o agente. Ele entrega dados sobre a região, compartilhando essa informação com todos os agentes. Assim evitando processar informação individualmente para cada um deles e esses apenas fazem suas escolhas (MARK, 2015).

Por exemplo, no mapa da Figura 2.1 um oásis em um deserto propagaria pelo mapa uma influência positiva sobre recursos, assim os agentes naquele mapa saberiam para onde ir caso precisassem daquele recurso. Quanto maior o valor mais próximo se esta do ponto de interesse.

Figura 2.1: Mapa de influência com oásis no centro. Fonte: do autor.

2.2.1 Parâmetros do mapa

A influência exercida por um setor tem nele um valor inicial, que depois propaga de forma decrescente para os setores adjacentes. Quanto maior a distância, menos da influência inicial é presente. Usam-se técnicas de propagação para fazer esse controle, geralmente usando uma relação exponencial. Por exemplo, um setor com 10 de influência e 0.5 de fator de propagação faria os setores adjacentes terem 5 de influência e os seguintes 2.5 assim em diante (TOZOUR, 2001).

O agente toma diferentes decisões ao longo do tempo. Para garantir a acurácia,

o mapa deve se atualizar propagando a influência desde o princípio, sendo a forma mais eficiente atualizar em períodos fixos de tempo. Uma mecânica mais flexível de atualizar é usando algum sinal relacionado a aquele mapa. (TOZOUR, 2001).

2.2.2 Adquirindo informações

Para entender o mapa, o método mais básico é analisar as informações do setor em que o agente está. Entretanto, normalmente um agente deve analisar uma combinação de mapas. Para combinar mapas um agente usa diferentes funções de combinação, com diferentes fórmulas relacionando os valores dos mapas individuais (MARK, 2015).

A função costumeira para unir mapas é uma soma do valor de cada mapa relacionado àquela decisão, sendo que cada valor é multiplicado por um peso que relaciona a importância daquele dado para a escolha em vigor (TOZOUR, 2001).

2.3 Geração Procedural de Conteúdo

A *Procedural Content Generation* (PCG) ou geração procedural de conteúdo, é o processo no qual uma entrada limitada vai gerar mais conteúdo como música, ilustrações, inimigos, equipamento, histórias e, de particular interesse deste trabalho, mapas. Como dito em Shaker, Togelius e Nelson (2016), os termos procedural e geração implicam que se está lidando com algoritmos que criam algo. Um dos usos dessa geração é a criação de uma grande quantidade de conteúdo base que depois pode ser ajustado e detalhado à mão.

Short e Adams (2017) falam do uso de geração procedural para lidar com a falta de armazenamento. Assim, apenas com a semente para o algoritmo gerador, seria possível criar conteúdo em tempo de execução, que ocuparia gigabytes de espaço em disco caso fosse armazenado.

Existem alguns tipos de geração procedural como a genérica que cria conteúdo independente do jogador. Ou seja, todos que jogarem têm as mesmas chances de ver o mesmo resultado da geração. E a geração adaptativa, que leva em consideração como o jogo está sendo jogado. Com isso pela forma de se jogar os jogadores podem influenciar no resultado da geração (SHAKER; TOGELIUS; NELSON, 2016). Diferentes algoritmos

podem ser utilizados para geração procedural e alguns serão discutidas a seguir.

Um autômato celular consiste em uma grade de células, com evolução em passos discretos de tempo. Cada célula tem um conjunto finito de estados, além de possuir um número finito de vizinhos. O autômato define regras de como uma célula muda de estado durante os passos de tempo em função das células vizinhas (CASTRO; CASTRO, 2015). Johnson, Yannakakis e Togelius (2010) mostram a aplicação de autômatos celulares em geração procedural de cavernas.

Já a técnica por *Wang Tiles* consiste num conjunto de ladrilhos com os lados marcados, de forma que apenas ladrilhos com lados de mesma marcação podem se conectar. Dessa forma, após escolher um ladrilho inicial, o algoritmo executa as regras de adjacência, assim gerando o conteúdo (ALSTES, 2004).

Shaker, Togelius e Nelson (2016) explicam que uma gramática é uma sequência de regras produtoras que, a partir de um símbolo inicial, gera construções mais complexas que seguem as regras. Todo símbolo da geração é analisado se pode virar outro, caso contrário é um ponto final da geração. E com esse método se criam as gramáticas generativas, as quais usam blocos de conteúdo como símbolos. Esse método gera resultados bem controlados e garante parâmetros como a conectividade, mas por esse mesmo motivo, muitas vezes, gera mapas repetitivos e previsíveis.

2.4 *Wave Function Collapse*

O termo *Wave Function Collapse*, ou em uma tradução livre, colapso da função de onda, vem da Mecânica Quântica, que descreve como múltiplos estados se colapsam em apenas um durante uma observação. A Figura 2.2 apresenta uma função de onda de uma partícula com sua densidade de probabilidade antes de uma observação. Se uma medida for feita, a sua função de onda "colapsa" para um único impulso (FITZPATRICK, 2020).

Esse conceito foi utilizado por Gumin (2016) para criar um algoritmo de geração de imagens. O algoritmo de WFC busca resolver um sistema de restrições no espaço (KARTH; SMITH, 2017). Ele divide o espaço em uma grade com cada célula podendo estar em um conjunto finito de possíveis estados. O algoritmo é popularmente explicado com o exemplo de solução do jogo *sudoku*. Esse jogo de escrever se passa em

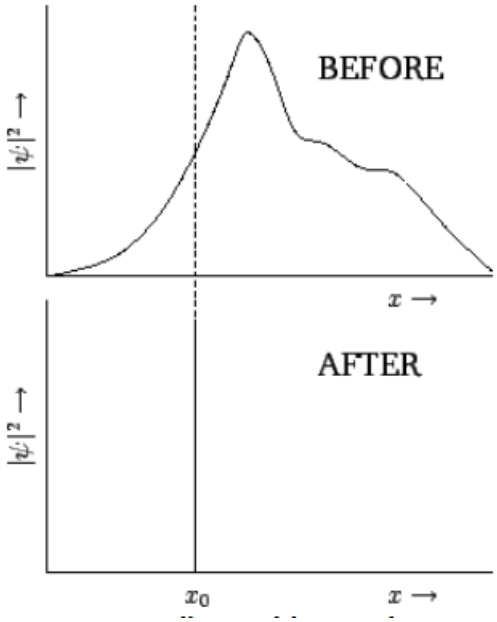


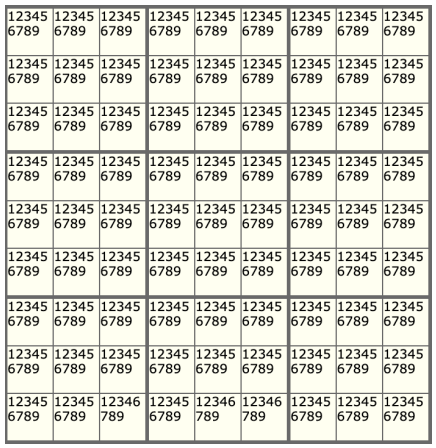
Figura 2.2: Na Mecânica Quântica, uma função de onda de uma partícula colapsa para um impulso após uma observação. Fonte: reproduzido de Fitzpatrick (2020).

uma matriz de 9 por 9, onde o estado das células pode ser um número de 1 a 9. A matriz é dividida em 9 subespaços de 3 por 3. As regras de adjacência são que um número pode ser colocado na célula se e somente se não tiver um igual na mesma linha, coluna ou no subespaço. O WFC pode ser utilizado para criar um tabuleiro do jogo antes de se apagar algumas das células para servir de quebra-cabeça.

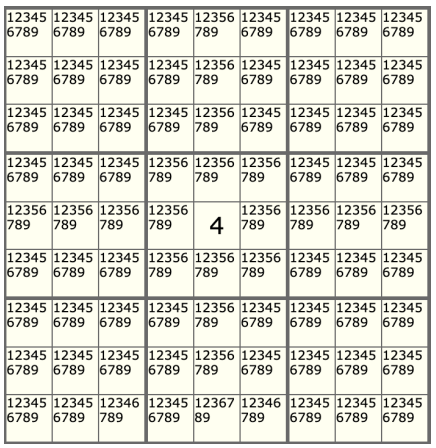
Diferentes passos do WFC podem ser vistos na Figura 2.3. Inicialmente, temos que as células estão em todos os estados, como uma sobreposição (Figura 2.3a). A entropia no WFC é definida como a sobreposição de estados. Uma célula em muitos estados tem uma entropia alta. Então, o algoritmo busca a região de menor entropia para realizar uma observação. Em caso de empate, uma das opções é sorteada entre as empatadas.

Já na Figura 2.3b a célula central foi definida (por sorteio ou usando qualquer critério) com o valor 4 (colapso do estado por observação/medição). Neste momento, todos os estados, que agora são impossíveis, são removidos das células impactadas pelas restrições. Neste caso, o 4 é removido dos estados de todas as outras células na linha, coluna e subespaço da célula que colapsou. Agora essas são as que possuem a menor entropia. Então a resolução continua ao sortear uma delas para realizar o próximo colapso/observação. O estado da Figura 2.3c apresenta o sorteio do valor 2 de uma das células, e a atualização para as outras células da linha, coluna e subespaço. Esse processo

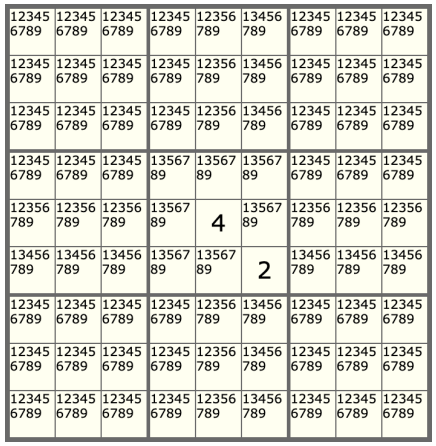
continua até que todas as células tenham se colapsado, resultando em um estado final como o exemplificado na Figura 2.3d.



(a) Estado inicial com células com estados sobrepostos.



(b) Primeira iteração: colapso da célula central e propagação.



(c) Segunda iteração: colapso de uma das células com menor entropia.

5	2	4	1	6	3	8	9	7
7	9	1	8	5	4	2	6	3
6	8	3	7	2	9	1	5	4
4	1	2	9	7	6	3	8	5
8	5	7	3	4	1	9	2	6
3	6	9	5	8	2	7	4	1
9	7	6	2	3	5	4	1	8
2	4	8	6	1	7	5	3	9
1	3	5	4	9	8	6	7	2

(d) Estado final com todas células colapsadas.

Figura 2.3: Uso do WFC para resolver um jogo de *sudoku* como exemplo. Fonte: adaptado de Donald (2020).

As implementações do WFC geram seus ladrilhos a partir de uma amostra, como pode ser visto em Gumin (2016) e derivam suas regras de adjacência com base na semelhança em relação a amostra original.

Entre as limitações da WFC original, tem-se que o método possui regras complicadas de configuração, falta de controle global, dificuldade de aplicar restrições de distância e a impossibilidade de gerar níveis multicamadas (CHENG; HAN; FEI, 2020). Essas limitações têm sido contornadas em uma série de trabalhos subsequentes e o WFC tem sido utilizado em diferentes contextos(KARTH; SMITH, 2017). Joseph Parker fez implementação do código para a ferramenta *Unity* adaptando o algoritmo para um am-

biente tridimensional. Pouco depois Joseph Parker, com um pequeno time, desenvolveu o jogo *Proc Skater 2016*[1], que usa WFC para produzir um parque de skate infinito. Já a empresa *Freehold Games* desenvolveu o jogo *Caves of Qud*[2], usando WFC para geração de seus mapas mas criando gerações intermediárias setorizadas para contornar a falta de controle da geração global.

Existem outros usos do algoritmo fora do desenvolvimento de jogos, como Martin O'Leary que desenvolveu um gerador de poesia[3] utilizando WFC, no qual as células são as sílabas e as restrições de vizinhança vêm da rima e métrica.

Team et al. (2021) utilizam WFC para gerar a topografia do terreno no qual treina agentes independentes. Com isso gera diversos ambientes para um treinamento não especializado, preparando a *inteligência artificial* (IA) para diferentes situações.

2.5 O ambiente PCGLab

Este trabalho é uma continuação do esforço de entender como a geração procedural influencia na experiência de jogador e pode ser direcionada para melhores resultados. A base de implementação é feita sobre o ambiente de código aberto criado por Costa e Knop (2020) e que atualmente é denominado PCGLab[4]. O projeto possui um ambiente que permite observar o resultado de algoritmos de geração e realizar o teste de jogabilidade, percorrendo o cenário recolhendo os tesouros e enfrentando inimigos. A estética inicialmente utilizada é a de um *dungeon crawler*, no qual o personagem se esgueira por masmorras em um contexto de fantasia medieval.

O funcionamento do jogo embutido no ambiente é simples. O jogador deve atravessar todas as salas do mapa indo de um portal ao outro. Um dos elementos do jogo são fogueiras espalhadas pelo mapa: o jogador tem um tempo limite que pode ficar distante de uma fogueira, ou ele perde uma vida na partida. Outra condição de derrota vem dos ataques de inimigos que estão espalhados pelo mapa: caso a vida do jogador seja reduzida a zero, ele perde uma vida. Os inimigos podem ser atacados pelo jogador e ao

[1]Site do jogo Proc Skater 2016. Disponível em ⟨https://arcadia-clojure.itch.io/proc-skater-2016⟩.
[2]Site oficial do jogo Caves of Qud. Disponível em ⟨https://www.cavesofqud.com⟩.
[3]Codigo do gerador de poesia disponível em ⟨https://github.com/mewo2/oisin⟩.
[4]Página do projeto PCGLab, disponível em ⟨https://github.com/ufjf-gamelab/pcglab/⟩.

morrerem desaparecem. Além disso, pelo mapa existem moedas para serem coletadas. O jogo pode ser visto na Figura 2.4.

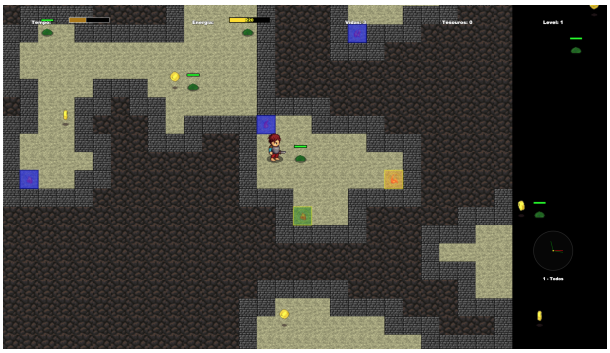


Figura 2.4: Jogo em execução.

O jogo funciona como uma prova de conceito. Ainda está em estágio inicial, mas seus elementos, como teleportes, fogueiras, inimigos e tesouros, estão diretamente relacionados a, respectivamente, navegabilidade, pontos de interesse, desafios e recompensas. Ou seja, os elementos constituintes da jogabilidade. Todos os elementos são posicionados utilizando mapas de influências para garantir uma distância uniforme entre os elementos. Os teleportes garantem a conectividade entre as diferentes salas do jogo. A Figura 4.5 usa linhas brancas para mostrar como os teleportes conectam as salas.

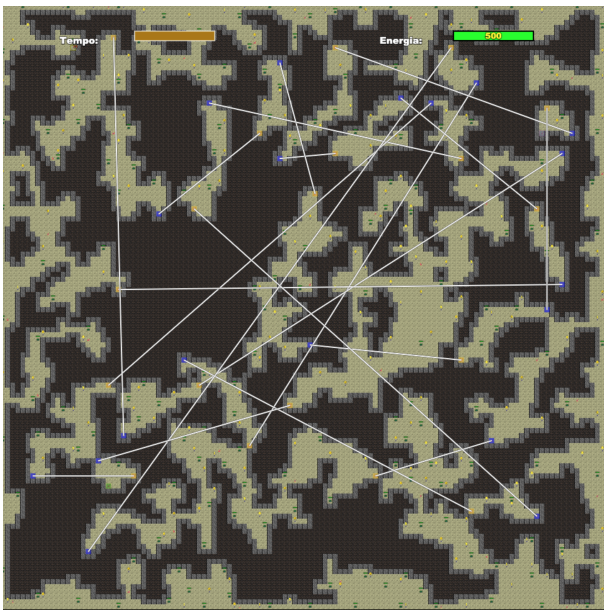


Figura 2.5: Conexão dos mapas via teleportes.

O ambiente permite abrir um modo de depuração no qual diversos elementos podem ser visualizados, como caixas de colisão, conectividade entre salas e mapas de

influência de cada um dos conceitos do jogo. Utilizando essas visualizações, espera-se fornecer uma série de métricas e avaliações que podem ser a base para construir modelos de experiência do jogador.

3 Método

Neste capítulo é feita a formalização do método seguido para a execução deste trabalho. Esta monografia apresenta uma pesquisa qualitativa, baseada na observação do comportamento do WFC em um ambiente de experimentação web, o PCGLab (COSTA; KNOP, 2019; COSTA; KNOP, 2020).

O PCGLab possui uma implementação de geração de mapas por autômatos celulares baseada no trabalho de Togelius et al. (2011). Ao implementar as gerações por WFC se espera avançar o projeto adicionando novos tipos de mapas gerados e, dessa forma, contribuir para o estudo de como eles podem alterar a experiência do jogador. Por exemplo, se artefatos de geração como corredores apertados ou salas grande, levam a jogabilidades diferentes. O objetivo é encontrar uma possível ligação entre o resultado da geração com a experiência de usuário.

Foi observado como a distribuição de elementos de jogo utilizando o mapa de influência interage com as topologias criadas pelo WFC. E como a distribuição de obstáculos e recompensas, do PCGLab, funciona nos diferentes cenários. Foi analisado se é possível observar a dificuldade, se com diferentes topologias o jogo se torna impossível ou muito simples.

Espera-se que os diferentes mapas gerem experiências diferentes de jogo. Alguns mais diretos com objetivos em vista e o desafio sendo somente desviar dos inimigos, enquanto outros vão exigir mais exploração. Com isso, espera-se observar a experiência sendo alterada pela geração.

4 Desenvolvimento

Este capítulo explica o processo de desenvolvimento do projeto, seguindo o método descrito previamente. A implementação do WFC e sua inserção no projeto PCGLab é explicada em detalhes. As dificuldades e problemas encontrados são discutidos e os resultados são apresentados.

A arquitetura foi implementada utilizando HTML, CSS e Javascript dentro do editor VS Code, executada pelo navegador Chrome e controle de versão *git*.

4.1 Implementação do WFC

Após estudar a bibliografia, foi feita uma classe chamada "wfc" para executar o algoritmo de WFC. Esta classe recebe os ladrilhos como parâmetro. Um de seus métodos define a área a ser gerada. Este método recebe como parâmetro dois pontos, sendo as coordenadas de início e fim da área. Com uma área definida, um outro método sem parâmetros é chamado para executar a geração e retornar o mapa. Durante a geração mais alguns métodos da classe "wfc" são chamados, como a função "entropia", que recebe uma célula da geração e retorna a entropia naquele ponto. As células da geração são matrizes que guardam os ladrilhos ainda possíveis para aquela posição. Outro método importante dessa classe é o de vizinhança, que recebe duas células e a direção de sua conexão (entre norte, sul, leste e oeste) e retorna se essa conexão é possível. O último método importante para essa classe é o de propagação, que é chamado para atualizar os ladrilhos possíveis para cada célula. Com esses métodos o código de geração segue buscando a célula de menor entropia, escolhendo pseudoaleatoriamente um de seus estados possíveis e então chamando o método de propagação.

As escolhas aleatórias feitas pelo código recebem uma semente de geração, essa semente pode ser passada pelo usuário ou escolhida aleatoriamente. O código nem sempre chega em um estado que é possível cumprir as condições de adjacência, quando isso acontece a geração recomeça do zero.

Para esta implementação do WFC, um ladrilho foi definido como uma *template string* no qual o número de linhas é sua altura e o número de caracteres por linha é sua largura. Um conjunto de ladrilhos é suficiente para definir as regras de geração. Um ladrilho nada mais é que um bloco de texto. a Figura 4.1 mostra, à esquerda, a representação textual e, à direita como isso é traduzido no mapa do jogo.

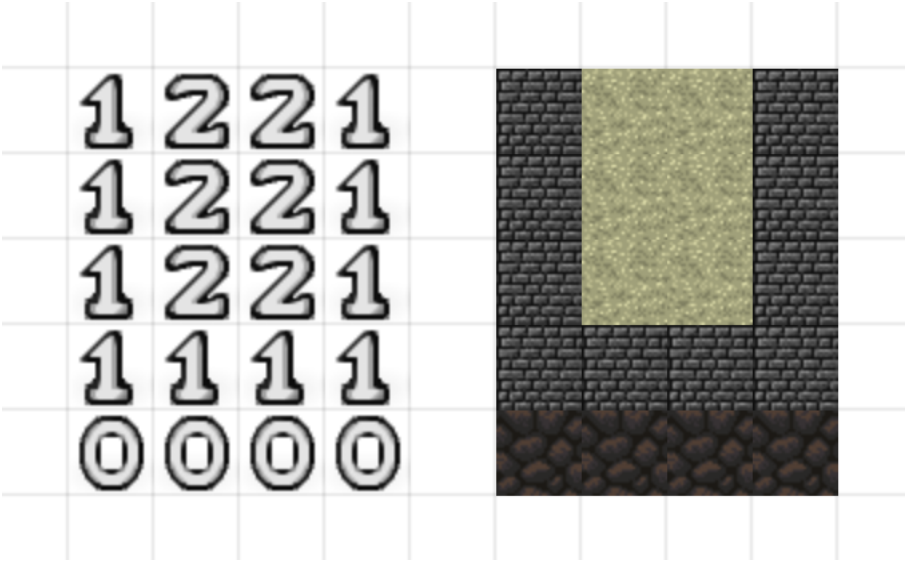
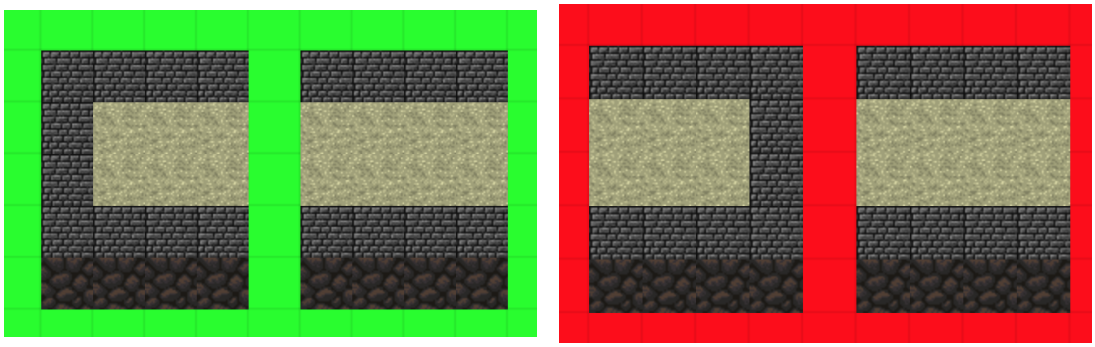


Figura 4.1: Representação de um ladrilho.

Este trabalho não gera seus ladrilhos a partir de uma figura de exemplo como em outros da literatura. Cada ladrilho é feito manualmente e eles são incluídos a partir de arquivos texto antes a cada execução.

A restrição de vizinhança dos ladrilhos é a que eles devem estar ao lado de um outro ladrilho que tenha a mesma borda. Na Figura 4.2a, pode-se ver os ladrilhos de forma que se conectam. Já na Figura 4.2b, observa-se que os ladrilhos não compartilham a borda e, portanto, não podem se conectar.

Com objetivo de trazer ferramentas úteis para o WFC, foi feito também uma classe "FerramentaWFC". Esta classe tem o método *addTile()* para receber e armazenar em um vetor os ladrilhos um a um. Outro método desta classe é o *transform()*, que, dados os ladrilhos se aplica uma função que gera três rotações para cada ladrilho. Como pode se ver na Figura 4.3. Com tudo isso, pode se chamar a função que inicializa o WFC usando todos os ladrilhos como regras. A Figura 4.6 mostra 4 exemplos de conjunto de ladrilhos. A classe "FerramentaWFC" também tem como método o *generateWFCInput()*



(a) Ladrilhos que se conectam (b) Ladrilhos que não se conectam

Figura 4.2: Compatibilidade de ladrilhos

que retorna o vetor de ladrilhos para execução do WFC.

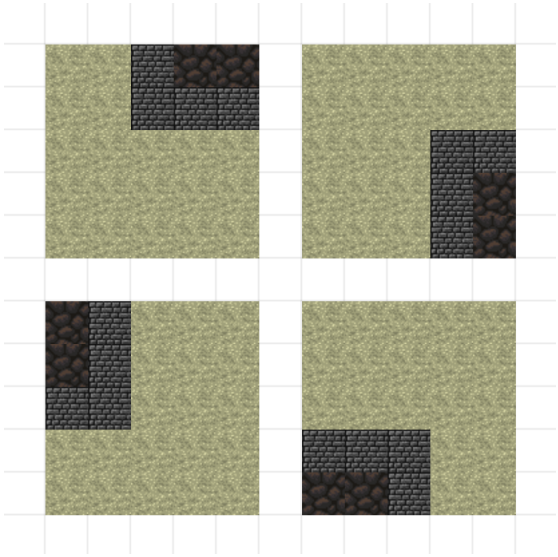
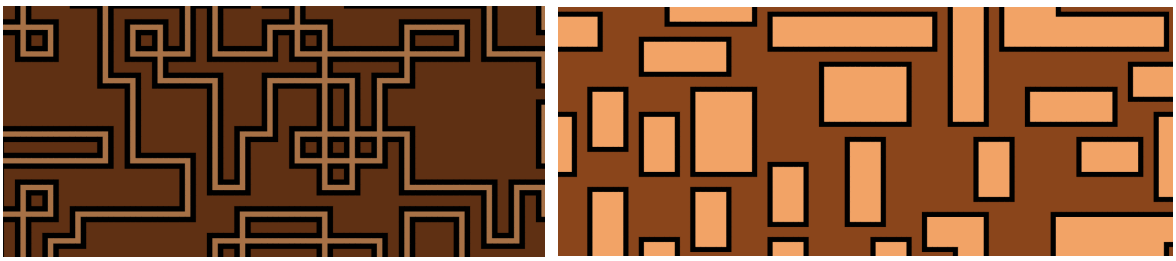


Figura 4.3: A partir de um ladrilho inicial, três outros são gerados ao aplicar uma, duas e três rotações em sequência.

O início do projeto seguiu o mesmo método do trabalho de Costa e Knop (2020), implementando os métodos de geração isoladamente para só depois realizar a integração no PCGLab. Foi criado uma ferramenta de apoio para visualizar o resultado da geração. A Figura 4.4 apresenta o resultado da geração nessa ferramenta. Ela foi utilizada em apoio para a experimentação rápida da montagem dos ladrilhos de referência que são utilizados na geração do WFC. Como nessa etapa não era necessário gerar o jogo inteiro, mas apenas para explorar a topologia, o visualizador apenas faz a conversão dos ladrilhos para cores sólidas.



(a) Geração mais estreita (b) Geração de salas amplas

Figura 4.4: Ferramenta de apoio criada: visualizador da geração para o WFC em JavaScript.

4.2 Evolução do PCGLab

O projeto teve como base o código aberto de PCGLab (COSTA; KNOP, 2019; COSTA; KNOP, 2020). Ao foram adicionadas as duas classes “wfc” e “FerramentaWFC” para a implementação do WFC. Além disso, uma classe “uniao” foi criada para armazenar os conjuntos de ladrilhos e executar o WFC.

A classe recebe o tamanho desejado do mapa e o nome do conjunto de ladrilhos escolhido, chamando a classe “FerramentaWFC” que lida com a geração. Com o mapa já gerado uma outra função é chamada, pois o WFC retorna o mapa dividido nos ladrilhos que foram fornecidos. Essa função converte o mapa para o formato do projeto PCGLab, que é apenas uma matriz de números com as mesmas proporções que o mapa.

Uma classe com os mesmos métodos que a classe “CellularAutomata” já existente, foi criada para que se proceda os mesmos passos da geração, mas substituindo o método por WFC. Com isso todo código relacionado a geração por autômato celular foi removido, a classe apenas recebe o mapa já gerado por WFC e procede para os próximos passos de geração, como contar o número de salas e posicionar os elementos.

Ao menu do PCGLab foi adicionada a opção de escolher o método de geração por WFC. Para tal, foi preciso mudar a ordem de operações de geração. Originalmente, o PCGLab fazia a geração ao ser carregado (é uma página executando no navegador). Uma nova partida era uma execução em cima da geração realizada no carregamento do sistema. Com as alterações realizadas, cada nova partida gera um novo mapa, a menos que uma semente de números aleatórios seja passada como parâmetro no endereço da página: nesse caso, uma mesma semente sempre leva ao mesmo mapa.

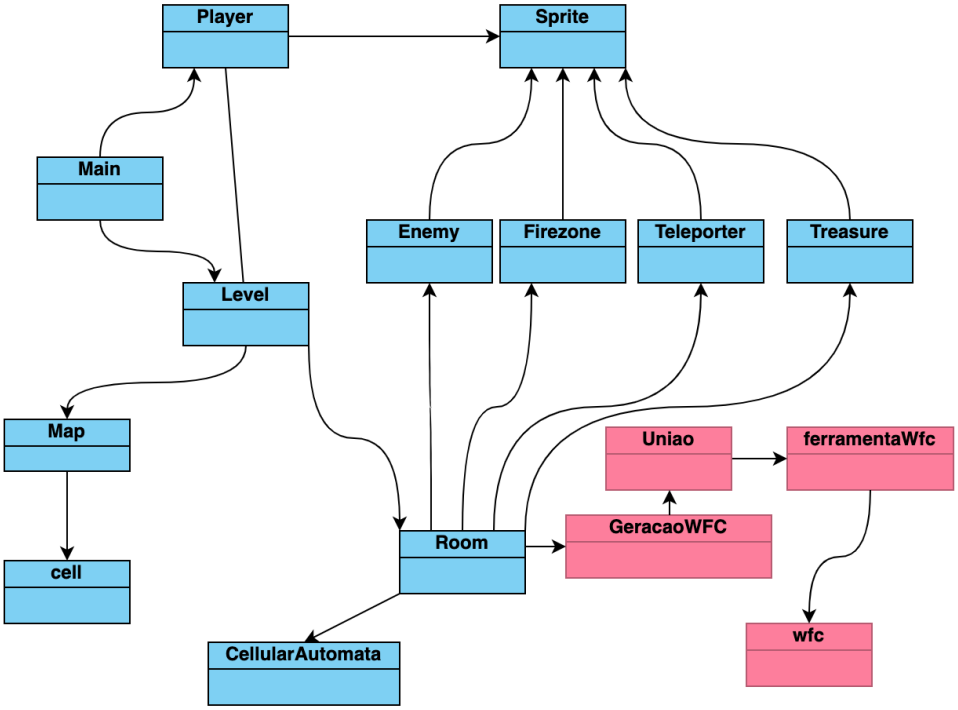


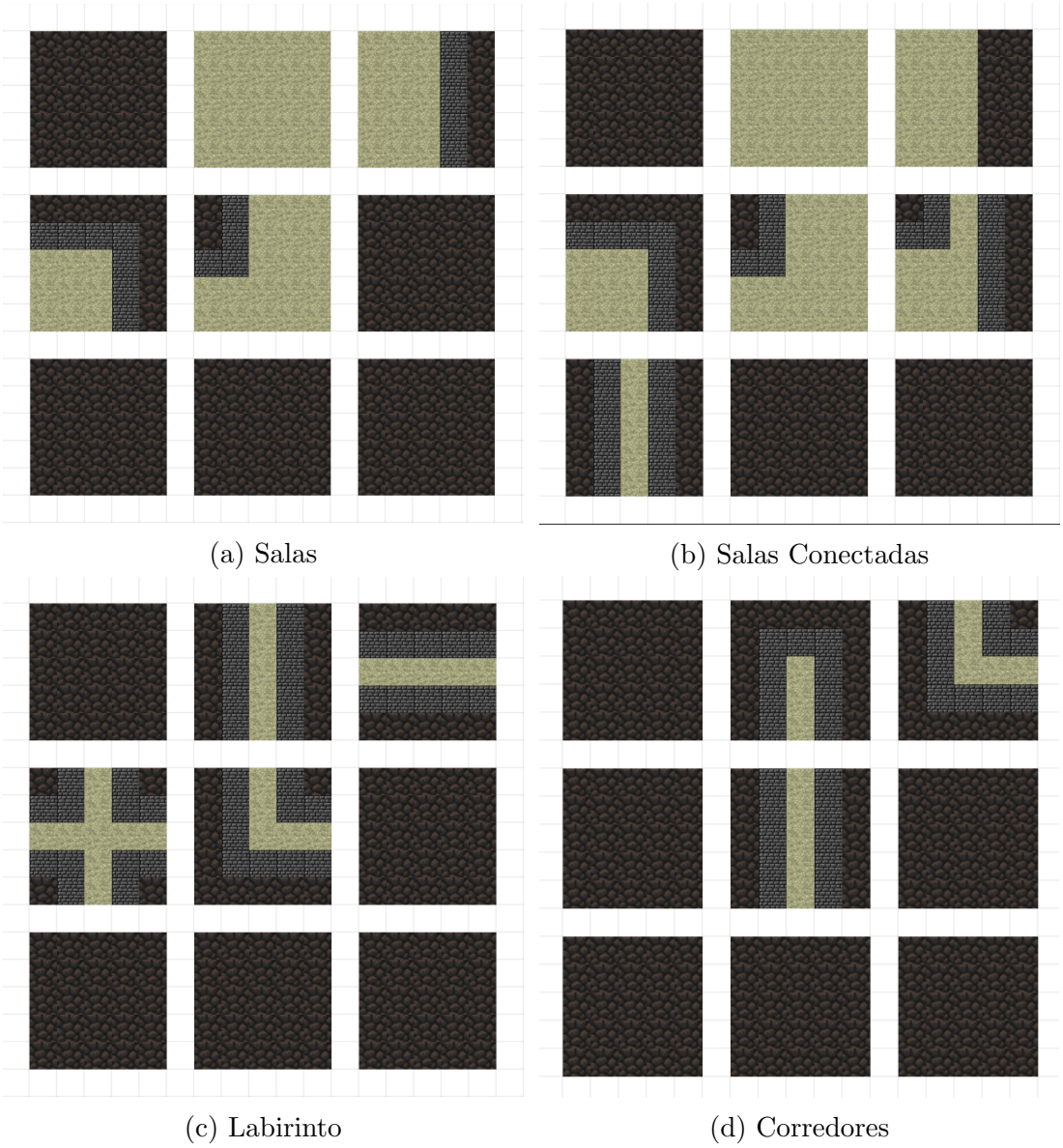
Figura 4.5: Diagrama de classes com a adição do WFC. As classes desenvolvidas nesse trabalho estão destacadas em rosa.

4.3 Cenários de Geração via WFC

Para realizar as observações sobre o comportamento da geração, quatro conjuntos de ladrilhos foram definidos: “Salas”, “Corredores” , “Labirinto” e “Salas Conectadas”. Os conjuntos de ladrilhos podem ser observados na Figura 4.6. Cada conjunto de ladrilhos foi gerado manualmente, e o conjunto final também possui, para cada ladrilho, as suas versões rotacionadas que são geradas automaticamente.

Ao contrário da geração por autômato celular, não é preciso fazer poda de salas pequenas com o resultado do WFC. Este cuidado é necessário apenas ao se montar os conjuntos de ladrilhos, pois estes podem gerar resultados incoerentes.

Para os resultados aqui apresentados, utilizou-se a geração baseada na semente de números aleatórios com o valor 10. O resultado da geração para os quatro cenários diferentes pode ser visualizado na Figura 4.7.



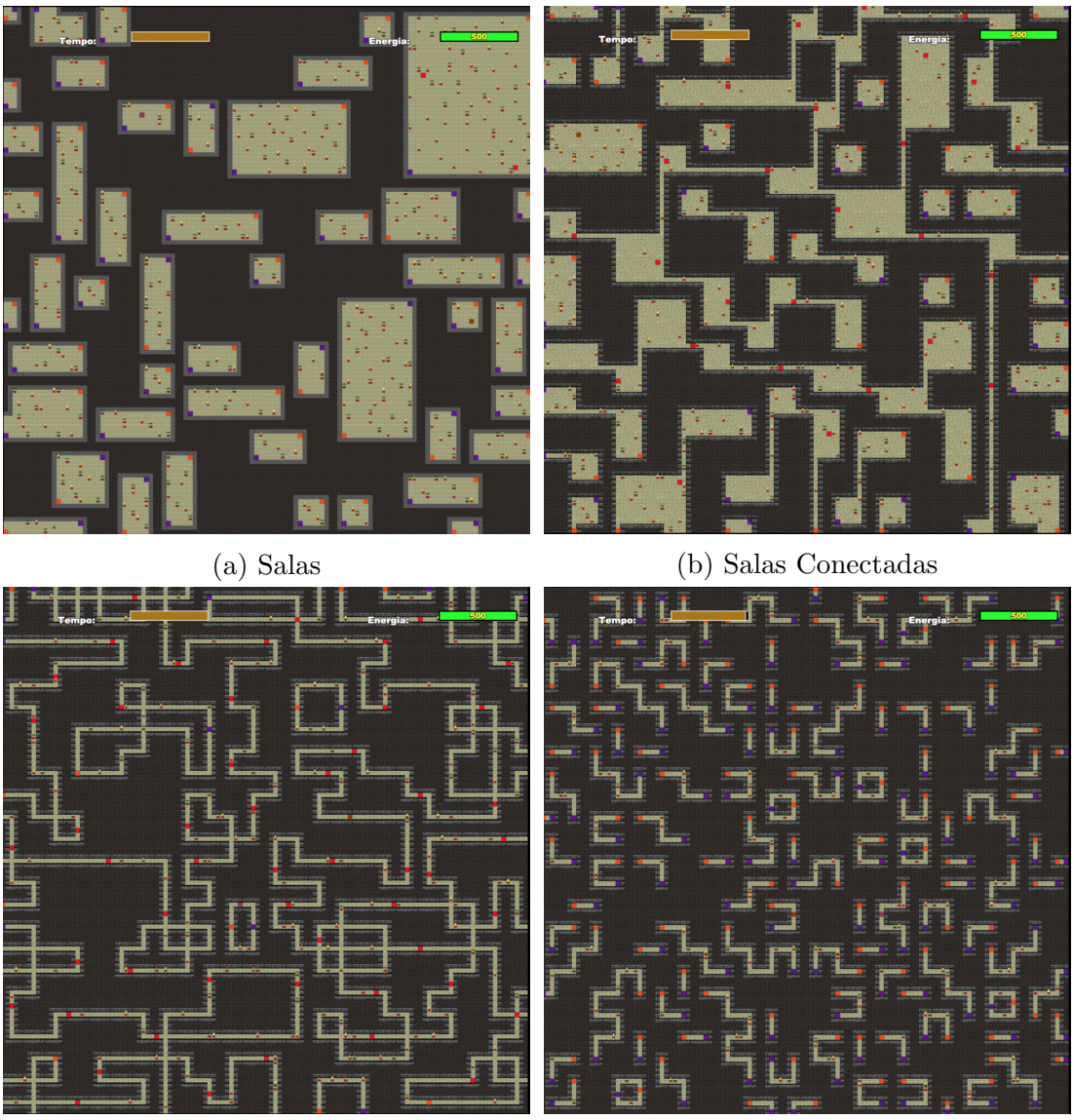
(a) Salas (b) Salas Conectadas

(c) Labirinto (d) Corredores

Figura 4.6: Os quatro conjuntos de ladrilhos utilizados para as observações do WFC.

4.4 Avaliação dos Cenários de Geração via WFC

A avaliação foi separada em cada conjunto de ladrilhos. Foi observado como o algoritmo de distribuição de elementos por mapa de influência utiliza os mapas gerados. Foram analisados os mapas de influência dos teletransportes (que representam a navegabilidade), das fogueiras (pontos de interesse), dos inimigos (desafios) e das moedas de ouro (recompensas). Desta forma, é possível discutir as opções que a geração fornece. Durante as análises, o mapa de influência demonstra proximidade usando um esquema onde vermelho está mais próximo ao ponto de interesse e verde seria o mais distante.



(a) Salas (b) Salas Conectadas

(c) Labirinto (d) Corredores

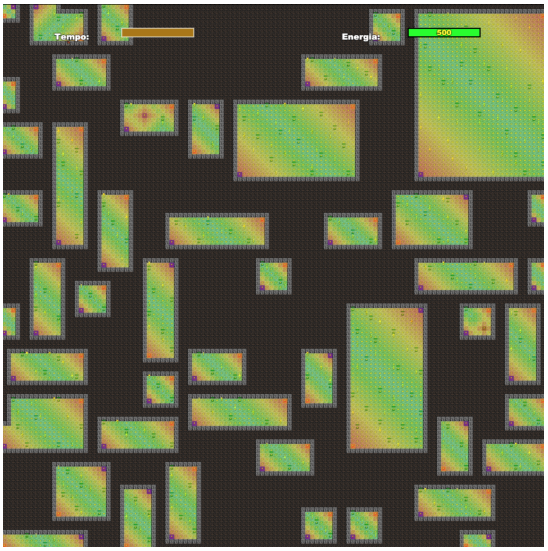
Figura 4.7: Resultado dos 4 conjuntos de ladrilhos com *seed* 10.

4.4.1 Cenário "Salas"

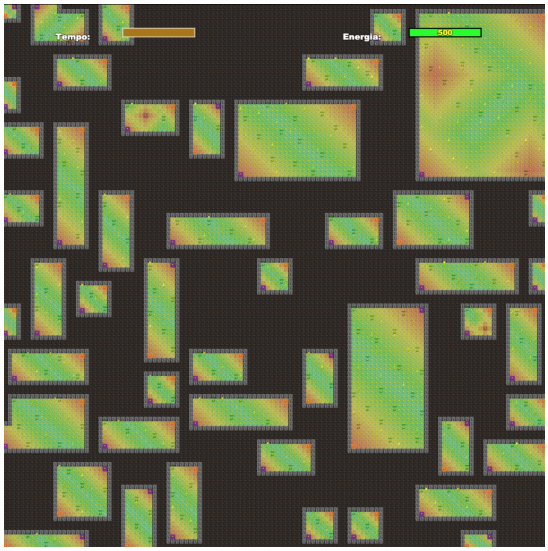
O conjunto de ladrilhos "Salas" é o mais simples pois gera salas retangulares de forma sistemática. A sua navegação é bem simples pois o algoritmo de posicionamento desenvolvido em Costa e Knop (2020) tem preferência por posicionar a entrada em um extremo e o mapa de influência faz a saída ficar no outro extremo, conforme pode ser observado nas Figuras 4.8a e 4.9a. O padrão só é quebrado em salas com a entrada e saída do nível (início e término da partida).

A distância entre as fogueiras ficou prejudicada pelos parâmetros do algoritmo de posicionamento: como cada teletransporte também age como uma fogueira, apenas

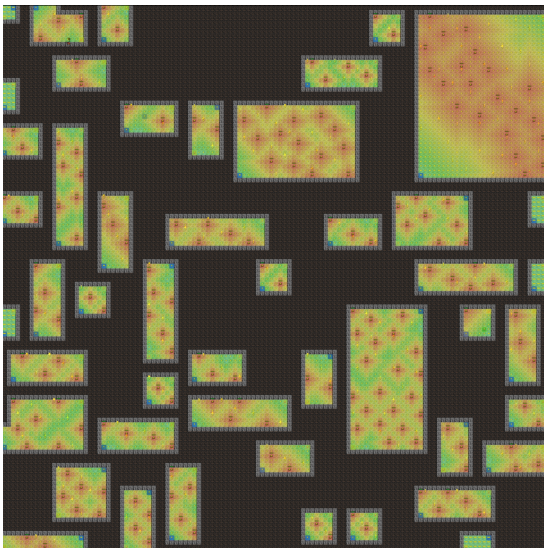
salas com áreas bem grandes necessitam de novos pontos. As Figuras 4.8b mostram essa distribuição de influência com a maior sala com duas fogueiras posicionadas.



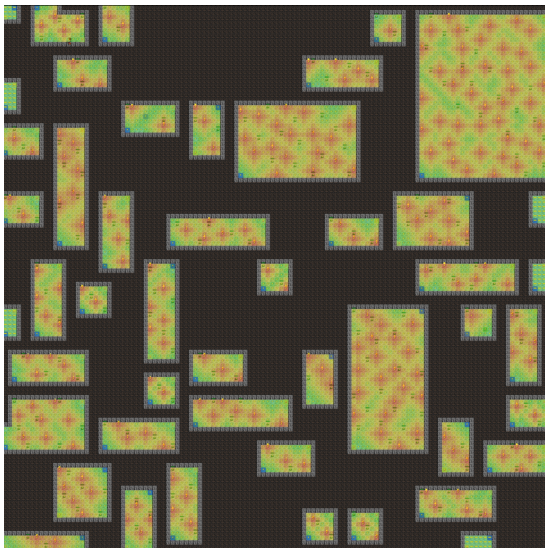
(a) Distância do teletransporte



(b) Distância da fogueira



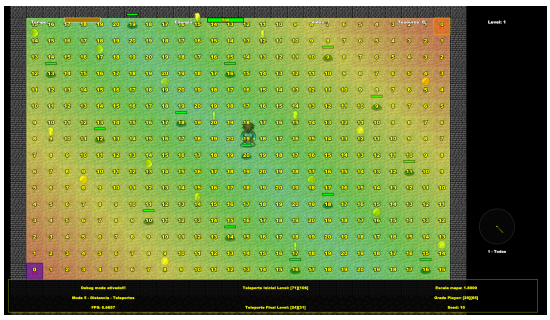
(c) Distância dos inimigos



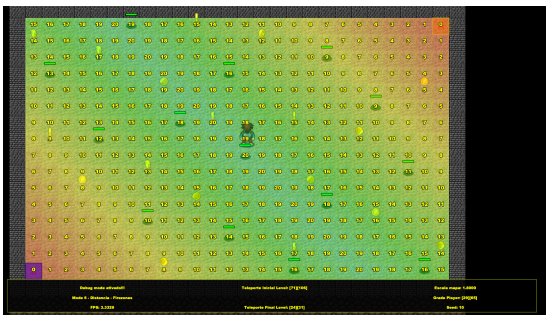
(d) Distância do tesouro

Figura 4.8: Análises com o conjunto de ladrilhos “Salas”.

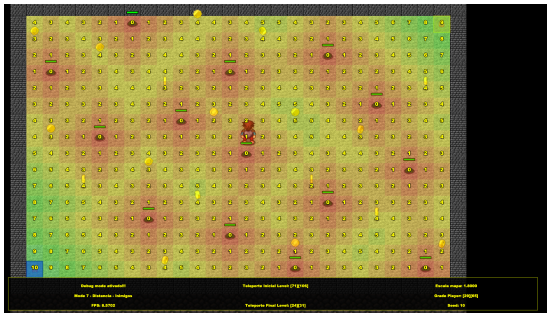
Neste cenário, os inimigos ficam bem distribuídos, evitando ficar muito próximos aos pontos de teletransporte, conforme pode ser observado nas Figuras 4.8c e 4.9c. O mesmo comportamento pode ser observado nas Figuras 4.8d e 4.9d para as moedas, que são posicionadas entre os espaços dos inimigos. Isso gera um bom espalhamento, permitindo que o jogador abra caminho em algum ponto ou mesmo tenha que cobrir toda a área, caso a coleta de todas as moedas ou extermínio de todos os inimigos sejam obrigatórios. Entretanto, o estilo repetitivo das salas do sistema de posicionamento poderia tornar tal tarefa monótona, pois há desafios na “ida” e não na “volta”. Os jogadores rapidamente



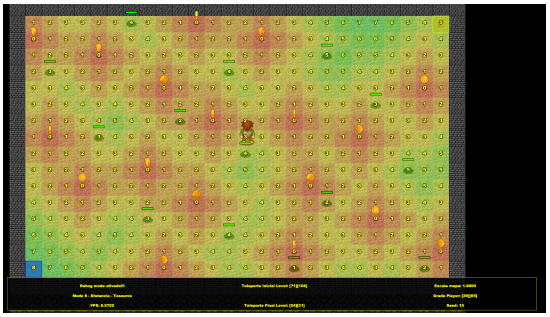
(a) Distância do teletransporte



(b) Distância da fogueira



(c) Distância dos inimigos



(d) Distância do tesouro

Figura 4.9: Análises em proximidade com o conjunto de ladrilhos “Salas”.

criariam padrões para fazer a varredura da sala e elementos extras de jogo deveriam ser adicionados para resolver isso, como pontos de renascimento ou barreiras internas como mobílias, pedras ou paredes.

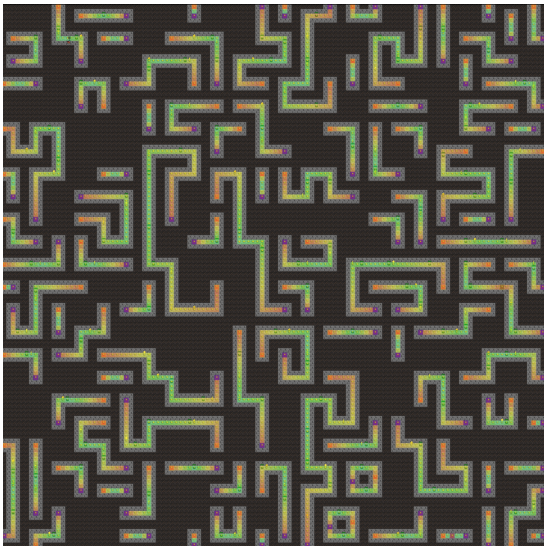
4.4.2 Cenário “Corredores”

No cenário para o conjunto de ladrilhos “Corredores”, o algoritmo gera uma sequência contínua, praticamente um caminho, pela falta de ladrilhos de bifurcações. Esse conjunto poderia ser utilizado para gerar situações claustrofóbicas, muito comuns em cenários de ficção científica como interiores de espaçonaves e bases, como pode ser visto nas Figuras 4.10a e 4.11a.

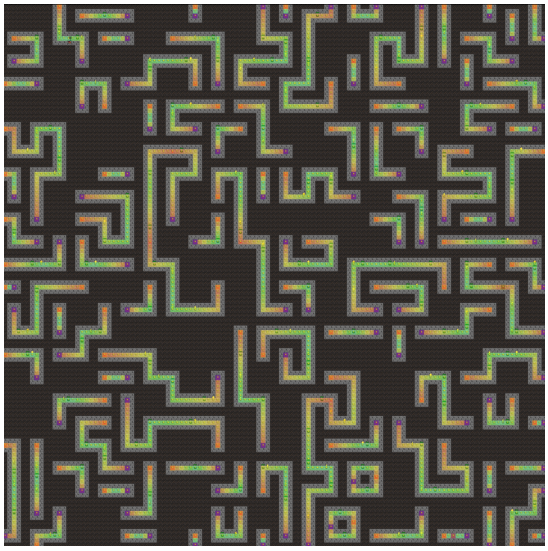
A movimentação é bem simplificada, pois tendo apenas um caminho para o teletransporte, o algoritmo de posicionamento novamente usa os extremos como pontos inicial e final da sala. Pelo caminho tão simples, as fogueiras estão sempre no caminho entre portais, como pode-se ver nas Figuras 4.10b e 4.11b. Com isso temos que a navegação é linear e o jogo provavelmente terá uma narrativa que estimula ou obriga percorrer o mapa em apenas uma direção.

Outro fator é que pelas salas serem tão estreitas é necessário enfrentar todos

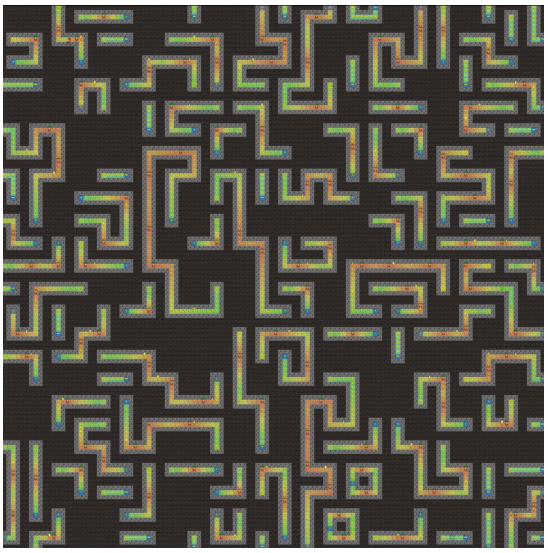
os inimigos que se encontra, não tendo como desviar (Figuras 4.10c e 4.11c). Coletar todos os tesouros é trivial, pois estarão em seu caminho, como pode se notar nas Figuras 4.10d e 4.11d. No estado atual da geração, os tesouros vão se alternar com os inimigos. Entretanto, pode acontecer de nenhum tesouro ou inimigo ser gerado nas salas muito pequenas e, novamente, novos elementos podem ser adicionados nessas situações.



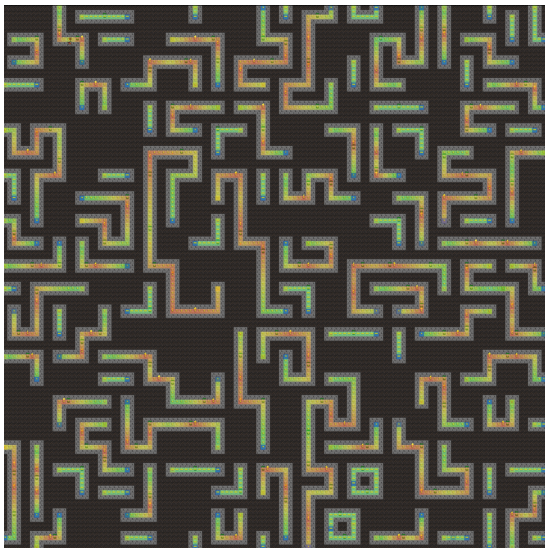
(a) Distância do teletransporte



(b) Distância da fogueira



(c) Distância dos inimigos



(d) Distância do tesouro

Figura 4.10: Análises com o conjunto de ladrilhos “Corredores”.

4.4.3 Cenário “Labirinto”

O conjunto de ladrilhos “Labirinto” pode ser visto como o conjunto “Corredor”, mas com ladrilhos de encruzilhada. Os mapas gerados apresentam vários caminhos alternativos e

(a) Distância do teletransporte

(b) Distância da fogueira

(c) Distância dos inimigos

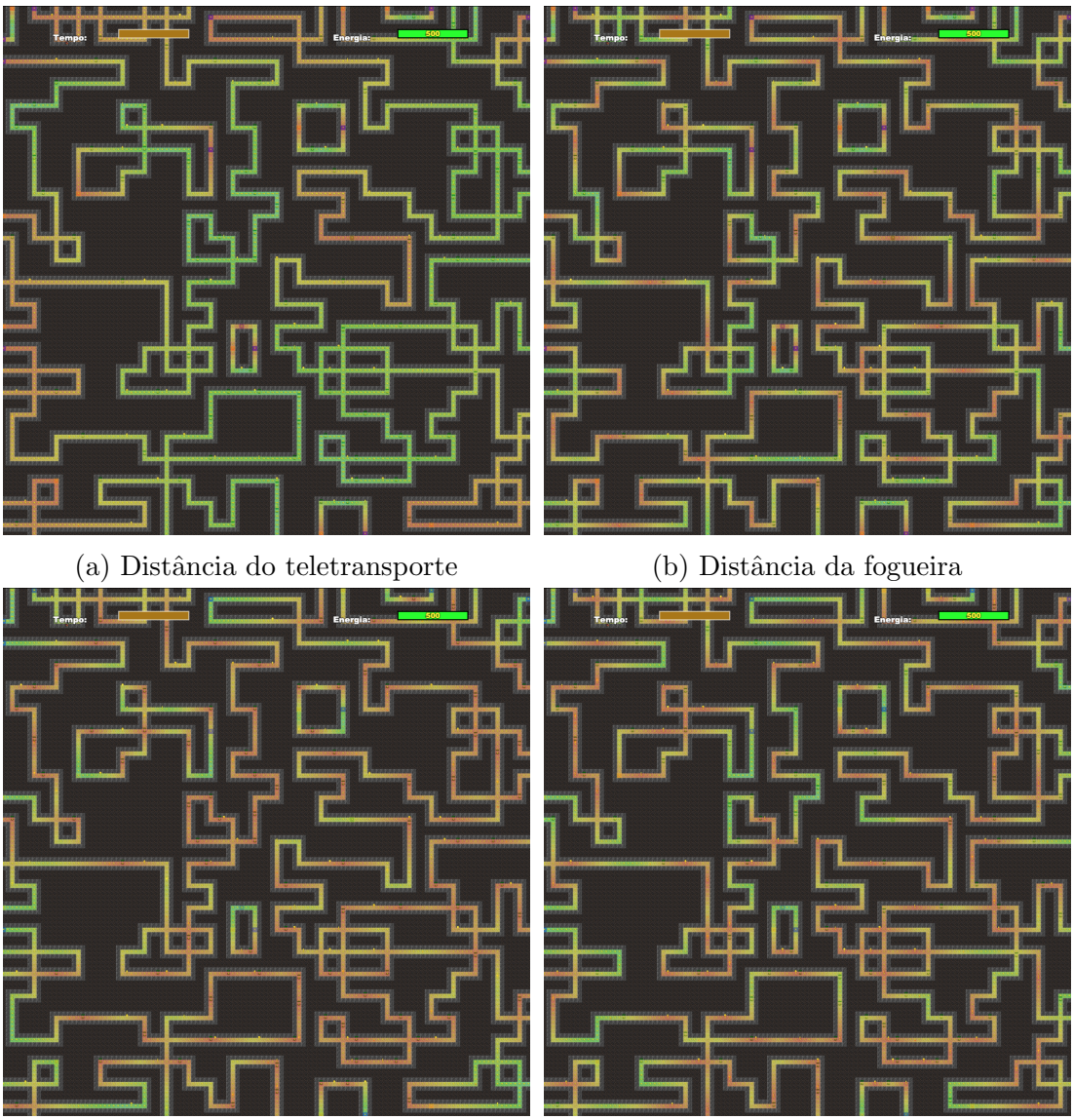
(d) Distância do tesouro

Figura 4.11: Análises em proximidade com o conjunto de ladrilhos “Corredores”.

as salas cobrem grandes áreas e regiões côncavas, como pode se observar nas Figuras 4.12 e 4.13.

Pela grande distância coberta, os teleportes podem não estar nos extremos, conforme alguns casos que podem ser observados nas Figuras 4.12a e 4.13a. Em função das grandes distâncias, as fogueiras são posicionadas ao longo das áreas, o que permite a navegação no labirinto, e mesmo que sua posição seja fixa, este foi o conjunto de ladrilhos no qual foi difícil de se direcionar e assim morrer por não encontrar uma fogueira.

Os inimigos estão distribuídos pelo labirinto com uma boa distribuição, mas com uma certa concentração em alguns pontos que exige uma investigação subsequente se foi efeito combinado de teletransportes e fogueiras (Figuras 4.12c e 4.13c). Como é possível desviar dos inimigos através das encruzilhadas e ciclos fechados, essa configuração dá opção de qual área o jogador pode escolher “limpar” e criar acesso aos tesouros. Os tesouros, por sua vez, também estão distribuídos ao longo dos caminhos, sem observar uma concentração diferente como com os inimigos (Figuras 4.12d e 4.13d).



(a) Distância do teletransporte (b) Distância da fogueira

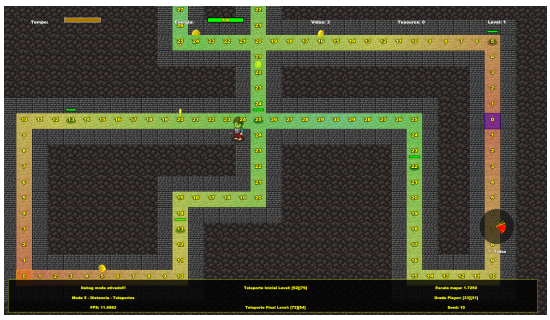
(c) Distância dos inimigos (d) Distância do tesouro

Figura 4.12: Análises com o conjunto de ladrilhos “Labirinto”.

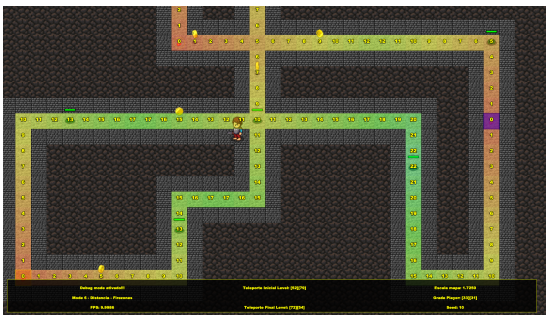
4.4.4 Cenário “Salas Conectadas”

O conjunto de ladrilhos “Salas Conectadas” é uma junção entre os conjuntos “Salas” e “Labirinto”, possuindo características compartilhadas entre os dois. As Figuras 4.14a e 4.15a mostram que ele possui as grandes regiões das salas e também as diversas passagens estreitas. Com isso a geração tem uma navegabilidade interessante, andando por corredores mais claustrofóbicos e por salas grandes.

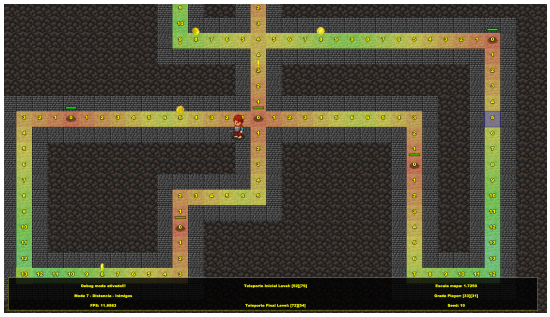
Assim como no “Labirinto”, longas distâncias são cobertas e com isso os teleportes podem não estar nos extremos, conforme pode ser observado na Figura 4.14a. Por cobrir espaços maiores, fogueiras são espalhadas pela geração, como é possível notar na



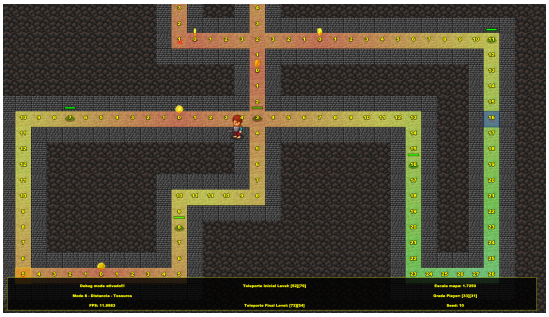
(a) Distância do teletransporte



(b) Distância da fogueira



(c) Distância dos inimigos



(d) Distância do tesouro

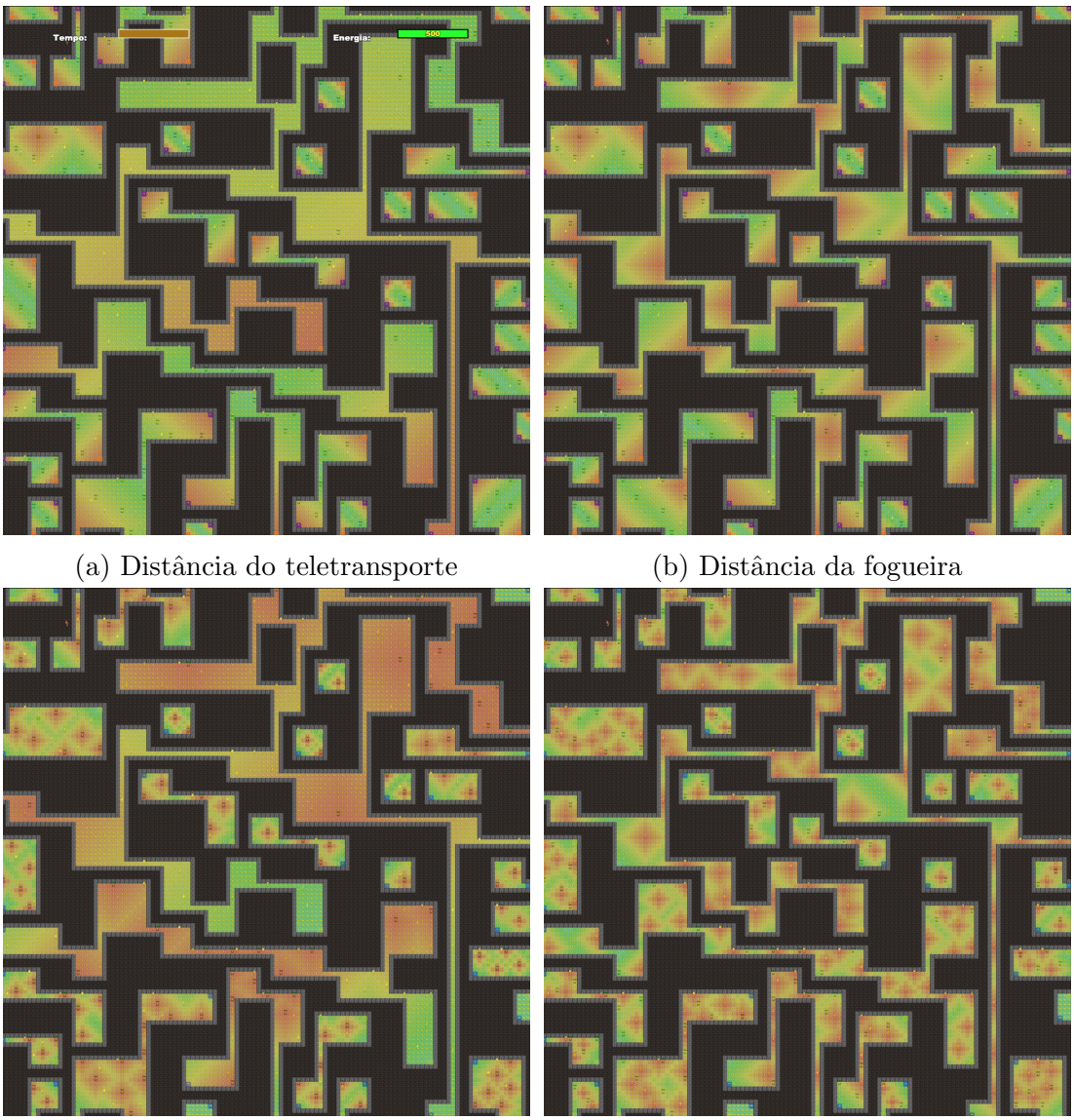
Figura 4.13: Análises em proximidade com o conjunto de ladrilhos “Labirinto”.

Figura **??**. Ao contrário do “Labirinto”, nessa geração estas são fáceis de se encontrar.

Também como no “Labirinto”, pode-se notar essa tendência dos inimigos evitarem alguns lugares, como é visível na Figura 4.14c. Por ser a geração com os maiores espaços, a escolha de quais espaços limpar em busca de tesouros é ainda mais relevante. Como é possível notar na Figura 4.14d, grandes espaços cheios de tesouros podem ser ignorados.

Através da WFC e manipulação dos conjuntos de ladrilhos, foi possível criar diferentes padrões para a geração de topologias. É importante ressaltar, entretanto, que há uma forte relação entre a montagem dos conjuntos e o resultado final: na presente implementação, a rotação é aplicada individualmente mesmo que o resultado seja uma cópia exata de outro ladrilho no conjunto (o que o torna mais um grupo do que um conjunto). Isso afeta as chances de sorteio durante a geração, pois o sorteio é feito neste grupo. Essa implementação é ingênua, mas permite ser expandida para, por exemplo, uma parametrização por Cadeias de Markov para a seleção.

Outra limitação na implementação é devida às restrições serem diretamente associadas com o lado do ladrilho. É possível realizar essas restrições por metadados ou as extrair diretamente de um exemplo de mapa, desde que o tamanho do ladrilho



(a) Distância do teletransporte (b) Distância da fogueira

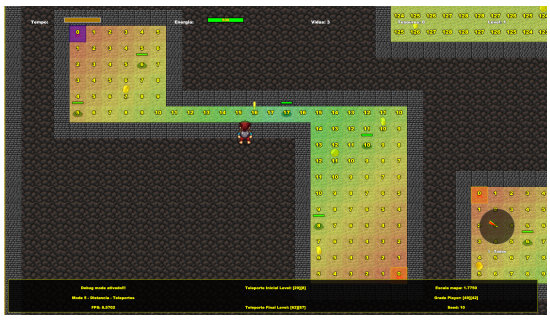
(c) Distância dos inimigos (d) Distância do tesouro

Figura 4.14: Análises com o conjunto de ladrilhos “Salas Conectadas”.

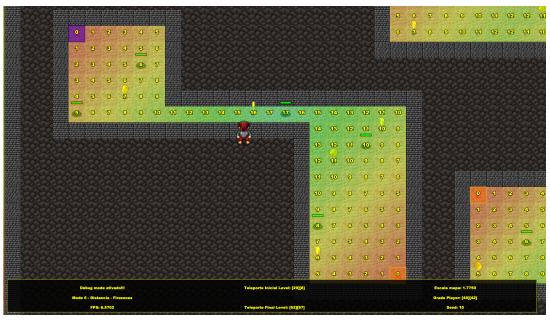
seja previamente definido. Entretanto, pela natureza de implementação do PCGLab, a abordagem pela comparação de vizinhança é suficiente, mas pode ter que ser revista se o modelo interno crescer em complexidade.

Pôde-se observar indícios da influência dos diferentes ladrilhos no fluxo de experiencia do Jogador, como no conjunto “Salas” que era repetitivo e assim cansativo e o conjunto “Salas Conectadas” que tem menos elementos previsíveis sendo assim menos cansativo.

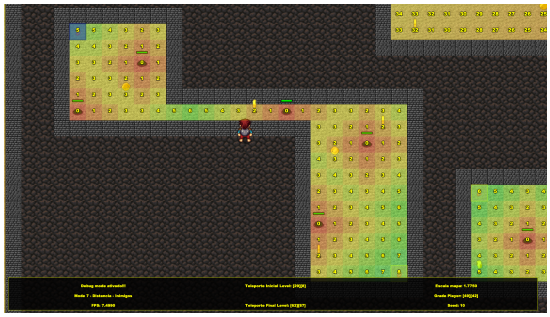
Este capítulo explicou o processo de implementação do WFC e como este foi integrado ao projeto PCGLab. Além disso, analisou os resultados da geração procedural por WFC. O próximo capítulo contém as considerações finais e planos para trabalhos



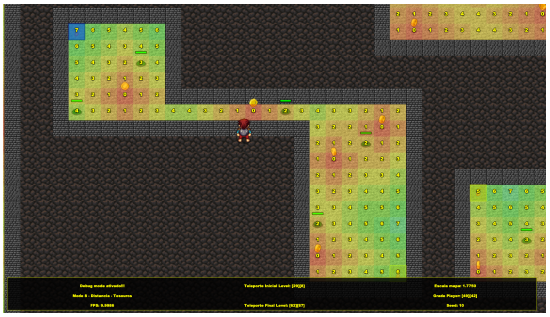
(a) Distância do teletransporte



(b) Distância da fogueira



(c) Distância dos inimigos



(d) Distância do tesouro

Figura 4.15: Análises em proximidade com o conjunto de ladrilhos “Salas Conectadas”.

futuros.

5 Considerações Finais

Este trabalho apresentou um esforço na busca em ligar a experiência do jogador com a geração procedural. Em específico usando *Wave Function Collapse* (WFC). Foi analisado como os diferentes tipos de conjuntos de ladrilhos podem influenciar na experiência do jogador. Utilizando o ambiente do PCGLab, o algoritmo WFC foi implementado, além de um software à parte para rápida prototipação dos conjuntos de ladrilhos. Os códigos produzidos aqui foram acoplados ao projeto original do PCGLab e estão disponíveis como código livre em JavaScript.

Na série de observações do resultado da geração, foram utilizados mapas de influência para avaliação de como a topologia gerada afeta no posicionamento dos elementos de jogo. Através das observações, teve-se indícios que é possível identificar o resultado da geração procedural na experiência de jogador. Mas mais estudos se fazem necessários para reforçar o laço entre o posicionamento de elementos de jogo e a escolha dos ladrilhos pelo WFC.

O objetivo de analisar a experiência do jogador pela geração do WFC foi parcialmente cumprido, pois o conceito é abstrato e mais experimentos devem ser realizados, cruzando dados de jogadores reais com as análises. Entretanto, acreditamos que o PCGLab fornece um ferramental para esse tipo de estudo.

5.1 Limitações e Trabalhos Futuros

A experiência de jogador é um aspecto abstrato, dependente de uma série de fatores e também das motivações pessoais que cada um tem ao jogar. A busca por um modelo de experiência é um problema complexo e deve ser construído em diversas etapas. Este trabalho faz uma análise argumentativa, mas uma subsequente coleta de opiniões de jogadores reais, quando o jogo do PCGLab estiver com um grau de desenvolvimento avançado, pode auxiliar em muito a identificação de problemas e melhorias na geração e construção do modelo de experiência.

Um objetivo futuro para o projeto é continuar seguindo a literatura do WFC e implementar um algoritmo que extrai um conjunto de ladrilhos a partir de uma imagem de exemplo.

No futuro se espera que o WFC seja usado para popular o mapa com inimigos e tesouros e comparar essa abordagem com o posicionamento atual que usa mapa de influência. Isso poderia ser feito colocando um novo elemento no ladrilho como um inimigo encostando num tesouro para garantir o desafio e recompensa.

Além disso, se espera que ao adicionar mais métodos de geração no projeto do PCGLab. Como uma gramática generativa, esta possa ser comparada ao WFC na habilidade de gerar topografia pouco previsível.

Bibliografia

ABT, C. C. *Serious games*. [S.l.]: Viking Press, 1970.

ALSTES, A. *Wang tiles for image and texture generation*. [S.l.], 2004.

CASTRO, M. L. A.; CASTRO, R. de O. Autômatos celulares: implementações de von neumann, conway e wolfram. *Revista de Ciências Exatas e Tecnologia*, v. 3, n. 3, p. 89–106, 2015.

CHENG, D.; HAN, H.; FEI, G. Automatic generation of game levels based on controllable wave function collapse algorithm. In: SPRINGER. *International Conference on Entertainment Computing*. [S.l.], 2020. p. 37–50.

COSTA, L. D.; KNOP, I. O. Geração procedural de conteúdo através de uma abordagem híbrida entre autômatos celulares e heurísticas. In: *Proceedings of SBGames 2020*. Juiz de Fora, MG, Brasil: SBC, 2020.

COSTA, L. D. da; KNOP, I. de O. *Autômatos celulares aplicados na geração procedural de conteúdo em jogos*. 2019. Apresentado nas Sessões Técnicas do VI Forum Acadêmico de Estudos Lúdicos, Rede Brasileira de Estudos Lúdicos, São Paulo. Disponível em: ⟨https://www.rebel.org.br/fael6artigos/⟩.

CSIKSZENTMIHALYI, M. et al. *Flow*. [S.l.]: New York: Harper & Row, 1990.

DJAOUTI, D. et al. Origins of serious games. In: *Serious games and edutainment applications*. [S.l.]: Springer, 2011. p. 25–43.

DONALD, M. *Superpositions, Sudoku, the Wave Function Collapse algorithm*. 2020. Disponível em: ⟨https://www.youtube.com/watch?v=2SuvO4Gi7uY⟩.

FITZPATRICK, R. *Introductory Quantum Mechanics*. University of Texas at Austin, 2020. Cited: Ago. 2021. Disponível em: ⟨https://phys.libretexts.org/{{\spacefactor\@m{}}}go/page/1⟩.

GUMIN, M. *Bitmap and tilemap generation from a single example by collapsing a wave function*. 2016. Disponível em: ⟨https://github.com/mxgmn/WaveFunctionCollapse⟩.

HUIZINGA, J. *Homo ludens: o jogo como elemento da cultura*. [S.l.]: Editora da Universidade de S. Paulo, Editora Perspectiva, 1971. v. 4.

JOHNSON, L.; YANNAKAKIS, G.; TOGELIUS, J. Cellular automata for real-time generation of. 09 2010.

KARTH, I.; SMITH, A. M. WaveFunctionCollapse is Constraint Solving in the Wild. In: *Proceedings of the 12th International Conference on the Foundations of Digital Games*. New York, NY, USA: Association for Computing Machinery, 2017. (FDG '17). ISBN 9781450353199. Disponível em: ⟨https://doi.org/10.1145/3102071.3110566⟩.

LEWIS, J. E. C. L. P. Homo ludens revisited. *Yale French Studies*, JSTOR, n. 41, p. 31–57, 1968.

MARK, D. Modular tactical influence maps. *Game AI Pro*, v. 2, p. 343, 2015.

MICHAEL, D. R.; CHEN, S. L. *Serious games: Games that educate, train, and inform*. [S.l.]: Muska & Lipman/Premier-Trade, 2005.

SALEN, K.; TEKINBAŞ, K. S.; ZIMMERMAN, E. *Rules of play: Game design fundamentals*. [S.l.]: MIT press, 2004.

SHAKER, N.; TOGELIUS, J.; NELSON, M. J. *Procedural Content Generation in Games*. 1st. ed. [S.l.]: Springer International Publishing, 2016. ISBN 9783319427164.

SHORT, T. X.; ADAMS, T. *Procedural Generation in Game Design*. 1st. ed. USA: A. K. Peters, Ltd., 2017. ISBN 1498799191.

TEAM, O. E. L. et al. *Open-Ended Learning Leads to Generally Capable Agents*. 2021.

TOGELIUS, J. et al. What is procedural content generation?: Mario on the borderline. In: ACM. *Proceedings of the 2nd international workshop on procedural content generation in games*. [S.l.], 2011. p. 3.

TOZOUR, P. Influence mapping. *Game programming gems*, v. 2, p. 287–297, 2001.