INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS TERESINA CENTRAL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RAFAEL RIBEIRO DA SILVA

AUTÔMATOS CELULARES EM GERAÇÃO PROCEDURAL: maximizando a conectividade em cavernas na Godot Engine

TERESINA, PIAUÍ
2026

RAFAEL RIBEIRO DA SILVA

AUTÔMATOS CELULARES EM GERAÇÃO PROCEDURAL: maximizando a conectividade em cavernas na Godot Engine

Trabalho de Conclusão de Curso (Artigo Científico) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina Central.

Orientador: Prof. M^{e}. Francisco Marcelino Almeida de Araújo

TERESINA, PIAUÍ
2026

AUTÔMATOS CELULARES EM GERAÇÃO PROCEDURAL: maximizando a conectividade em cavernas na Godot Engine

Rafael Ribeiro da Silva[1]
Francisco Marcelino Almeida de Araújo[2]

RESUMO

Um dos métodos mais utilizados pela indústria de jogos para a geraçâo de cavernas são os autômatos celulares, mas que se não tratados de maneira adequada, podem criar ambientes que não servem para o contexto de jogos. Para isso o presente trabalho tem o objetivo de analisar e otimizar os parâmetros utilizados nesse sistema de geração de forma a aumentar a taxa de conectividade das áreas navegáveis mantendo a estética desejável de cavernas orgânicas. Foram realizados testes na Godot Engine, variando a taxa de preenchimento inicial e o número de iterações, analisados posteriormente por um algoritmo de Flood-fill. Foi descoberta que taxas de preenchimento mais altas aumentam facilmente a conectividade, mas falham no requisito de ambiente de jogo, criando áreas limpas e quase sem obstáculos. Descobriu-se então que o melhor cenário seriam taxas de preenchimento mais baixas com um maior número de iterações, pois mesmo que a porcentagem de áreas conectadas seja menor, garante a estética e o desafio desejado para um jogo, e que as áreas mortas podem ser elimitadas através de um método de poda, criando um mapa com uma área totalmente navegável com baixo custo de processamento de CPU.

Palavras-chaves: geração procedural; autômatos celulares; level design; Godot Engine; flood-fill.

ABSTRACT

One of the most widely used methods by the gaming industry for cave generation is cellular automata; however, if not applied appropriately, they can create environments that are unsuitable for a gaming context. Therefore, the present study aims to analyze and optimize the parameters used in this generation system to increase the connectivity rate of navigable areas while maintaining the desirable aesthetics of organic caves. Tests were conducted in the Godot Engine by varying the initial fill rate and the number of iterations, with the results subsequently analyzed by a Flood-fill algorithm. It was discovered that higher fill rates easily increase connectivity but fail to meet the requirements of a game environment, creating wide-open areas with almost no obstacles. It was then concluded that the ideal scenario consists of lower fill rates combined with a higher number of iterations, as this ensures the desired aesthetics and challenge for a game, even if the initial percentage of connected areas is lower. Furthermore,

[1] Graduando em Análise e Desenvolvimento de Sistemas pelo Instituto Federal do Piaui (IFPI). E-mail: rafaelrsilva.devgmail.com.

[2] Doutorando em Ciência da Computação (UFMA/UFPI). Especialista em Educação Digital (Centro Universitário SENAI/SC). Mestre em Biotecnologia (UFPI). Graduado em Engenharia Elétrica (UESPI). Professor efetivo e coordenador do grupo de pesquisa LABIRAS no Instituto Federal do Piauí (IFPI) - Campus Teresina Central. E-mail: francisco.marcelino@ifpi.edu.br.

dead areas can be eliminated through a pruning method, delivering a fully navigable map with a low CPU processing cost.

Keywords: procedural generation; cellular automata; level design; Godot Engine; flood-fill.

Data de Aprovação: 25 de Junho de 2026

1 INTRODUÇÃO

Com a aceitação e evolução do mercados de jogos, diferentes gêneros passaram a se estabelecer e conquistar números cada vez maiores de jogadores, entre eles podemos citar o *Roguelike*, de acordo com Hannula (2025, p. 34, tradução nossa) nesse tipo de jogo, a aleatoriedade aprimora a experiência de jogo porque proporciona variedade e experiências únicas a cada partida, um dos elementos de destaque para essa sensação é o ambiente.

Para suprir a crescente demanda por complexidade nos jogos modernos sem elevar os custos de produção de forma insustentável, o mercado adotou técnicas de Geração Procedural de Conteúdo (GPC). Rocha & Prada (2025, p. 545, tradução nossa) destacam que essa abordagem se tornou um padrão tanto na pesquisa quanto na indústria, facilitando o processo de criação de conteúdo, reduzindo os custos e destacando-se, em especial, como a solução mais eficiente para a geração automatizada de níveis de jogo.

Uma das formas criadas para GPC são os autômatos celulares, segundo Paes, Junio & Rodriguez (2025, p. 211, tradução nossa) essa técnica é largamente utilizada em jogos *Roguelike* para gerar cavernas e estruturas de labirintos e se provou adequada para representar salas jogáveis. Ele define os autômatos celulares como:

> "uma matriz bidimensional onde as células são atualizadas iterativamente com base em regras de vizinhança. Os parâmetros de entrada permitem definir a densidade das paredes e a configuração espacial dos elementos, contribuindo diretamente para a flexibilidade e rejogabilidade do jogo."(PAES; JUNIO; RODRIGUEZ, 2025, p. 211, tradução nossa)

Segundo Zhang, Zhang & Huang (2022, p. 189, tradução nossa) algoritmos de geração procedural não conseguem garantir a jogabilidade de conteúdo gerados para jogos. Um dos problemas encontrados é a geração de ilhas isoladas, locais onde o jogador não consegue acessar e pode ficar preso.

Diante deste cenário, o presente trabalho tem o objetivo de analisar o impacto da parametrização da taxa de preenchimento e o número de iterações em um algoritmo de autômatos celulares. Conforme destacado por Salinas (2024, p. 15, tradução nossa), a manipulação dessas variáveis é um fator fundamental para equilibrar a complexidade e a conectividade em autômatos celulares. Descobrindo uma configuração que maximize

a conectividade do mapa gerado, diminuindo o número de ilhas isoladas evitando comprometer o tempo de execução da CPU.

Para testar as gerações e verificar a existência de cavernas isoladas, existem algoritmos específicos, como o *Flood-fill*. Segundo Shuaeb, Kamruzzaman & Ali (2021, p. 16, tradução nossa), o algoritmo de *Flood-fill* é simples e eficiente para precisão, sendo um algoritmo de inundação, ele é capaz de preencher áreas inteiras, o que é bastante útil no contexto deste trabalho.

1.1. TRABALHOS RELACIONADOS

A GPC é amplamente explorada em diversas abordagens, com o objetivo de reduzir o esforço manual no desenvolvimento de jogos. Dias *et al.* (2025, tradução nossa) propõe um algoritmo focado na propagação de energia que se destaca por operar em tempo linear $\mathcal{O}(n)$. Contudo, os autores admitem ter evitado intencionalmente análises de usabilidade e aceitação, o que não leva em conta a avaliação estética e o impacto prático no *Level Design*.

Relacionado a métodos de busca, So *et al.* (2024, tradução nossa) utilizam Algorítmos Genéricos (GA) para otimizar o posicionamento de salas e o balanceamento de inimigos em *dungeons*. Embora a técnica seja eficaz na distribuição de espaço, a solução necessita de centenas de gerações para convergir. Isso a torna um método de alto custo computacional, inadequado para geração em tempo de carregamento.

Salinas (2024, tradução nossa) avalia cinco métodos clássicos de geração de cavernas bidimensionais e conclui que os Autômatos Celulares oferecem o melhor equilibrio geral para ambientes orgânicos. Entretanto, a pesquisa aponta como limitação a natureza probabilística do método, que frequentemente resulta em cavernas isoladas e inacessíveis. Embora o autor sugira que recursos adicionais sejam necessários para conectar as áreas, uma solução prática e otimizada não é consolidada em seu escopo.

O presente trabalho diferencia-se ao preencher a lacuna deixada por essas abordagens. Em vez de sobrecarregar o processamento com algoritmos de busca complexos ou ignorar as ilhas isoladas geradas pelos Autômatos Celulares, propõe-se uma geração subótima intencional aliada a um método de pós-processamento de poda via *Flood-fill*. Além disso, ao restringir a validação espacial à Vizinhança de Von Neuman, evita-se a ocorrência de falsos-positivos diagonais. Essa metodologia assegura mapas navegáveis e esteticamente adequados a um *Level Design*. Para materializar essa eficiência e garantir que o custo de processamento atenda às exigências da indústria, a escolha de desenvolver a solução na Godot Engine, utilizando sua linguagem nativa GDScript, torna-se estrategicamente justificada. O GDScript possui integração direta com o gerenciamento de memória do motor gráfico, permitindo executar as operações iterativas em matrizes bidimensionais com máxima performance. Dessa forma, a pesquisa transcende a validação teórica para entregar uma solução técnica leve,

autônoma e imediatamente aplicável, estabelecendo as fundações operacionais para a metodologia detalhada a seguir.

2 DESENVOLVIMENTO

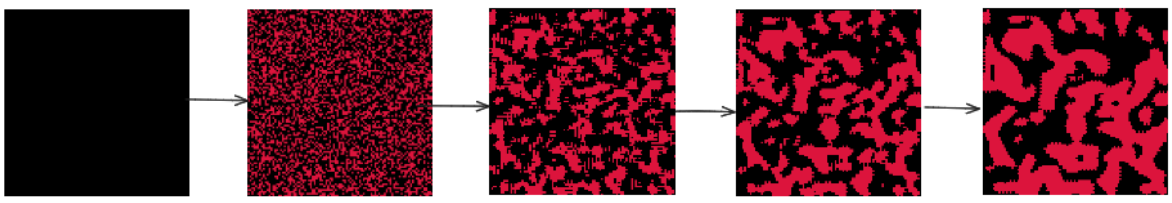
O desenvolvimento deste trabalho fundamenta-se na adaptação de modelos matemáticos para a criação de cenários virtuais, explorando a transição de matrizes aleatórias para estruturas orgânicas complexas. Nas subseções a seguir, detalha-se o funcionamento técnico dos autômatos celulares utilizanco a Vizinhança de Moore, os problemas críticos de isolamento espacial que comprometem a navegabilidade do jogador e os mecanismos de validação estrutural via algoritmo de *Flood-fill*, fundamentais para assegurar a jogabilidade do conteúdo gerado.

2.1. AUTÔMATOS CELULARES E A REGRA DE VIZINHANÇA

Segundo Weisstein (2024, tradução nossa), um autômato celular é uma coleção de células em uma grade que evoluem seguindo regras definidas com base no estado das células, utilizando uma série de passos de tempo. Para o nosso teste cada célula possui dois estados, zero significando chão, ou seja, por onde o jogador pode passar, representado pela cor vermelha e um significando as paredes que limitam o ambiente navegável, representado pela cor preta, conforme pode ser visto na Figura 1.

No início do algoritmo, cada célula recebe o estado de parede com base em uma probabilidade predefinida, que será chamada de taxa de preenchimento inicial criando um ruído inicial. A partir daí são aplicados ciclos de iteração que serão responsáveis por suavizar esse ruído, aproximando de formações parecidas com cavernas com um aspecto mais orgânico.

Figura 1 – Demonstração de avanço do ciclo de iterações



Fonte: Produzido pelo autor (2026).

A lógica da mudança de estados das células utiliza a vizinhança de Moore (N_M). Segundo Kari (2005, p.5, tradução nossa), esta vizinhança é definida formalmente pela norma do máximo ($\|y\|_\infty \leq 1$), o que resulta, em um espaço bidimensional ($d = 2$), em um total de 3^d células, o que equivale a 9 posições, incluindo a célula central. Na

prática da geração de cavernas, isso significa avaliar as oito células adjacentes, tanto ortogonais quanto diagonais, para determinar a evolução do estado da célula central.

Entretanto, utilizando apenas essa lógica de forma descontrolada algumas das células poderiam ficar isoladas reduzindo a aparência de caverna e se aproximando mais a um efeito de esponja, para resolver isso foi utilizado uma regra que favorecesse a formação de cavernas, foi definido então que o valor mínimo e o máximo de células adjacentes encontradas para definir o valor da central seria quatro e oito respectivamente, ou seja, se uma célula possuir entre quatro e oito adjacentes que são paredes, ela também será uma parede, diminuindo o número de células isoladas. Segundo Johnson, Yannakakis & Togelius (2010, tradução nossa) esse algoritmo de deslocamento tem como principais vantagens a facilidade de gerar mapas verossímeis e o fato de serem tipicamente muito computacionalmente eficientes.

2.2. O PROBLEMA DO ISOLAMENTO ESPACIAL E A NAVEGABILIDADE

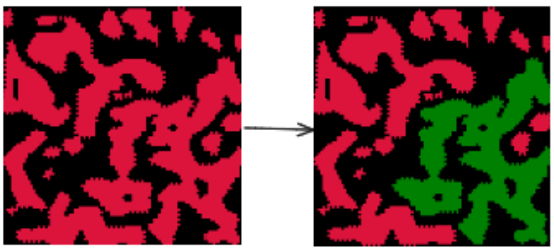
Um problema típico sempre ocorre quando utilizamos automatos celulares, a formação de ilhas isoladas, conforme relatado por Yannakakis & Togelius (2025, p. 166, tradução nossa). Algumas células se agrupam e formam cavernas mas que acabam não se conectando umas as outras, criando áreas inacessíveis ao jogador e consequentemente, sendo um custo de processamento desperdiçado.

Além disso, como explicado anteriormente o algoritmo utiliza a lógica de oito direções o que pode causar a criação de duas ilhas que matematicamente estão conectadas, mas que para a lógica de um ambiente navegável é impossível, esse caso seria a geração de ilhas conectadas por células diagonais. A geração dessas diagonais cria um falso-positivo, onde aparentemente duas áreas estão conectadas, mas são instransitáveis.

2.3. VALIDAÇÃO ESTRUTURAL COM O ALGORITMO FLOOD-FILL

Para verificar a conectividade das áreas da caverna gerada, principalmente após a constatação dos falsos-positivos, fez-se necessária uma verificação pós-geração, foi então utilizado o algoritmo de *Flood-fill*. De acordo com Hearn & Baker (2004, p. 205, tradução nossa) este procedimento inicia-se a partir de uma coordenada interna específica e percorre as posições adjacentes, reatribuindo os valores de todas as áreas que compartilham o estado inicial desejado. Aplicado ao contexto deste trabalho, o algoritmo seleciona uma célula de chão aleatória e mapeia iterativamente toda a extensão conectada daquela caverna, aqui representada pela cor verde, conforme a Figura 2:

Figura 2 – Demonstração do algoritmo de *Flood-fill*



Fonte: Produzido pelo autor (2026).

Ainda conforme Hearn & Baker (2004, p. 205, traduçao nossa), o Flood-fill pode operar através de abordagens de 4 ou 8 conexões. Para evitar que durante o teste de validação ocorresse o mesmo problema da conectividade matemática utilizando as diagonais, o algoritmo foi configurado para uma abordagem de 4 conexões, baseada na vizinhança de Von Neumann (N_{vN}). De acordo com Kari (2005, p. 5, tradução nossa), essa vizinhança é definida através da norma de Manhattan ($\|y\|_1 \leq 1$), o que limita os vizinhos de uma célula em um plano $2D$ apenas aos deslocamentos ortogonais (cima, baixo, esquerda e direita). Essa restrição é fundamental para evitar os falsos-positivos gerados por conexões diagonais, garantindo que o algoritmo valide apenas caminhos onde há compartilhamento de arestas entre as células de chão, imitando os movimentos possíveis para o jogador.

Sob a ótica da teoria dos grafos, o uso do *Flood-fill* justifica-se pelo objetivo da análise, mensurar a área da caverna através da Rotulagem de Componentes Conexos (*Connected Component Labeling*), e não traçar rotas (*pathfinding*). Assim, algoritmos de roteamento como o A* gerariam custo computacional desnecessário.

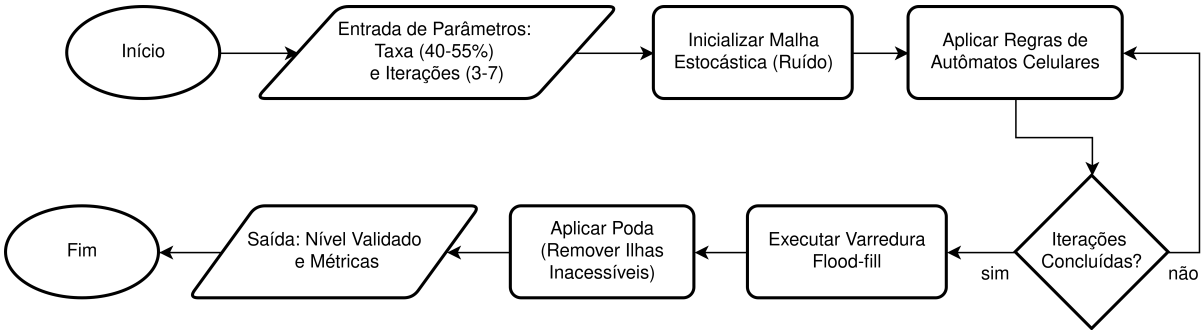
O *Flood-fill* foi implementado por meio de uma Busca em Profundidade (*Depth-First Search*) iterativa baseada em pilha (*stack*). Essa abordagem evita o risco de *stack overflow* associado a implementações recursivas e apresenta baixo consumo de memória prática quando comparada à Busca em Largura (*Breadth-First Search*), especialmente em mapas extensos. Além disso, a identificação da região navegável ocorre em tempo linear ($O(V + E)$).

3 METODOLOGIA

Para quantificar o impacto das variáveis de geração na conectividade das cavernas, estabeleceu-se um protocolo experimental controlado que isola o desempenho lógico do motor gráfico das restrições de processamento visual. O fluxo metodológico adotado, detalhado na Figura 3, inicia-se com o processamento de parâmetros de entrada que refinam a malha estocástica via ciclos de autômatos celulares, culminando no

uso do algoritmo Flood-fill como filtro de qualidade para assegurar a navegabilidade e eliminar estruturas descontinuadas decorrentes do método probabilístico. A seguir, descrevem-se as especificações do ambiente, as variáveis de controle e a automação do *benchmark* utilizada para mitigar vieses decorrentes da pseudoaleatoriedade algorítmica.

Figura 3 – Fluxograma da metodologia de geração e validação



Fonte: Produzido pelo autor (2026).

3.1. AMBIENTE E VARIÁVEIS

Para o presente trabalho o ambiente de desenvolvimento escolhido foi o motor de jogos Godot Engine (versão 4.6.1) e a linguagem de programação nativa GDScript, tendo esta um curto tempo de execução (Godot Engine, 2026). Para os testes foi utilizada uma máquina com um processador Intel de quatro núcleos e oito threads, 16 gigabytes de memória RAM operando sob um sistema operacional Linux.

Para manter um padrão e facilitar a coleta de resultados, o tamanho da grade foi fixado em 100x100 e como dito anteriormente, os valores mínimos e máximos de propagação foram definidos em quatro e oito respectivamente, visando favorecer a formação de estruturas semelhantes a cavernas rochosas.

Tendo o ambiente controlado definido, o experimento foi baseado na manipulação de duas variáveis. A taxa de preenchimento inicial, responsavel pela probabilidade de nascimento das células no estado de parede, testando os valores de 40%, 45%, 50% e 55%. Este intervalo tem como base o valor referencial de 50% testado por Johnson, Yannakakis & Togelius (2010, p. 3, tradução nossa), permitindo uma exploração empírica das fronteiras de conectividade e densidade da malha.

Paralelamente, o número de iterações, sendo este o número de ciclos de suavização aplicados, com os valores 3, 5 e 7 ciclos sucessivos. Partindo do teste de Johnson, Yannakakis & Togelius (2010, p. 3, tradução nossa) de 4 iterações para a estabilização topográfica, a amplitude escolhida neste trabalho visa identificar o limite de convergência onde o algoritmo atinge a estagnação geométrica e o processamento adicional torna-se redundante para a CPU. O cruzamento dessas variáveis resultou em 12 cenários de experimentação.

Para medir a eficácia dos cenários, duas variáveis serão verificadas, o tempo de execução, medido em milissegundos (ms), que mostra o tempo necessário para a execução do cenário e a porcentagem de conectividade navegável, que mostra o percentual da área total que o *Flood-fill* detectou de células chão conectadas na ilha principal.

3.2. BENCHMARK

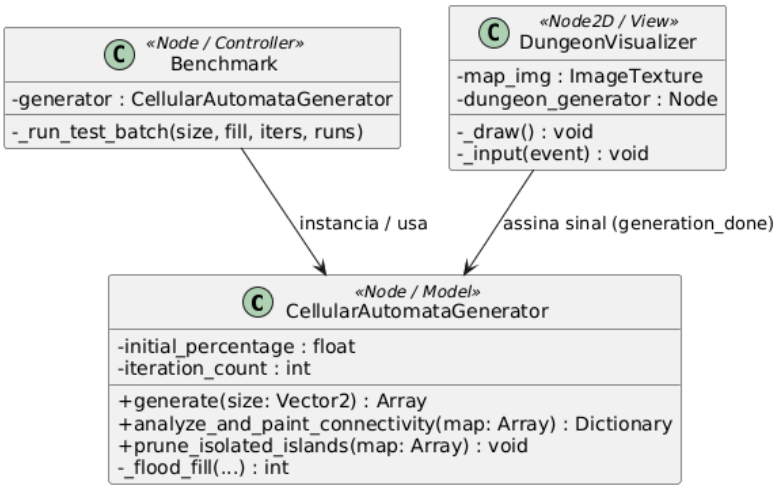
Para que a geração visual não impactasse a performance da máquina durante os testes, o algoritmo de automação do *benchmark* foi criado em modo de execução cega, o gerador foi instanciado na memória e isolado das rotinas de desenho de pixel. De acordo com Akenine-Möller *et al.* (2018, p. 786, tradução nossa), a abordagem mais eficaz para testar os limites de processamento da Unidade Central de Processamento (CPU) em aplicações visuais é suprimir a carga de trabalho da Unidade de Processamento Gráfico (GPU), garantindo que o processamento lógico do motor seja o único gargalo medido. evitando que o consumo de processamento gráfico da GPU impacte a medida de desempenho lógico na CPU.

Por fim, como é utilizado algoritmo com pseudoaleatoriedade, é possível a geração de um cenário bastante eficiente, ou um que seja impossível de navegar seguindo a lógica de um jogador. Para diminuir as chances de um viés na geração dos cenários o algoritmo de *benchmark* realiza cinquenta execuções independentes para cada um dos doze cenários testados. A análise dos resultados deste trabalho se baseia na média aritmética dos valores obtidos nessas cinquenta rodadas.

3.3. ARQUITETURA DO SOFTWARE E OTIMIZAÇÃO NA GODOT ENGINE

Sob a ótica da Engenharia de Software, o projeto adotou uma arquitetura focada na separação de responsabilidades. A lógica central foi encapsulada na classe autônoma *CellularAutomataGenerator*, viabilizando testes de *benchmark* em formato *headless* (sem renderização), garantindo métricas puras de CPU. O componente *DungeonVisualizer* atua desacoplado, recebendo dados através do padrão de sinais adotado pelo motor gráfico. Para otimizar a renderização de matrizes densas, optou-se pela manipulação direta de pixels em memória (*ImageTexture*) ao invés do sistema nativo de *TileMap*, eliminando sobrecarga de instanciamento e colisão física. A Figura 4 ilustra o diagrama de classes e o fluxo de comunicação dessa arquitetura.

Figura 4 – Diagrama de classes e separação de responsabilidades na Godot Engine



Fonte: Produzido pelo autor (2026).

4 RESULTADOS

A análise dos resultados reflete a busca pelo equilíbrio entre a precisão matemática e a viabilidade artística necessária ao desenvolvimento de jogos. A discussão desdobra-se em três pontos, sendo eles, a avaliação quantitativa das métricas de conectividade obtidas, o impacto visual e estético das diferentes densidades de preenchimento na percepção de *Level Design* e, por fim, a mensuração do custo computacional para a CPU, validando a aplicação prática do modelo em cenários reais.

4.1. ANÁLISE QUANTITATIVA DE CONECTIVIDADE E DESEMPENHO

A execução do teste gerou métricas precisas que facilitam a análise dos cenários e a viabilidade de cada um, os dados das cinquenta execuções para cada cenário estão dispostos na Tabela 1.

Observando-se os dados obtidos, é possível notar uma correlação direta entre a taxa de preenchimento inicial e a taxa de conectividade. Os cenários de maior preenchimento (50% e 55%) facilmente atingiram 99% e 100% de conectividade independentemente do número de iterações, enquanto as taxas menores, principalmente o cenário de 40%, precisaram de mais ciclos de suavização.

Se observado o caráter puramente matemático, o melhor cenário seria o que possui taxa de preenchimento de 55%, porque até mesmo com o menor número de iterações a conectividade é bastante alta. Mas tendo um caráter de aplicação prática em um jogo, outro fator deve ser analisado para se chegar a um resultado adequado.

Tabela 1 – *Benchmark* dos cenários de geração de autômatos celulares

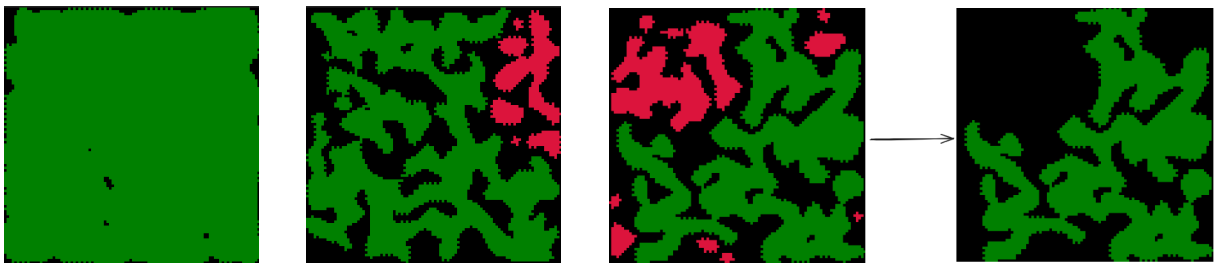
Preenchimento (%)	Nº de Iterações	Tempo Médio (ms)	Conectividade (%)
40%	3	153.12 ms	36.61%
40%	5	338.00 ms	57.42%
40%	7	486.48 ms	65.53%
45%	3	229.52 ms	95.38%
45%	5	372.60 ms	98.17%
45%	7	545.66 ms	99.17%
50%	3	272.38 ms	99.80%
50%	5	417.26 ms	99.91%
50%	7	500.04 ms	99.97%
55%	3	242.88 ms	99.96%
55%	5	362.12 ms	100.0%
55%	7	501.64 ms	100.0%

Fonte: Produzido pelo autor (2026).

4.2. A ESTÉTICA E O NÍVEL DE DENSIDADE

Quando se analisa visualmente os cenários testados, pode-se notar que o modelo visual contraria a eficiência aparente dos cenários de alta conectividade, conforme mostra a Figura 5 (a). Os cenários de maior conectividade sofrem de um fenômeno de excesso de densidade.

Figura 5 – Painel comparativo de topologias procedurais em diferentes parametrizações. (a) Desintegração estrutural e formação de campo aberto sob taxa de 55%; (b) Densidade estrutural equilibrada e geração de *choke points* sob taxa de 40%; (c) Isolamento topológico final após aplicação de *Flood-fill* e poda.



(a) Taxa de 55% (b) 40% 7 iterações (c) 40% (Processo de Poda)

Fonte: Produzido pelo autor (2026).

Como demonstrado, o algoritmo de *Flood-fill* consegue preencher praticamente toda a área porque o autômato celular fundiu quase todas as paredes em chão, resultando em um enorme campo aberto com quase nenhum obstáculo. Como dito anteriormente, matematicamente esse cenário atende aos requisitos, mas tendo este trabalho o objetivo de utilizar essas gerações em um ambiente de jogo, do ponto de vista de *Level Design*, ele falha em entregar um proporção equilibrada entre área navegável e massa

estática, privando o mapa dos pontos de estrangulamento (*choke points*) necessários para causar a sensação de desafio ao jogador.

Segundo Totten (2019, p. 32, tradução nossa) um espaço criado no contexto de *Level Design* deve ter algumas características como o desenvolvimento de linhas de visão, condições de iluminação, sombra e penumbra, exploração, orientação, ritmos espaciais e até mesmo como fazer com que espaços épicos sejam ainda mais épicos.

Por outro lado, cenários com parâmetros menores demonstraram uma fidelidade visual maior em relação ao formato desejado. Conforme pode ser observado na Figura 5 (b), o cenário com taxa de preenchimento de 40% e o total de 7 ciclos de suavização gerou uma rede de corredores seguimentados por pontos de estrangulamento bem definidos e rotas visualmente orgânicas.

Observando os resultados desse cenário, nota-se que ele obteve uma conectividade de apenas 65,53%, ou seja, cerca de 35% da área navegável não ficou conectada à ilha principal, o que pode ser resolvido.

Após o *Flood-fill* detectar a maior ilha e definir a área jogável, pode-se utilizar uma estratégia de pós-processamento de poda na matriz do mapa. O algoritmo então percorre a matriz sobrescrevendo as células marcadas como chão, mas que não estão conectadas à ilha principal, ou seja, áreas mortas, em células de parede. Isso evita áreas inacessíveis ao jogador que seriam custo desnecessário ao processamento, como demonstrado na Figura 5 (c)

Essa abordagem caracteriza o melhor dos dois mundos, pois aproveita a estrutura orgânica gerada, com a maior área conectada, e realiza uma poda após a verificação do *Flood-fill*, criando uma área totalmente navegável e adotando um bom *Level Design*.

4.3. ANÁLISE DE CUSTO COMPUTACIONAL

Além dos resultados relacionados à conectividade, o experimento também mediu o tempo de processamento da CPU para cada cenário. É possível notar um crescimento linear do tempo com base no aumento do número de iterações. Para o cenário de parametrização escolhido (40% de preenchimento e 7 iterações), o tempo médio foi de aproximadamente 486 milissegundos (ms) para executar todo o ciclo de geração, o teste de *Flood-fill* e a etapa de poda das áreas inacessíveis. Nota-se que no contexto de jogos, um congelamento de meio segundo poderia causar quedra de *framerate*, jogos do gênero *Roguelike* possuem o padrão de gerar seus níveis previamente. Segundo Gregory (2018, p. 508, tradução nossa), uma abordagem comum e eficiente na arquitetura de motores gráficos é realizar o processamento de recursos em massa logo antes do início do gameplay, retendo o jogador em uma tela de carregamento ou uma barra de progresso. Dessa forma, o tempo de geração não gera um impacto

negativo na experiência do jogador.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve o objetivo de analisar e otimizar a geração procedural de níveis com foco em ambientes subterrâneos utilizando autômatos celulares viabilizando a conectividade da área navegável e a usabilidade em *Level Design* no motor gráfico Godot Engine.

O experimento demonstrou que utlizar a regra da Vizinhança de Moore de forma mais restrita evitou a geração de um efeito esponja, criando na verdade um sistema semelhante ao de cavernas orgânicas. Além disso, limitar o algoritmo de *Flood-fill* à Vizinhança de Von Neumann permitiu evitar a criação de áreas acessíveis matematicamente, mas intransitáveis a um jogador.

O ponto-chave deste trabalho está no encontro satisfatório entre a eficiência da geração e a estética desejada para o contexto de jogos. Notou-se que os cenários que permitiram uma conectividade quase total se mostravam um fracasso quando analisando a complexidade de obstáculos e de um ambiente jogável. Conclui-se, então, que o modelo ideal é alcançado sacrificando intencionalmente parte dessa conectividade em prol de um ambiente propício para um *Level Design*. Conforme visto anteriormente, as áreas inacessíveis podem ser corrigidas utilizando a técnica de poda, criando mapas coesos, dinâmicos e 100% navegáveis.

Por fim, o custo computacional se mostrou dentro do aceitável, podendo ser facilmente aplicado a jogos do gênero *Roguelike* em telas de carregamento eliminando a percepção de congelamento durante a geração.

Como proposta para trabalhos futuros, sugere-se a expansão deste modelo bidimensional para ambientes em três dimensões utilizando a estrutura de *Voxels*.

REFERÊNCIAS

AKENINE-MÖLLER, T. *et al.* **Real-Time Rendering**. 4. ed. Boca Raton: CRC Press, 2018.

DIAS, M. V. P. *et al.* Procedural map generation by energy propagation. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Anais do XXIV Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)**. Salvador: SBC, 2025.

Godot Engine. **Documentação do Godot Engine**. 2026. Disponível em: <https://docs.godotengine.org/pt-br/4.x/>. Acesso em: 28 abr. 2026.

GREGORY, J. **Game Engine Architecture**. 3. ed. Boca Raton: CRC Press, 2018.

HANNULA, R. **Balancing Randomness in Action Roguelike Game Design**. Monografia (Bachelor's Thesis) — South-Eastern Finland University of Applied

Sciences, Finland, 2025. Disponível em: <https://www.theseus.fi/>. Acesso em: 14 maio 2026.

HEARN, D.; BAKER, M. P. **Computer Graphics with OpenGL**. 3. ed. Upper Saddle River: Pearson Prentice Hall, 2004.

JOHNSON, L.; YANNAKAKIS, G. N.; TOGELIUS, J. Cellular automata for real-time generation of infinite cave levels. In: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES. **Proceedings of the 2010 Workshop on Procedural Content Generation in Games (PCG '10)**. New York: ACM, 2010.

KARI, J. Theory of cellular automata: A survey. **Theoretical Computer Science**, v. 334, n. 1-3, p. 3–33, 2005.

PAES, J.; JUNIO, R.; RODRIGUEZ, L. C. Machine learning for playable room generation: A modular approach with vae and pcg. In: **Anais do 24º Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)**. Porto Alegre: Sociedade Brasileira de Computação, 2025. p. 609–618.

ROCHA, J. B.; PRADA, R. Procedural content generation for cooperative games—a systematic review. **IEEE Transactions on Games**, v. 17, n. 3, p. 545–557, 2025.

SALINAS, H. **An Exploration of Procedural Methods in Game level design**. Monografia (Honors Thesis) — University of Arkansas, Fayetteville, 2024.

SHUAEB, S. M. A. A.; KAMRUZZAMAN, M.; ALI, M. H. Extracting a bounded region from a map using flood fill algorithm. **Asian Journal of Research in Computer Science**, v. 7, n. 1, p. 14–20, 2021.

SO, A. R. P. *et al.* Automating dungeon level design using search-based procedural generation. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Anais do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames)**. Manaus: SBC, 2024.

TOTTEN, C. W. **An Architectural Approach to Level Design: Processes and Experiences**. 2. ed. Boca Raton: CRC Press, 2019.

WEISSTEIN, E. W. **Cellular Automaton**. 2024. MathWorld–A Wolfram Web Resource. Disponível em: <https://mathworld.wolfram.com/CellularAutomaton.html>. Acesso em: 17 abr. 2026.

YANNAKAKIS, G. N.; TOGELIUS, J. **Artificial Intelligence and Games**. 2. ed. Cham: Springer, 2025.

ZHANG, Y.; ZHANG, G.; HUANG, X. A survey of procedural content generation for games. In: IEEE. **Proceedings of the 2022 International Conference on Culture-Oriented Science and Technology (CoST)**. [S.l.]: IEEE, 2022.

AGRADECIMENTOS

Em primeiro lugar eu agradeço a Deus por ter chegado até aqui, mesmo com todos os problemas e desafios que apareceram durante essa jornada, Ele sempre me protegeu e me guiou, a Ele ofereço este trabalho e tudo que aconteceu para que ele fosse concluído.

Agradeço também à Bem-Aventurada Virgem Maria, minha mãe celeste, a quem sob o título de Virgem do Rosário, sempre esteve presente na minha jornada.

Acima de todas as pessoas eu agradeço à minha mãe, Anasilda Pereira, foi ela que desde sempre foi minha guia, minha heroína, que tanto sacrificou para que eu chegasse onde estou, sempre presente, me incentivando, fazendo o papel de pai e mãe ao mesmo tempo, a quem devo tudo o que tenho e o que conquistarei. Obrigado por ser quem é e por tudo que fez, por tudo. Sempre te amarei. Como ensinado por Santa Gianna Beretta Molla: “Vejam as mães que realmente amam seus filhos: quantos sacrifícios elas fazem por eles. Elas estão dispostas a tudo, até mesmo a doar o próprio sangue para que seus bebês cresçam fortes e saudáveis”.

Agradeço ao meu padrinho Jozifran Campos, que me acolheu e me possibilitou continuar meus estudos, sempre serei grato ao seu auxílio.

Sobre meus amigos, se fosse nomear todos não teria espaço, cito meu amigo Ítalo José, amigo da minha infância, sempre disposto a me escutar e me dar conselhos, te considero um irmão mais velho que nunca tive. Agradeço também ao meu amigo Paulo Prudente, que não me conhecia mas se dispôs a me ajudar com esse trabalho, me dando dicas e sugetões. Ao mesmo tempo agradeço a Thiago Sales, que durante esses anos de amizade tanto me ajudou, inclusive neste trabalho, me auxiliando em correções e sugestões. A vocês meu muito obrigado.

Para meus amigos do IFPI primeiro gostaria de agradecer ao Gabriel de Almeida, em uma aula normal fizemos dupla e a partir de então fomos a dupla oficial de todos os trabalhos que pudemos, agradeço a você por me ajudar a chegar até aqui. Também agradeço ao Leonardo Vitório, minha segunda dupla que também apareceu de repente e se mostrou uma ótima amizade, através deles agradeço a todos os meus amigos que o IFPI me proporcionou.

Agradeço ao meu orientador Profº Francisco Marcelino, que mesmo que nem tenhamos nos visto pessoalmente, aceitou me orientar, ele é uma pessoa a quem eu gostaria de ter tido mais contato, mas que fico feliz de ter encontrado. Agradeço também à minha coordenadora, Profª Aline Leal, que já me ajudou de diversas formas, e tantas vezes, obrigado pelas oportunidades.

Para concluir, agradeço a todos os professores e servidores do IFPI que de alguma forma me trouxeram até aqui, e ao próprio IFPI pela oportunidade.