Criação de Mapas de Cidades para RPGs de Mesa Utilizando Algoritmos de Geração Procedural

João Henrique Lopes, Saulo Popov Zambiasi

Universidade do Sul de Santa Catarina (UNISUL)
Tubarão – SC – Brasil

h.joaolopes@gmail.com, saulopz@gmail.com

Abstract. *Locations are an important element of any tabletop RPG, and one of the most common locations are cities. A difficult task that RPG game masters face is obtaining maps for these urban localities. Procedural generation algorithms could help solve this problem by generating random numbers combined with constraint solving rules to construct a large amount of content with less effort. This article's objective is to show how procedural generation algorithms could be used in the process of medieval city map creation for usage in tabletop RPGs.*
Keywords: *Procedural Generation, City Modeling, Tabletop RPG, Voronoi Diagrams, Recursive Subdivision*

Resumo. *Cenários são um elemento importante para qualquer RPG de mesa, e, entre estes, um dos mais comuns é o cenário das cidades. Uma dificuldade que narradores de RPG de mesa podem ter é obter mapas para esses cenários de cidades. Algoritmos de geração procedural poderiam ajudar a resolver esse tipo de problema gerando números aleatórios combinados com regras de restrição para construir grandes quantidades de conteúdo com menor esforço. Nesse contexto, este trabalho propõe apresentar e demonstrar algoritmos de geração procedural voltados à criação de mapas de cidades medievais para o uso em jogos de RPG de mesa.*
Palavras-chave: *Geração Procedural, Modelagem de Cidades, RPG de Mesa, Diagramas de Voronoi, Subdivisão Recursiva*

1. Introdução

Tabletop role-playing games, ou RPGs de mesa, são jogos de interpretação de papéis em que os jogadores interpretam personagens em campanhas, seguindo uma série de regras ou sistemas previamente definidos, a fim de desenvolver uma história de forma colaborativa. Neste contexto, existem dois tipos de participantes: jogadores, que representam seus personagens; e o **mestre**, ou **narrador**. O narrador é responsável por criar e elaborar a narrativa, o que inclui descrever os cenários, desafios e demais situações em que os outros jogadores se encontram. [Ruas Machado Gomes et al. 2021].

O cenário é um elemento de extrema importância para um RPG de mesa, pois é lá que os personagens dos jogadores estão espacialmente situados em suas aventuras. Além disso, é usual a utilização de mapas físicos, sejam impressos ou em forma de maquete, e, em certos casos, miniaturas dos personagens dos jogadores são colocadas sobre os mapas [Tan 2021]. Frequentemente o narrador pode usar cenários de cidades medievais para suas

histórias, e os mapas destas contêm certos elementos comuns: ruas, pontos de interesse e formas geométricas que representam prédios, casas ou construções.

Para a obtenção de mapas de cidade, um recurso que o narrador pode utilizar são os suplementos de campanha: livros que disponibilizam mapas, descrições e conteúdos para o cenário. Um grande exemplo é o livro "City of Splendors: Waterdeep" que descreve o passado e fornece detalhes da cidade medieval de Waterdeep, situada no universo fictício do RPG de mesa *Dungeons & Dragons* [Boyd 2005]. No entanto, esses recursos nem sempre contêm os mapas que o narrador deseja para seu cenário.

Ao invés de utilizar livros de suplemento, o narrador também pode optar por fazer seus próprios mapas de forma manual, desenhando em papel com canetas e materiais artísticos [Blando 2019]. Entretanto, caso o narrador não tenha habilidade criativa ou artística, esse trabalho pode se tornar moroso ou não trazer resultados que agradem.

E, por fim, uma outra forma de resolver esse problema — aspecto central deste artigo — é através de algoritmos de geração procedural que são capazes de criar vários mapas verossímeis de forma fácil, até que o narrador encontre um que o agrade. As vantagens de utilizar algoritmos computacionais para produção de tais mapas são diversas: facilidade de gerar um grande número de opções, customização e rapidez na descoberta de diferentes modelos [Short and Adams 2017].

Neste sentido, o presente trabalho pretende apresentar diferentes algoritmos existentes de geração procedural de conteúdo e, a partir disso, escolher um destes para o desenvolvimento de um programa que o utilize para a criação de mapas de cidades medievais, com ruas, estruturas e pontos de interesse. Para esse objetivo, as cidades geradas terão o formato similar ao da figura 1, ou seja, cidades de pequena escala, sem planejamento, com formatos orgânicos e divisões e separações distintas entre blocos de estruturas, casas e prédios.

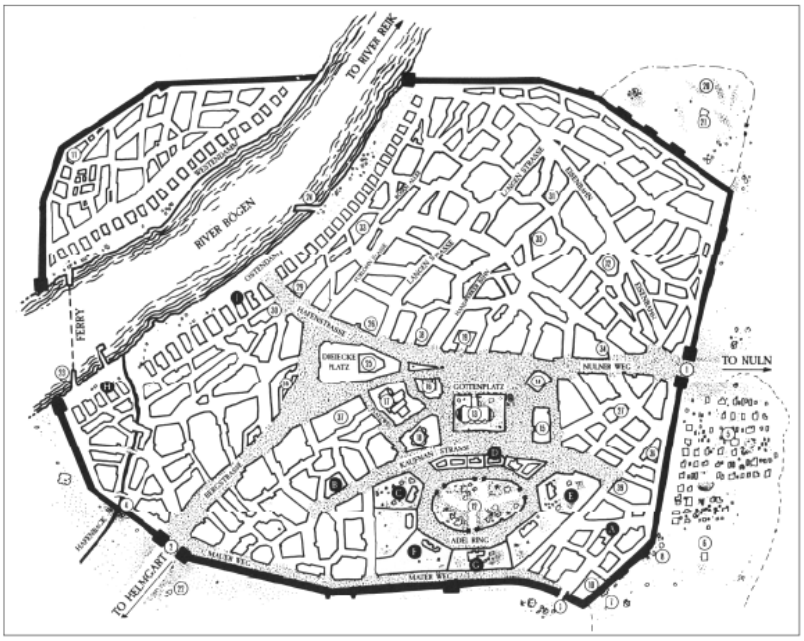

Figura 1. Um mapa da cidade de Bögenhafen, do universo fictício de *Warhammer Fantasy Role-playing* [Gallagher and etc. 1995].

Esse artigo se organiza da seguinte forma: ao longo da Introdução foi realizada uma apresentação do que são RPGs de mesa, da importância dos cenários, das dificuldades enfrentadas pelos narradores, e das diferentes maneiras de se obter mapas de cidades para campanhas. Na Revisão Bibliográfica são apresentadas diferentes técnicas de geração procedural que podem ser usadas na criação de mapas que contêm ruas, formas geométricas e pontos de interesse. No Desenvolvimento, uma das técnicas é escolhida e o projeto é desenvolvido utilizando um motor de jogos para representação gráfica. No capítulo de Avaliação, é feita uma pesquisa qualitativa baseada nos resultados do programa e as repostas são analisadas.

2. Revisão Bibliográfica

Algoritmos de geração procedural são algoritmos utilizados para criação de conteúdo que tipicamente seria feito por humanos. Estes fazem uso frequente de números aleatórios, e têm como objetivo substituir ou replicar a criatividade e design de profissionais humanos [Smith 2015].

No artigo *"A survey of procedural techniques for city generation"* de Kelly e McCabe são apresentadas técnicas de algoritmos que podem ser utilizadas para modelagem de cidades. Destas, L-Sistemas e Diagramas de Voronoi são exploradas neste trabalho [Kelly and McCabe 2006]. Além disso a técnica "Wave Function Collapse" também será abordada [Gumin 2016].

2.1. Geração das ruas, estradas e formatos dos blocos da cidade

A primeira técnica que pode ser escolhida para a geração de mapas de cidades para o fim deste artigo, é chamada de **L-Sistema** ou **Sistema de Lindenmayer**, descrito por Aristid Lindenmayer em seu artigo *Mathematical Models for Cellular Interactions in Development* de 1968 [Lindenmayer 1968]. Um Sistema de Lindenmayer é um sistema de reescrita paralela baseado em regras de produção. Ou seja, é um conjunto de símbolos e regras, e essas regras são aplicadas iterativamente obtendo um conjunto maior de símbolos, que então podem ser convertidos em representações geométricas [Baader and Nipkow 1998].

Na figura 2 é visto um L-Sistema que cria uma visualização de árvore, e para explicar o processo que levou a isto, existem algumas definições formais:

- Alfabeto (V): O conjunto de símbolos
- Axioma (ω): O estado inicial
- Regras: (P): Definições de como substituir os símbolos em cada iteração

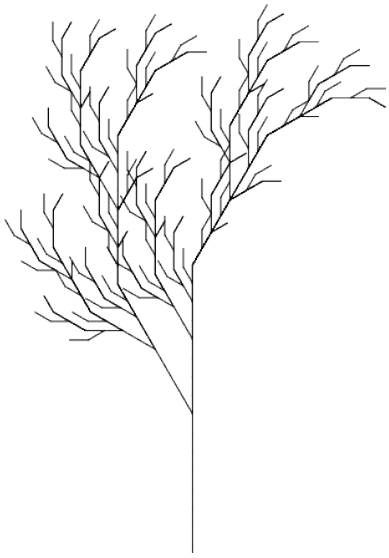

Figura 2. Uma visualização de L-Sistema de árvore utilizando um visualizador online [Chong 2016].

No exemplo da figura 2, o alfabeto utilizado foi X e F e o axioma foi X. Além disso, foram escolhidas duas regras: A primeira, $F = FF$, comunica que cada símbolo F deve ser substituído por dois símbolos F.

A segunda regra, $X = F-[[X+X]+X+X]+F[+F]$, explica que cada símbolo X deve ser substituído por $F - [[X + X] + X + X] + F[+F]$.

Neste exemplo, em cada lugar que o símbolo F aparece pode-se interpretá-lo visualmente como uma linha reta que sobe, enquanto os sinais de $-$ e $+$ representam angulações de 30º graus para esquerda e para direita nesta linha, respectivamente.

Os colchetes criam ramificações, isto é, dividem a árvore em dois caminhos distintos, que podem ser observados na imagem da figura 2.

Por causa de sua gramática simples, utilizar L-Sistemas da forma original para contemplar todas as regras que um mapa de cidade descreve faz com que o nível de complexidade aumente drasticamente, principalmente no que diz respeito às gerações das ruas. Para resolver isso, é possível uma extensão do L-Sistema, através da introdução de um conceito de *sucessor ideal*, explorado a fundo por Yoav Parish e Pascal Müller em seu artigo *Procedural Modeling of Cities* [Parish and Müller 2006].

Mesmo com essas otimizações, a abordagem de Sistemas de Lindenmayer pode ser desnecessariamente complexa para o fim de criação de mapas de cidades medievais, então uma das possíveis alternativas a este método é a simplificação do processo para utilizar filas de prioridade ao invés de L-Sistemas [Barrett 2009].

Outra técnica que pode ser utilizada para geração procedural de mapas é chamada de **Wave Function Collapse**, um algoritmo de resolução de restrições que gera seu resultado baseado em exemplos [Gumin 2016]. O nome do algoritmo remete às regras de Colapso de Ondas do campo da mecânica quântica, porém o algoritmo em si funciona em uma matriz NxN, na qual N denota os números de altura e largura de blocos que

constituem a matriz, que, por sua vez, representa uma imagem. O primeiro passo é ler a matriz inicial e extrair todos os possíveis estados que uma célula pode ter, e também quais as regras para conexões entre células.

Na figura 3, temos uma imagem formada por blocos que poderiam representar estradas, a única regra existente é que a célula de estrada precisa estar conectada com outra célula de estrada, e por isso, não pode existir uma conexão vazia com uma conexão de início de estrada.

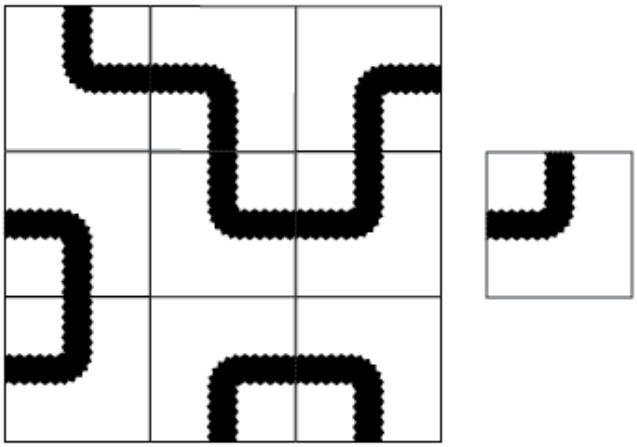

Figura 3. Uma matriz 3x3, que contém 4 possíveis estados, todas os estados são simetrias da imagem separada à direita (Autor).

Após a etapa inicial, o algoritmo prepara a matriz de saída, inicializando-a inteiramente num estado não-observado, ou seja, todas as células podem conter qualquer estado. Após isso, são encontradas as células com o menor valor de entropia, cujo valor em questão também não seja zero. Isto significa encontrar a célula com o menor número de possíveis estados, e, a partir disso, colapsa-la escolhendo um dos estados de forma aleatória e definindo seu estado até o final do algoritmo. Caso existam múltiplas células com o mesmo número de entropia, uma aleatória é escolhida. Além disso, após uma célula ser colapsada, as demais ao seu redor têm seu número de entropia diminuído, pois agora precisam ser restritas a somente os estados que se encaixam com a célula recém colapsada. Este processo se repete até o que todas as células sejam preenchidas [Gumin 2016].

Na figura 4, uma vez que a célula central é colapsada, o número de estados possíveis das células adjacentes diminui, pois, agora, aquelas vizinhas da central precisam permitir a conexão das estradas. Os estados possíveis que não se encaixam são eliminadas. Em relação às não adjacentes, ainda existe o número máximo de entropia (4 estados).

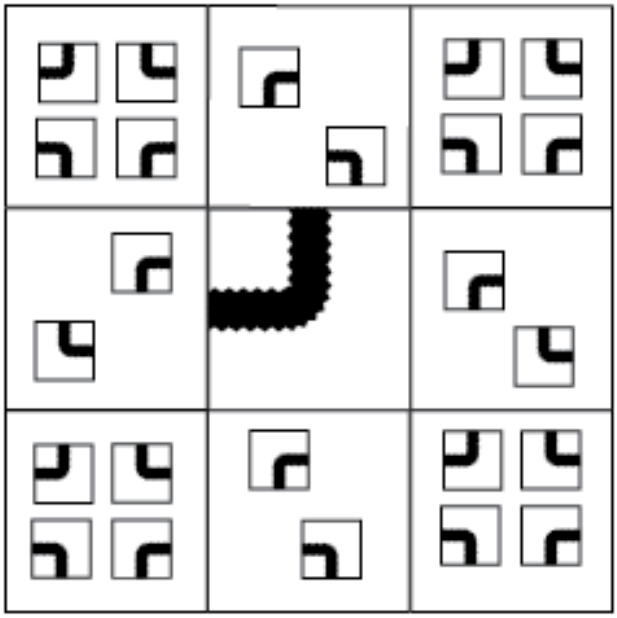

Figura 4. A matriz 3x3 de saída, com a célula central colapsada e os possíveis estados das outras células adjacentes apresentados (Autor).

O algoritmo original gera resultados homogêneos, pois ele não pode ser especificamente otimizado para a geração de mapas de cidades. No entanto, atráves do uso de restrições iniciais, isso pode ser mitigado: colocam-se pontos de interesses ou estruturas, e, então, utiliza-se o algoritmo de Wave Function Collapse para a geração da cidade em volta destes [Gaisbauer et al. 2019].

Uma outra forma de abordar o assunto de geração de mapas de cidade é através de **Diagramas de Voronoi**, uma forma de divisão de um plano que utiliza a distância de pontos distintos para constituir o que são chamadas de células [Aurenhammer 1991]. Uma forma para a criação de Diagramas de Voronoi se dá através da escolha de pontos aleatórios em um plano. Estes são os núcleos das células e, portanto, sua quantidade irá definir o número total de células. Feito isso, é aplicada uma regra para todos os outros pontos: um ponto pertence à região da célula do núcleo mais próximo. Para melhor visualização, todos os pontos pertencentes a uma mesma célula podem ser pintados com a mesma cor, conforme ilustra a figura 5.

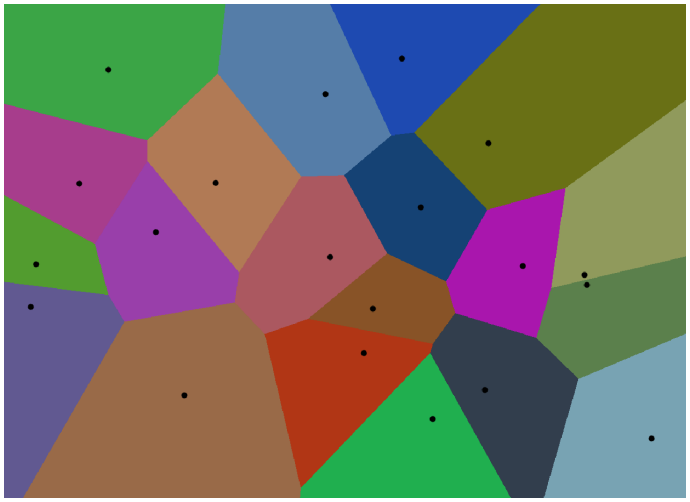

Figura 5. Um Diagrama de Voronoi com 20 células (Autor).

Os parâmetros do Diagrama de Voronoi podem ser manipulados para gerar resultados mais personalizados, como por exemplo variar as distâncias e os posicionamentos dos núcleos, além das das cores, que podem ser trocadas por imagens [Worley 1996]. Partindo deste princípio, os resultados dos diagramas podem ser adaptados também para a construção de mapas de cidades. A título de exemplo, células podem representar bairros e seus limites, ruas ou estradas, como vistos na figura 6 [Dolya 2017].

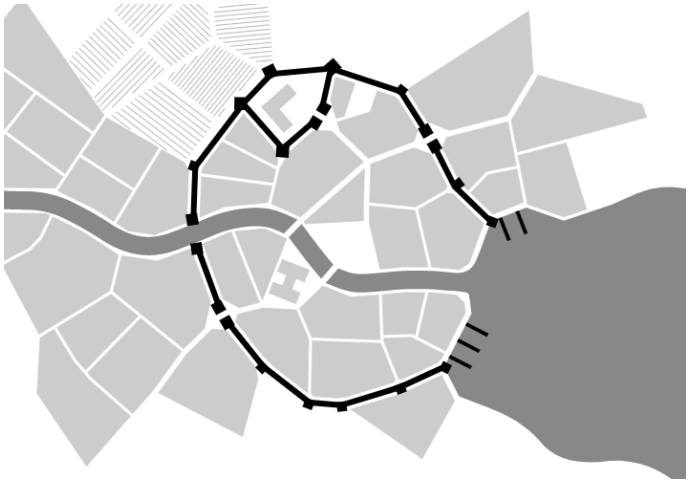

Figura 6. Um mapa de cidade gerado usado a ferramenta online *Medieval Fantasy City Generator* [Dolya 2017].

2.2. Geração de estruturas dentro dos blocos da cidade

Uma vez que o formato das ruas é definido, as formas dos blocos que compõem a cidade também são formadas. Agora, é necessário preencher esse espaço com estruturas, casas ou prédios com o objetivo de chegar a um resultado crível, representante uma cidade medieval.

Esses processos são chamados de *Block Subdivision Algorithms* — ou Algoritmos de Subdivisão de Blocos — e representam as maneiras em que o espaço de-

finido por um bloco de qualquer formato pode ser dividido em pedaços menores [Wiseman and Patterson 2016]. Existem diferentes modos de alcançar esse objetivo [Vanegas et al. 2012].

Um jeito possível de realizar uma subdivisão é através da utilização de um algoritmo de *Straight Skeleton Subdividing*, ou **Subdivisão de Esqueleto Reto** [Huber 2018]. Tal algoritmo consegue identificar as linhas centrais de um bloco, e, a partir das faces geradas pela divisão central do esqueleto, arestas que são adjacentes e de curvatura semelhante são agrupadas. Os grupos de faces são cortados perpendicularmente às bordas do bloco, de modo a criar pequenos lotes. Essas etapas podem ser observadas nas etapas da figura 7.

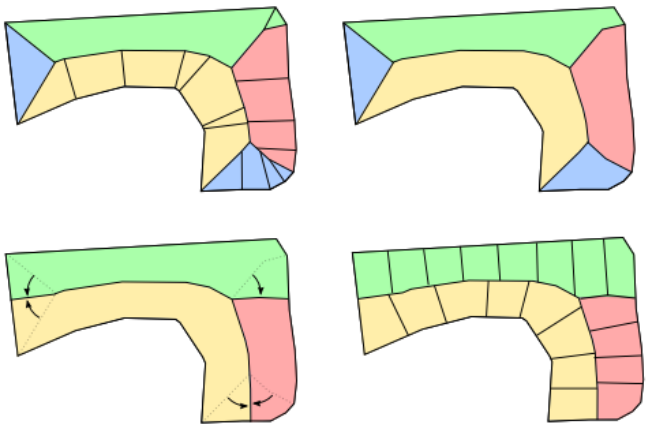

Figura 7. Divisão de um bloco com o uso da Subdivisão de Esqueleto Reto [Esri RD Center Zurich 2022].

Outra forma de dividir essa área, que produz resultados menos homogêneos, é chamada de **subdivisão recursiva**. Baseado no conceito de particionamento binário de espaço, esse processo, a partir de uma área inicial, divide esta em dois, bem como também as áreas subsequentes, de forma recursiva [Fan et al. 2018]. Esses passos podem ser vistos na figura 8, onde é utilizada uma profundidade de 4 iterações para as subdivisões.

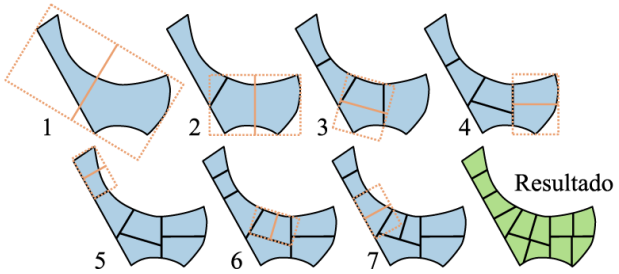

Figura 8. Divisão de um bloco utilizando subdivisão recursiva [Vanegas et al. 2012].

3. Desenvolvimento

Para a construção e visualização dos mapas de cidade criados de acordo com as técnicas mencionadas anteriormente, diversas ferramentas estão disponíveis. Neste trabalho, a ferramenta escolhida é o motor de jogos *Unity*.

Normalmente a ferramenta Unity é utilizada para a construção de jogos, que, por sua vez, contêm interação do usuário. No entanto, o Unity é, neste projeto, usado como forma de representação gráfica do processo de geração de cidades implementado, uma vez que contém recursos de renderização gráfica que são úteis para as etapas que serão aplicadas, como, por exemplo, a renderização e manipulação de polígonos [Halpern 2019].

O Unity por sua vez, utiliza como linguagem de programação o C#, e será nela que os algoritmos e as técnicas serão implementadas [Ferrone 2021].

3.1. Técnica Escolhida

Considerando todas as técnicas abordadas no capítulo de revisão bibliográfica, e pensando em facilidade de implementação, assim como flexibilidade para expansão e ajustes, a técnica escolhida para o desenvolvimento será a de **Diagramas de Voronoi**.

3.2. Criação dos Blocos da Cidade

Para a criação do Diagrama de Voronoi, diversas abordagens podem ser utilizadas, a mais simples é a criação de uma matriz NxN, e, a partir desta, gerar aleatoriamente as coordenadas dos núcleos desejados, considerando que, para um único núcleo, sua posição na coluna da matriz será um número gerado de 0 até N, e, para posição na fileira da matriz, também outro número é com a mesma regra. Esta etapa pode ser repetida até ser criada a quantidade de núcleos desejados.

A partir disso, para cada posição dessa matriz, poderá ser calculado a qual núcleo esta posição pertence: isso é obtido pelo resultado do cálculo da distância euclidiana entre essa a coordenada desta posição e as coordenadas de todos os núcleos do diagrama. O núcleo cuja distância resultar no menor valor, será o núcleo em que a coordenada da posição em questão pertencerá, e sua cor pode ser pintada para representar isso.

Em uma definição formal, para n células $\{p_1, ..., p_n\}$, todos os pontos da célula C_i são determinados pela seguinte fórmula [Roelandts 2020]:

$$C_i = \{x \in R^2 \mid d(x, p_i) \leq d(x, p_j)\ \forall i \neq j\}$$

Essa técnica irá criar um diagrama exatamente como o da figura 5. No entanto, é possível perceber que este método pode gerar tamanhos e formatos muito discrepantes. Para diminuir essa diferença de tamanhos, mas ainda manter a ideia de formatos variados, é possível primeiro dividir a matriz inicial em blocos de, por exemplo tamanho $\frac{N}{5}$, de tal forma que seja gerado somente um único núcleo em cada bloco $\frac{N}{5}$x$\frac{N}{5}$, no total gerando, portanto 25 núcleos.

Na figura 9 é possível perceber a diferença. Desta vez, o espaço inicial foi dividido em 25 secções, e dentro de cada uma destas secções, foi escolhido um ponto aleatório para ser um núcleo. As próximas etapas são as mesmas do processo sem a divisão $\frac{N}{5}$x$\frac{N}{5}$. Essa pequena alteração na maneira de gerar o diagrama de Voronoi resulta em formas mais

semelhantes, uma vez que a coordenada de criação dos núcleos é limitada pelos limites de cada secção.

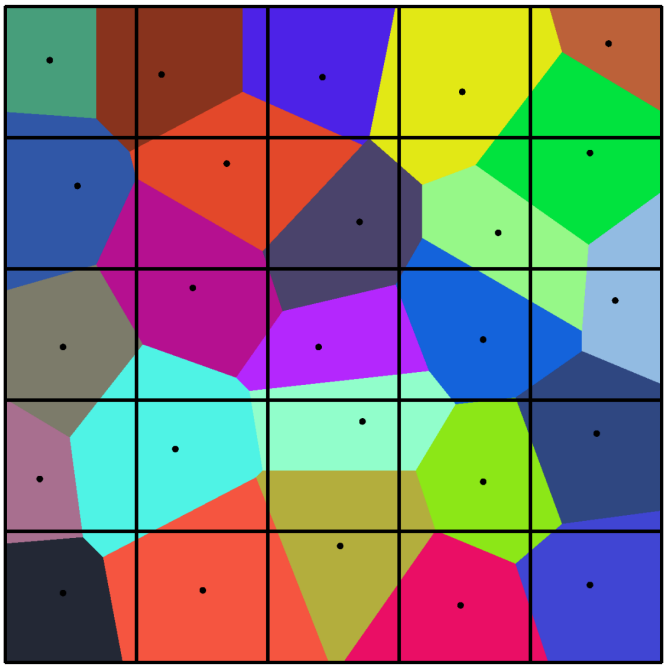

Figura 9. Diagrama de Voronoi com 25 células, utilizando divisão de blocos (Autor).

3.3. Criação das Ruas Entre os Blocos

Para a criação das ruas que ficam situadas entre os blocos da cidade, podemos definir que estas simplesmente serão as arestas de cada célula do diagrama de Voronoi. Uma forma fácil de criar essas ruas é encolher cada vértice que forma a célula em uma medida arbitrária. Desta forma, todas as células são encolhidas uniformemente, gerando um efeito de espaçamento entre elas. Esse processo pode ser visualizado na figura 10.

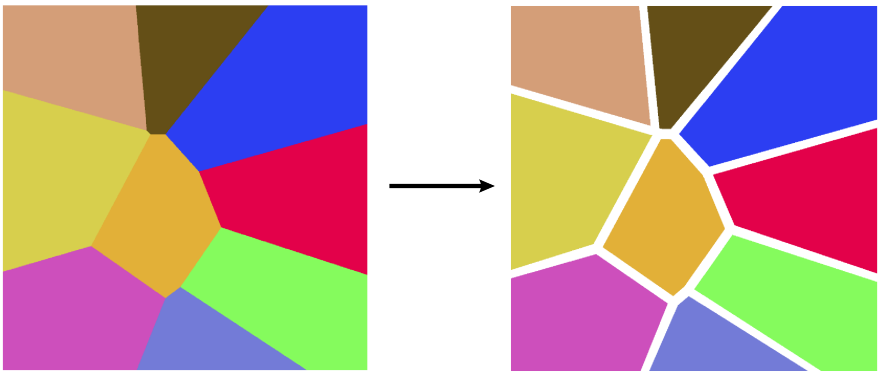

Figura 10. Um diagrama de Voronoi e o mesmo diagrama após cada célula ser encolhida (Autor).

Para um vértice C_i do polígono que compõe uma célula de Voronoi, podemos

calcular qual a nova posição do vértice (P) após o encolhimento pelo fator x a partir da seguinte fórmula:

$$V_1 = C_i - C_{i+1}$$
$$U_1 = \frac{V_1}{|V_1|}$$
$$V_2 = C_i - C_{i-1}$$
$$U_2 = \frac{V_2}{|V_2|}$$
$$D = (U_1 + U_2) \times x$$
$$P = C_i - D$$

3.4. Formato da Cidade

Uma característica dos diagramas apresentados até agora, é que todos têm o formato perfeitamente quadrado. Porém, é extremamente raro encontrar uma cidade perfeitamente quadrada.

Para introduzir um formato mais orgânico à cidade, é possível definir um raio limite a partir do centro do diagrama, considerando que qualquer núcleo que se encontre fora desse raio deve ser excluído do diagrama. Na figura 11 observa-se o processo em ação. O formato final é menos artificial que o inicial.

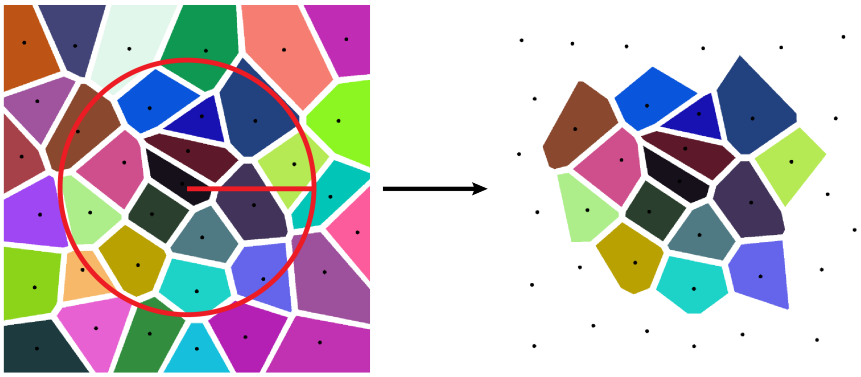

Figura 11. Um diagrama de Voronoi antes e depois de eliminarmos as células fora do círculo (Autor).

Dado uma célula i, ela é removida caso a distância entre o núcleo da célula e o centro do diagrama c for superior ao valor do raio r da circunferência, conforme a fórmula:

$$(i_x - c_x)^2 + (i_y - c_y)^2 \leq r^2$$

Nesta etapa, além de um círculo, outras formas geométricas podem ser utilizadas, como por exemplo elipses, retângulos ou uma combinação destes, dependendo do resultado final desejado.

3.5. Subdivisão dos Blocos de Cidade

A próxima etapa é subdividir cada bloco da cidade utilizando a técnica de subdivisão recursiva, apresentada na revisão bibliográfica. Nesta implementação, iremos calcular o centroide do formato em cada etapa, e a partir disso, encontrar a linha de menor tamanho que passe por este centroide, que será usada para a divisão.

Esse processo cria duas metades que compõem a forma original, e então a subdivisão pode ser aplicada novamente, de forma recursiva para alcançar o nível de divisões desejado, como pode-se observar na figura 12.

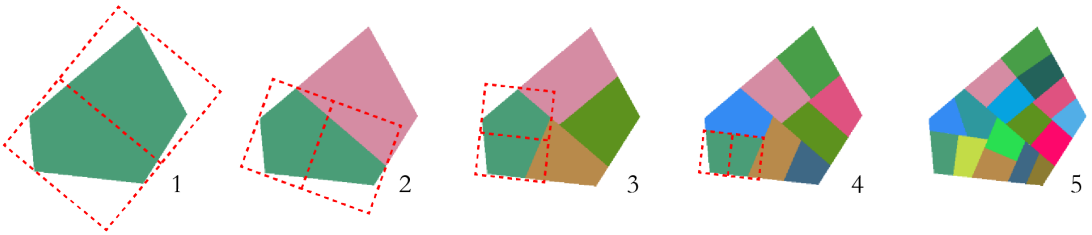

Figura 12. Um elemento de bloco da cidade, sendo subdividido 4 vezes (Autor).

O processo para criação dessa subdivisão de blocos é definido a seguir, em pseudocódigo, no algoritmo 1 [Vanegas et al. 2012].

Algorithm 1 Subdivisão OBB ("Oriented Bounding Box")

$subdivOBB(B)$
  $L \leftarrow \emptyset$
  $recSubdivOBB(C(B))$

$recSubdivOBB(l)$
if $area(l) \notin (A_{min}, A_{max})$ **and** $frontSideWidth(l) \notin (W_{min}, W_{max})$ **then**
  $s \leftarrow computeSplitLine(l)$
  $[l_A, l_B] \leftarrow split(B, s)$
  if l_A or l_B have no street access **then**
    Rotate s 90 degrees about the normal vector of the plane containing B, with probability r
    $[l_A, l_B] \leftarrow split(B, s)$
  end if
  $recSubdivOBB(l_A)$
  $recSubdivOBB(l_B)$
else
  Append l to L
end if

$computeLineSplit(l)$
  $OBB \leftarrow computeOBB(l)$
  Let the direction of l be the direction of the shortest side of OBB
  Let the pivot point of l be the middle point of OBB
  Apply a random translation of magnitude $\omega \times d_{OBB}$ to the pivot point of l, where d_{OBB} is the length of the shortest side of OBB

Ao aplicar essa técnica em todos os blocos da cidade, gera-se um resultado como o da figura 13.

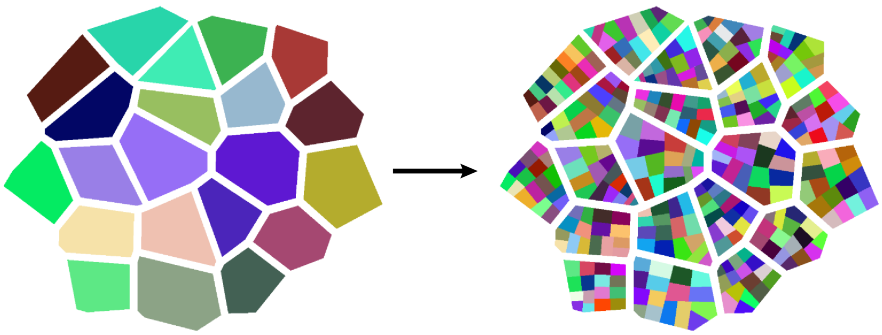

Figura 13. Um conjunto de blocos de uma cidade gerada, antes e depois de serem subdivididos 4 vezes (Autor).

Como é observado no algoritmo 1, é possível adicionar uma um fator probabilístico que condiciona a subdivisão. A diferença pode ser vista na figura 14.

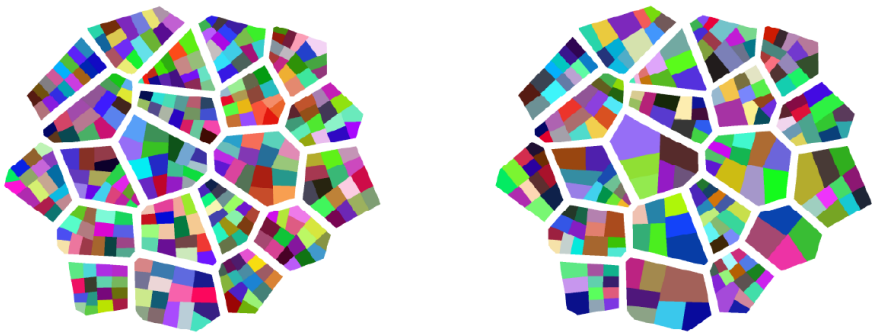

Figura 14. Uma cidade com probabilidade de 100% de subdivisão (esquerda), e a mesma cidade com uma probabilidade de 80% de subdivisão (direita) (Autor).

3.6. Desregularização dos Blocos Subdivididos

Uma característica que pode-se observar após a subdivisão, é que todos os blocos estão perfeitamente conectados aos blocos adjacentes. Para diminuir a regularidade destes, com o objetivo de tornar a cidade mais humana e menos perfeita, é possível aplicar de forma aleatória uma probabilidade para que uma das faces dos blocos seja reduzida em uma dada distância.

O processo pode ser observado na figura 15, na qual foi usada uma chance de 40% de que qualquer bloco ter um de seus lados reduzidos.

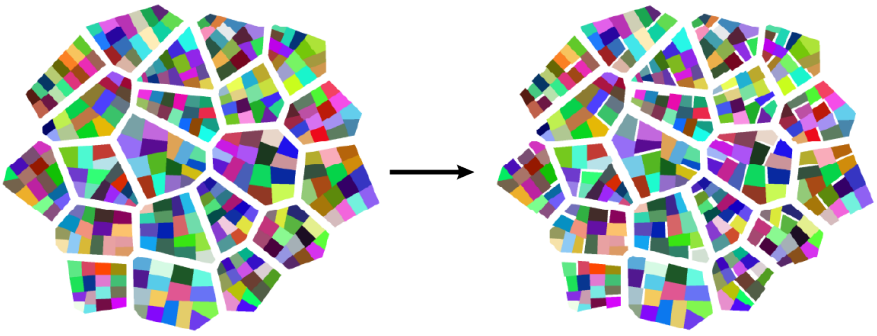

Figura 15. A cidade com os blocos subdivididos antes e depois de ser aplicado o processo de desregularização (Autor).

Para alcançar este efeito, um processo similar ao do capítulo 3.3 pode ser utilizado, com a diferença de que apenas os vértices que compõem uma face devem ser encolhidos, ao invés de todos.

3.7. Pontos de Interesse

O próximo passo a ser tomado, é a criação de pontos de interesse. Nesta etapa, existem diversas ideias e opções que podem ser empregadas. Pode-se criar cemitérios, templos, torres, ou até castelos. Neste trabalho, são criadas duas estruturas, para demonstrar como pontos de interesse podem ser introduzidos no mapa.

Uma praça de mercado foi criada, considerando a regra que ela deve substituir um bloco central da cidade. Além disso, também uma igreja foi criada, com a especificação que ela deve ser a união de certos sub-blocos de um bloco próximo ao do mercado. Na figura 16 os pontos de interesse criados podem ser observados. O mercado e a igreja são representados com a cor cinza e com seus respectivos ícones.

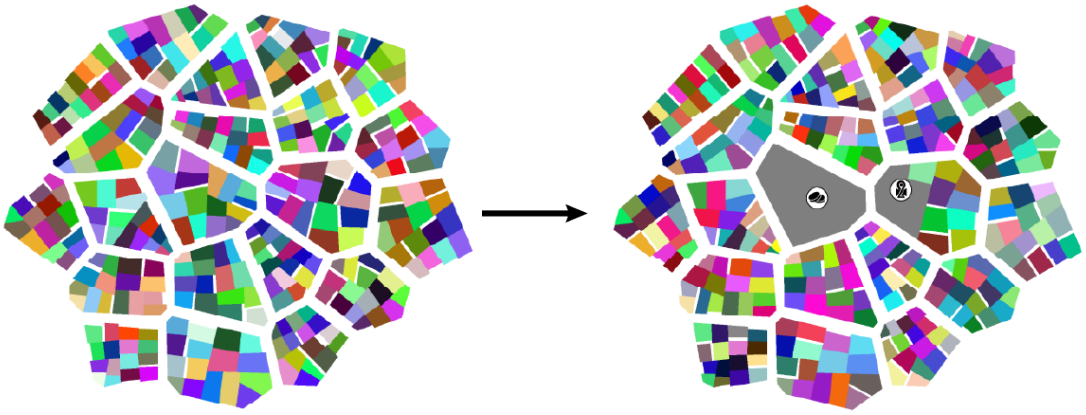

Figura 16. A cidade após dois pontos de interesse serem adicionados[1] (Autor).

3.8. Estilização Visual

Por fim, a cidade estando quase pronta, uma última etapa que deve ser feita é transformar a representação gráfica do mapa em algo mais similar ao do exemplo da introdução deste trabalho (figura 1).

Para isso, primeiro, podemos aplicar um contorno nos sub-blocos e remover as cores, como demonstra a figura 17.

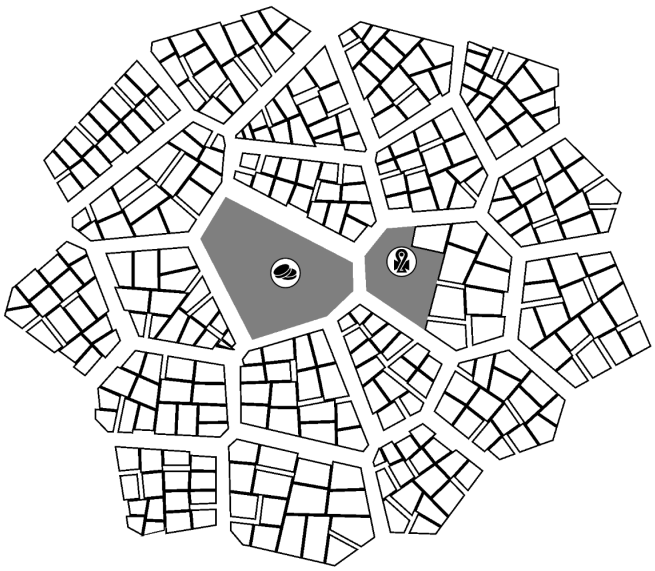

Figura 17. Cidade com a representação de linhas de contorno (Autor).

[1] Ícones do website https://game-icons.net/

Outra etapa visual a ser feita é a ondulação das linhas de contorno, com objetivo de tornar o mapa mais rústico. Para alcançar isso, para uma determinada linha que compõe o sub-bloco, são escolhidos 10 pontos uniformemente espaçados na linha, e cada ponto tem sua posição ligeiramente alterada de forma aleatória, na direção horizontal e vertical. A nova linha então é traçada ao longo destes pontos. Isso pode ser observado na figura 18.

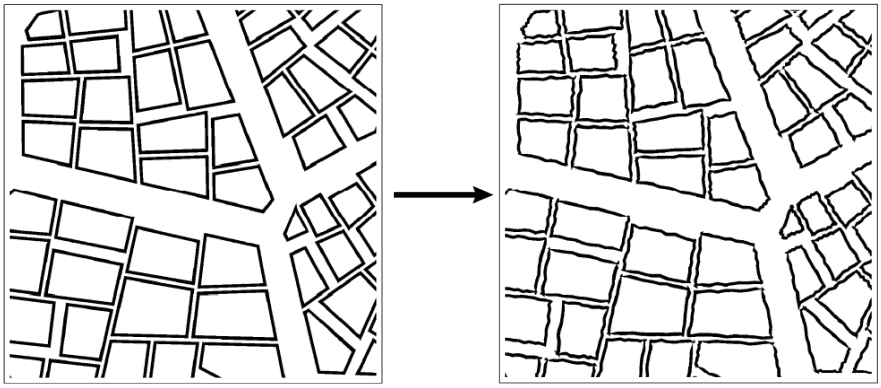

Figura 18. As linhas de contorno, antes e depois de aplicarmos a ondulação (Autor).

3.9. Parametrização

Todas as opções que apresentadas anteriormente podem ser alteradas para alcançar resultados diferentes. Na figura 19 são apresentadas 3 cidades configurações variadas.

A primeira cidade (1), com semente de randomização 4124, possui 36 células inciais, uma chance de subdivisão de 83%, e uma chance de desregularização de 41%. Já a segunda cidade (2), de semente 12412, é composta por 64 células iniciais, tem chance de subdivisão de 50%, e de conta com 80% de probabilidade de desregularização. Por fim, a terceira cidade (3), com semente 3477, composta por 49 células iniciais, tem a probabilidade de subdivisão de 95% e a chance de desregularização de 60%.

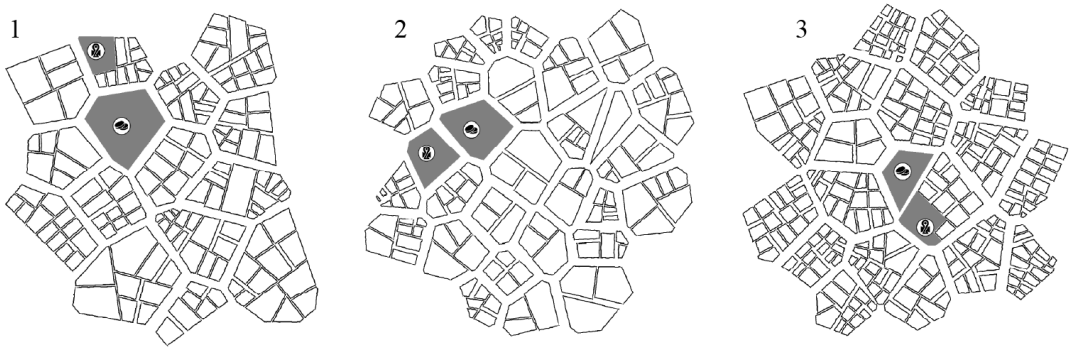

Figura 19. Três variações de cidades (Autor).

Além desses parâmetros, algumas outras opções que podem ser alteradas são: tamanho das ruas entre os blocos da cidade; número máximo de subdivisões; tamanho do raio para delimitar o formato da cidade; quantidade máxima e mínima para

desregularização; quantidade máxima e mínima de blocos que formam a igreja; grossura das linhas de contorno; e quantidade de ondulação nas linhas de contorno.

4. Avaliação

Para que seja possível chegar em uma conclusão sobre a qualidade do programa e dos algoritmos utilizados no desenvolvimento, foi realizada uma avaliação qualitativa, na qual os entrevistados responderam 5 questões. Neste caso, os 7 entrevistados são narradores de RPG de mesa.

Quatro das perguntas são respondidas através da escala de Likert, em que o entrevistado aponta o quanto concorda com a pergunta ou afirmação apresentada, podendo escolher entre as opções: "Concordo fortemente", "Concordo", "Indiferente", "Discordo", "Discordo fortemente", e "Não sei opinar" [Likert 1932]. A última pergunta é dissertativa, aberta para comentários, sugestões ou críticas. A seguir, cada pergunta é apresentada e uma breve análise é feita sobre os resultados.

> *"Na sua opinião, os mapas gerados pela ferramenta reproduzem bem a temática medieval de um RPG?"*

Nesta pergunta, 5 responderam com "Concordo", 1 "Concordo fortemente" e 1 "Indiferente"; figura 20 (1). De maneira geral, pode-se dizer que os entrevistados estão satisfeitos com a representação da temática medieval.

> *"Você utilizaria essa ferramenta para gerar mapas para as suas campanhas de RPG?"*

Na segunda pergunta, 4 dos entrevistados responderam com "Concordo", 2 "Concordo fortemente" e 1 "Indiferente"; figura 20 (2). Os mapas gerados, então, podem ser utilizados em campanhas, conforme o objetivo do trabalho.

> *"Você acredita que as opções de parametrização da ferramenta são suficientes para as suas necessidades?"*

Na terceira pergunta, respondeu-se 1 com "Discordo", 1 "Indiferente", 2 "Concordo", e 3 "Concordo fortemente"; figura 20 (3). Aqui já pode-se concluir que o programa poderia ter mais opções de customização.

> *"Na sua opinião, os mapas gerados pela ferramenta são esteticamente agradáveis?"*

Na pergunta referente a estética, teve-se 3 respostas com "Discordo", 2 "Concordo" e 2 "Concordo Fortemente"; figura 20 (4). Aqui, a opinião é dividida, porém, é possível perceber que certos entrevistados pensam que o aspecto visual do trabalho poderia melhorar.

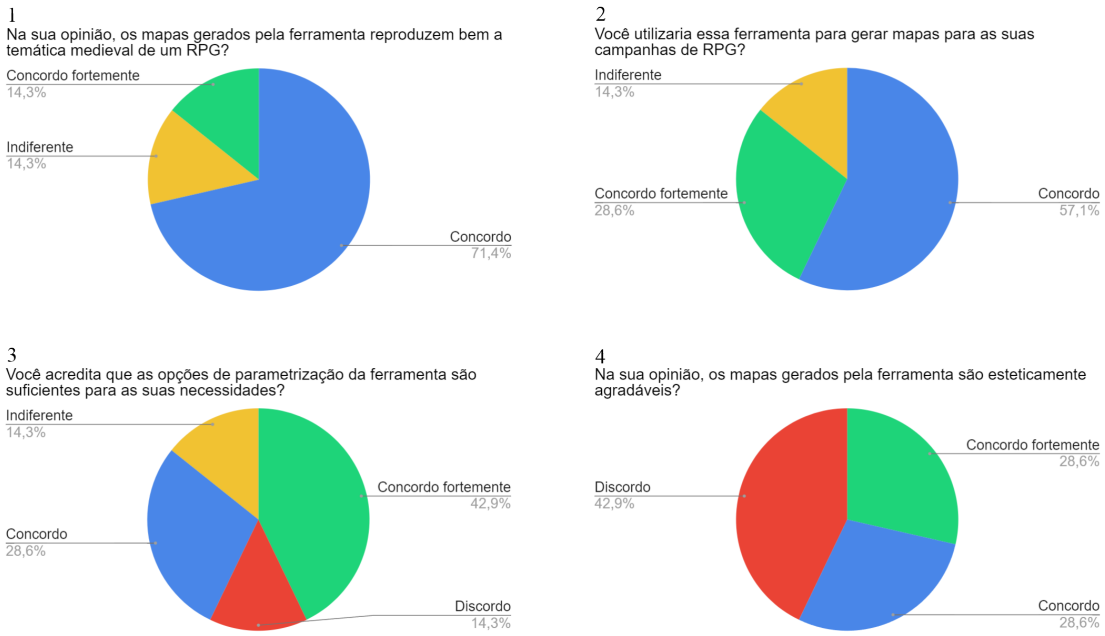

Figura 20. Resultados da pesquisa qualitativa (Autor).

> *"Por favor, utilize o espaço a seguir para fazer comentários, sugestões e críticas, em especial para os casos de respostas "discordo" às perguntas."*

As respostas para essa última pergunta dissertativa são variadas. Porém, essas podem ser divididas em três categorias: aspectos visuais; modelagem da cidade; e conteúdo para pontos de interesse.

Alguns entrevistados comentam que o aspecto visual do trabalho é simples demais, com algumas sugestões de que as linhas que formam as cidades poderiam ser menos quadradas e retas, além de ícones melhores para representar os pontos de interesse.

Outros pontos levantados foram em relação à modelagem da cidade. Entrevistados gostariam de opções como a criação de avenidas e ruas principais para a cidade. A introdução de rios ou corpos de água que cortariam a cidade também foi um assunto recorrente. Um último detalhe foi uma sugestão para que os blocos subdivididos sempre tenham conexão com as ruas da cidade, coisa que não está presente no trabalho atual.

Na questão de conteúdo adicional para pontos de interesse, as sugestões são: novas estruturas, como muralhas que englobam a cidade, e fazendas ao redor. Uma outra ideia levantada foi a adição de pequenos negócios ou comércios já nos blocos subdivididos, como lojas, tabernas ou estações de artesão.

5. Conclusão

Como elaborado anteriormente, cenários são elementos cruciais para RPGs de mesa. No entanto, esses recursos nem sempre atendem às necessidades do narrador, que pode optar por criar mapas manualmente, mas isso pode ser demorado e nem sempre satisfatório.

Foi proposta uma solução que utiliza algoritmos de geração procedural para criar mapas de cidades medievais com rapidez, menor esforço e alta personalização. Estes

algoritmos permitem a geração automática de ruas, estruturas e pontos de interesse, oferecendo uma alternativa eficiente e customizável para a criação de mapas medievais em RPGs de mesa.

Procurou-se analisar diversas técnicas de geração procedural, como L-Sistemas, geração por funções de onda e diagramas de Voronoi. No entanto, somente esta última foi escolhida, sendo combinada também com algoritmos de subdivisão recursiva, para a implementação do projeto, por motivos de simplicidade, e levando em consideração as necessidades definidas.

Foi criado um programa no motor de jogos Unity que alcança um nível alto de customização, possibilitando, assim, a geração de cidades de vários tamanhos e formas diferentes, com alguns pontos de interesse e com opções para expansão. Uma pesquisa qualitativa foi feita com narradores de RPG e os resultados foram positivos de forma geral, com diversas sugestões para melhorias.

Um próximo passo seria a implementação de algumas das sugestões dadas na pesquisa. Destas, considera-se mais importante a inclusão de rios e também a adição de novos e mais variados pontos de interesse e estruturas. Além disso, podem-se abordar melhorias em relação a estilização visual do projeto, como formas menos quadradas e retas, além de melhores representações para os pontos de interesse.

Referências

Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Comput. Surv., 23(3).

Baader, F. and Nipkow, T. (1998). Term Rewriting and All That. Cambridge University Press.

Barrett, S. (2007-2009). L-systems considered harmful: Acessado em 22/10/2022 http://nothings.org/gamedev/l_systems.html.

Blando, J. (2019). Fantasy Mapmaker: How to Draw RPG Cities for Gamers and Fans. IMPACT Books, Cincinnati, OH.

Boyd, E. (2005). City of Splendors: Waterdeep. Dungeons & Dragons: Forgotten Realms. Wizards of the Coast.

Chong, S. M. (2016). L-systems renderer: Acessado em 22/10/2022 http://piratefsh.github.io/p5js-art/public/lsystems/.

Dolya, O. (2017). Medieval Fantasy City Generator: Acessado em 04/11/2022 https://github.com/watabou/TownGeneratorOS.

Esri RD Center Zurich (2022). Description of algorithms, skeleton subdivision algorithm: Acessado em 26/11/2022 https://doc.arcgis.com/en/cityengine/latest/help/help-layers-block-parameters.htm.

Fan, X., Li, B., and Sisson, S. (2018). The binary space partitioning-tree process. In Storkey, A. and Perez-Cruz, F., editors, Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research. PMLR.

Ferrone, H. (2021). Learning C# by Developing Games with Unity 2021: Kickstart Your C# Programming and Unity Journey by Building 3D Games from Scratch, 6th Edition. Packt Publishing.

Gaisbauer, W., Raffe, W., Garcia, J., and Hlavacs, H. (2019). Procedural Generation of Video Game Cities for Specific Video Game Genres Using WaveFunctionCollapse (WFC).

Gallagher, P. and etc. (1995). Shadows Over Bögenhafen. The Enemy Within Campaign. Hogshead Publishing, London, England.

Gumin, M. (2016). Wave Function Collapse Algorithm: Acessado em 28/10/2022 https://github.com/mxgmn/WaveFunctionCollapse.

Halpern, J. (2019). Developing 2D Games with Unity. Apress Berkeley, CA.

Huber, S. (2018). The topology of skeletons and offsets. In Proc. 34th Europ. Workshop on Comp. Geom.(EuroCG'18).

Kelly, G. and McCabe, H. (2006). A survey of procedural techniques for city generation. The ITB Journal, 7:5.

Likert, R. (1932). A technique for the measurement of attitudes. Archives of psychology.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development i. filaments with one-sided inputs. Journal of Theoretical Biology.

Parish, Y. I. H. and Müller, P. (2006). Procedural modeling of cities. Proceedings of the 28th annual conference on Computer graphics and interactive techniques.

Roelandts, T. (2020). What is a voronoi diagram?: Acessado em 04/05/2023 https://tomroelandts.com/articles/what-is-a-voronoi-diagram.

Ruas Machado Gomes, R., Quevedo, M. A. d. A., and Pereira, G. P. (2021). Rola um d20, pedro bala! criação literária usando rpg de mesa. Letras & Letras.

Short, T. and Adams, T. (2017). Procedural Generation in Game Design. CRC Press.

Smith, G. (2015). An analog history of procedural content generation. In International Conference on Foundations of Digital Games.

Tan, A. M. (2021). TABLETOP RPGS: MINIATURES OR NOT?: Acessado em 22/10/2022 https://gnomestew.com/tabletop-rpgs-miniatures-or-no.

Vanegas, C., Kelly, T., Weber, B., Halatsch, J., Aliaga, D., and Müller, P. (2012). Procedural generation of parcels in urban modeling. Computer Graphics Forum.

Wiseman, N. and Patterson, Z. (2016). Testing block subdivision algorithms on block designs. Journal of Geographical Systems.

Worley, S. (1996). A cellular texture basis function. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, New York, NY, USA. Association for Computing Machinery.