Anais do Congresso de Matemática Aplicada e Computacional
CMAC Nordeste 2012
ISSN 2317-3297

Aplicação de Algoritmos de Grafos para Gerar e Percorrer Jogos de Labirintos Aleatórios

Rômulo C. Menezes Jr., Profa. Dra. Liliane S. Machado, Prof. Dr. Álvaro F. C. Medeiros
Programa de Pós Graduação em Informática, UFPB
58051-900, João Pessoa, PB
E-mail: romulojnr@gmail., liliane@di.ufpb.br, alvaro@di.ufpb.br

Palavras-chave: *Computação Gráfica, Algoritmos de Grafos, Busca em Largura, Busca em Profundidade, Desenvolvimento de Jogos*

Resumo: *Neste artigo é apresentado um arcabouço para o desenvolvimento de jogos de labirinto criados aleatoriamente e a mecânica de movimentação de personagens utilizando algoritmos baseados em busca em profundidade e busca em largura em grafos. Este arcabouço foi utilizado para representar um labirinto como um grafo, a partir de um algoritmo de busca em profundidade e um algoritmo de busca em largura para encontrar o menor percurso entre dois pontos. Assim, são descritos os conceitos necessários para a implementação das funções e a estrutura utilizada neste projeto. Por fim, é discutido como esta estrutura pode ser útil no protótipo de jogos de labirinto, ressaltando a importância do conhecimento dos algoritmos de grafo para o desenvolvimento de jogos.*

1 Introdução

Criar cenários para jogos com caminhos gerados aleatoriamente pode ser uma estratégia interessante para compor o ambiente do jogo. A tarefa de encontrar o caminho para a saída ou completar objetivos durante o percurso sempre em cenários distintos pode ser interessante para o jogador, eliminando a utilização repetitiva de mapas e cenários, o que pode causar desmotivação em relação ao jogo.

O jogo Minecraft e o jogo *The Maze EV* [3] são exemplos de jogos com cenários criados dinamicamente. O cenário do jogo Minecraft é formado em tempo real, assim que o jogador explora novas áreas o tamanho do cenário aumenta e novos elementos são criados dinamicamente[1]. No jogo *The Maze EV* são utilizadas telas de toque e ambientes virtuais para compor a interface de entrada para dois jogadores, enquanto um jogador projeta os caminhos do labirinto na tela, o outro jogador tentar escapar do labirinto gerado em tempo real .

Este trabalho utilizou grafos, estruturas de dados bem conhecidas e aplicadas em diversos problemas na computação para gerar cenários formados aleatoriamente para um jogo de labirinto. Através da modelagem do labirinto como um grafo, dois algoritmos foram utilizados para criar um labirinto gerado aleatoriamente e percorrê-lo. O algoritmo de busca em profundidade foi utilizado para gerar o cenário de um labirinto com caminhos aleatórios. O algoritmo de busca em largura foi utilizado para encontrar o menor percurso entre dois pontos [2].

Este artigo está estruturado da seguinte forma: Na Seção 2 são descritos a utilização de grafos e o funcionamento dos principais algoritmos de percurso; na Seção 3 as funções utilizadas no arcabouço para gerar o protótipo de um jogo de labirinto. Por fim, na Seção 4 são discutidas

[1]Geração de Terreno - Parte 1: http://notch.tumblr.com/post/3746989361/terrain-generation-part-1

ISSN 2317-3297

a importância e utilização dos grafos para as aplicações computacionais, especialmente os jogos digitais.

2 Algoritmos de Grafos

Grafos são estruturas utilizadas na computação que podem representar de forma abstrata diversos tipos de problema. Existem vários algoritmos para resolver problemas modelados a partir de estrutura como os grafos. Alguns problemas clássicos que podem ser solucionados a partir destes algoritmos são: problema do caminho mínimo entre dois nós, problema do caixeiro viajante e problema da árvore espalhada mínima, dentre outros. Para obter a solução para estes problemas é necessário conhecer a estrutura do grafo por meio de pesquisa sistemática das arestas de modo a alcançar todos os vértices. Isto pode ser feito através de algoritmos de busca em profundidade e busca em largura.

Foi utilizada uma modificação do algoritmo de busca em profundidade para criar um labirinto de caminhos aleatórios que gera como resultado uma árvore espalhada. Para realizar movimentação e solucionar o labirinto foi utilizado o algoritmo do menor caminho de Dijkstra [2] que utiliza os princípios da busca em largura. São descritas nas seções a seguir o funcionamento do algoritmo de busca em profundidade e busca em largura, em seguida, como as implementações destes algoritmos foram utilizadas para gerar um labirinto de caminhos aleatórios e realizar locomoção dentro deste.

2.1 Busca em Profundidade

Dado um grafo orientado $G = (V, E)$. Onde V é o conjunto de vértices e E o conjunto de arestas ligando os vértices. A estratégia utilizada no algoritmo de busca em profundidade consiste em percorrer o grafo, passando por todos os vértices v que ainda tem arestas inexploradas. Quando todos os vértices de um caminho são explorados a busca regressa para explorar outras arestas que deixam o vértice a partir do qual v foi descoberto [1]. O tempo de execução deste algoritmo é da ordem de $O(|V| + |E|)$.

2.2 Busca em Largura

O algoritmo de busca em largura é utilizado para percorrer grafos, explorando sistematicamente arestas vizinhas a partir de um nó origem s, expandindo a fronteira de vértices descobertos uniformemente e explorando a cada iteração os nós vizinhos não visitados. O algoritmo consegue explorar todo o grafo, descobrindo vértices e as distâncias para o vértice origem quando o grafo esta representado em lista de adjacências em $O(|E| + |V|)$ [1]. O algoritmo de Dijkstra utiliza ideias semelhantes em sua estrutura para encontrar o menor percurso em um grafo ponderado a partir de escolhas gulosas dentre os vértices descobertos a cada iteração.

3 Implementação dos Algoritmos

O algoritmo de busca em profundidade, descrito na Seção 2.1, foi utilizado para gerar um labirinto de caminho aleatório desenhados com o OpenGL e as bibliotecas utilitárias GLUT [4]. O algoritmo de Dijkstra foi utilizado para encontrar o caminho entre dois pontos neste cenário. Ao gerar o ambiente e mecânica de movimentação, foi gerado um arcabouço para o desenvolvimento de um protótipo para um jogo de labirinto, mostrados na Seção 3.3.

3.1 Criação do Labirinto de Caminhos Aleatórios

Existem diversos algoritmos para criar labirintos como, por exemplo, a versão aleatória do algoritmo de busca em profundidade, modificações dos algoritmos de estratégia gulosa de Prim

ISSN 2317-3297

e Kruskal [2]. Estes algoritmos conseguem gerar cenários distintos a cada execução já que incluem em seu procedimento escolhas aleatórias para a formação do caminho. O procedimento utilizado neste projeto é uma versão aleatória do algoritmo de busca em profundidade.

O procedimento utilizado para gerar e desenhar o cenário de um labirinto de caminhos aleatórios utiliza os seguintes passos: Dada uma matriz $n \times n$, inicialmente todos os elementos são considerados como paredes, o algoritmo cria caminhos no labirinto realizando uma busca em profundidade em um grafo. Um elemento é escolhido para ser o ponto inicial e a partir de escolhas aleatórias entre os elementos vizinhos (NORTE, SUL, LESTE e OESTE) a busca em profundidade é feita e o caminho do labirinto é formado. A Figura 1a ilustra os passos de escolha dos caminhos e retrocesso ao escolher um caminho sem saída. A Figura 1b ilustra o cenário após desenho na janela com o OpenGL e a GLUT.

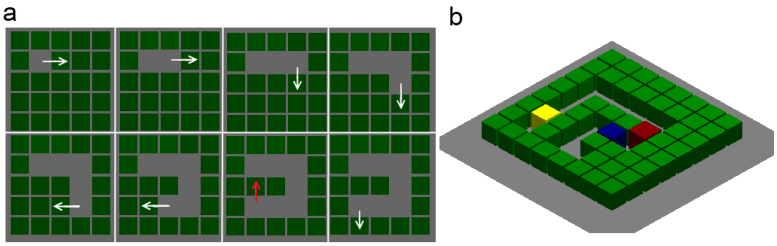

Figura 1: a) Formação dos caminhos do labirinto e b) Perspectiva isométrica aplicada ao labirinto.

3.2 Percorrendo o Labirinto

O problema para percorrer o labirinto é semelhante ao problema de solucionar um labirinto. Marcando dois pontos, o inicial e o final, deve-se realizar o percurso do ponto inicial ao final para soluciona-lo. Existem diversos algoritmos que podem ser utilizados para solucionar um labirinto: o algoritmo da mão direita, o algoritmo de Trémaux e o do menor caminho de Dijkstra são alguns exemplos.

O algoritmo de Dijkstra foi utilizado, pois garante encontrar o menor caminho entre dois nós em um grafo com um custo de $O(|E| + |V|log|V|)$. O problema do menor caminho é uma generalização do problema da busca em largura para grafos com arestas ponderadas [1]. O algoritmo de Dijkstra resolve o problema de caminhos mais curtos de uma única origem em um grafo orientado ao estabelecer decisões gulosas quando escolhe a aresta de menor custo para alcançar a próxima aresta.

3.3 Protótipo de um Jogo

É possível realizar a movimentação do personagem no labirinto do início até o fim, passando por toda estrutura, com o teclado e a movimentação de um NPC *Non-player Character* a partir da utilização do algoritmo do menor caminho descrito na Seção 3.2. Na implementação de um jogo de labirinto foram utilizados estes dois elementos para compor a mecânica do jogo. Enquanto o jogador utiliza as teclas para se movimentar buscando encontrar a saída, um NPC é colocado no outro lado do labirinto e utiliza o algoritmo do menor caminho para encontrar o personagem.

É ilustrado na Figura 2a a estrutura e o relacionamento das funções utilizadas para gerar o cenário de um jogo de labirinto de caminhos aleatórios e a movimentação do NPC através do algoritmo do menor caminho de Dijkstra. Esta estrutura é utilizada para implementar o protótipo de um jogo mostrado na Figura 2b. Os arquivos *Dijkstra.c* e *Labirinto.c* implementam os algoritmos de busca em largura e busca em profundidade através da modelagem dos elementos do labirinto com um grafo. Os objetos 3D que compõem o cenário foram criados em modelos no

ISSN 2317-3297

formato x3d, importados pelas funções contidas em *x3dReader.c* e desenhadas com o OpenGL (*Maingl.c*).

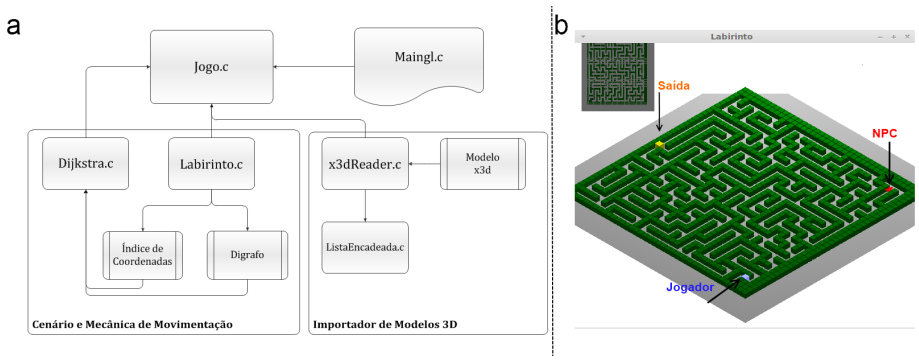

Figura 2: a) Conjunto de funções que formam o arcabouço para a implementação de um jogo de labirinto. b) Execução do protótipo de um jogo, com a movimentação do personagem (Azul) através do teclado e do NPC (Vermelho) com o algoritmo de Dijkstra.

4 Análise de Resultados e Conclusões

Neste artigo foi verificada a importância da aplicação do conhecimento de conceitos fundamentais da ciência da computação, como a teoria dos grafos, busca em largura e em profundidade e como estes podem ser aproveitados em diversas aplicações, incluindo o desenvolvimento de jogos digitais.

Foi verificado que é possível desenvolver jogos digitais a partir dos algoritmos que implementam estas teorias. No presente trabalho foi utilizado o algoritmo de busca em profundidade para criar a estrutura de um labirinto e o algoritmo de Dijkstra, baseado em busca em largura, para realizar a movimentação de NPC's. Com isto foi possível gerar o protótipo de um jogo, mostrado em um cenário isométrico, a partir da implementação destes algoritmos na linguagem C utilizando o OpenGL e as bibliotecas GLUT.

Posteriormente, pretende-se utilizar outros algoritmos existentes para construção de labirintos aleatórios e outras técnicas para resolução de labirintos para obter parâmetros de comparação. A partir deste arcabouço será possível implementar jogos com recursos mais atrativos, como modelos em 3D e o cálculo de pontuações para avaliar o desempenho do jogador.

Referências

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest e C. Stein, Introduction to Algorithms, *The MIT Press e McGraw-Hill*, (2009).

[2] J. A. Bondy e U. S. R. Murty, Graph Theory, *Springer*, (2008)

[3] J. Lloret e T. Kirton, The Maze EV: a two player installation game, *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, (2011) 93:1–93:2.

[4] M. Woo, J. Neider, T. Davis e D. Shreiner, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2 3rd,*Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA* (1999)