Algoritmo dos Retângulos: um *pathfinding* para jogos cliente-servidor

Eduardo D. Lima Christiano L. Santos*

Universidade Federal de Sergipe, DCOMP, Brasil

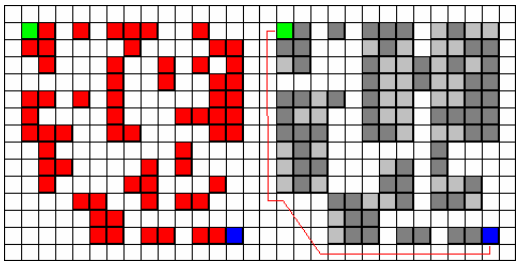

Figura 1: O Algoritmo dos Retângulos em um mapa de jogo convencional.

Abstract

This paper describes the creation of a client-server pathfinding algorithm for games as a faster alternative to traditional algorithms used. It was created looking the minimum possible complexity, targeting not the best way, but any good possible way. Therefore, it will be able to be implemented in servers, helping and monitoring the clients.

Resumo

Esse artigo descreve a criação de um algoritmo de *pathfinding* voltado para jogos cliente-servidor como uma alternativa mais rápida aos algoritmos tradicionais usados. Ele foi criado de uma forma que o seu tempo de execução seja o mínimo possível, visando não o melhor caminho, mas sim encontrar um bom caminho possível. Sendo assim, ele será capaz de ser implementado em servidores, auxiliando e policiando os diversos clientes.

Palavras chave: pathfinding, jogos, inteligência artificial, mapa, cliente-servidor

Contatos:
doriaeduardo@gmail.com
*christianolimasantos@yahoo.com.br

1. Introdução

Algoritmos de *pathfinding*, além de outras, possuem uma grande utilidade em jogos de computadores. Praticamente todo jogo possui personagens não controlados por humanos que precisam ser capazes de se deslocar pelo mapa chegando de um ponto a outro e demonstrando assim ter comportamento aparentemente inteligente. Até mesmo em personagens controlados por humanos, existem situações onde o usuário não o controla passo a passo, ele apenas escolhe um ponto no mapa e o jogo direciona-o até determinado ponto, seguindo uma trajetória correta e às vezes até a mais rápida de todas [Buckland 2002; Rabuske 1995].

Geralmente um algoritmo de *pathfinding* possui uma alta complexidade. Isso acontece devido a uma grande quantidade de informações que devem ser processadas e à enorme possibilidade de caminhos que ele pode seguir. Um fato interessante é que a grande maioria dos algoritmos procura sempre o melhor ou mais rápido caminho, dando a impressão de que o personagem do jogo possui uma capacidade de raciocínio sobre-humana.

Nesse artigo é apresentado o Algoritmo dos Retângulos, que foi desenvolvido visando ter um custo computacional muito abaixo dos algoritmos tradicionais, porém com resultados satisfatórios. É demonstrado também seu uso em jogos cliente-servidor e como ele pode ser mais eficiente que outros.

2. Estado da Arte

Muitos fatores devem ser levados em conta quando se escolhe um algoritmo de *pathfinding* no desenvolvimento de um jogo de computador. Um dos fatores principais que afetam a "inteligência" do algoritmo é o tipo do mapa. O desenvolvedor de jogo precisa saber exatamente qual o tipo de mapa que seu jogo terá. Se forem mapas no estilo labirintos, ou mapas com objetos pequenos e esparsos, como vegetação, ou até construções um pouco mais densas. Outro fator importante é o número de vezes que o *pathfinding* é usado – alguns jogos possuem uma infinidade de personagens não controlados por humanos e poucos personagens controlados, em outros jogos isso se inverte. Logo, tudo influencia no tipo do algoritmo que deve ser usado. Se ele deve ser mais complexo e menos eficiente computacionalmente falando ou se deve ser mais simples e muito mais eficiente [Rabuske 1995].

2.1 Método A*

A solução mais famosa e usada atualmente nos jogos é o algoritmo A*. Ele não possui uma grande complexidade no caso médio e é simples de ser

implementado. Seu funcionamento baseia-se em achar o melhor vizinho para chegar a determinado destino [Rich e Knight 1993].

Num jogo de computador geralmente o personagem se encontra em um mapa dividido em blocos e esses blocos são separados em dois tipos: os livres e os bloqueados. Nos blocos livres o personagem pode caminhar, nos bloqueados ele não pode, pois ali existe algum objeto, como paredes ou outros personagens. Essa divisão em regiões é feita a fim de melhorar o desempenho do algoritmo, pois se a análise e cálculo fossem feitos pixel a pixel a precisão do movimento seria ideal, porém o tempo de processamento do algoritmo seria muito maior.

Desta forma, a maior parte dos jogos convenciona a movimentação do personagem em oito direções (nos eixos vertical, horizontal e diagonais), permitindo assim que, a partir de um dado bloco do mapa possa chegar a outro.

O algoritmo A* determina a melhor opção por meio do processamento e combinação de dois elementos: o custo para chegar ao quadrado vizinho e o custo estimado (heurística) para chegar deste ao destino [Lester 2005].

2.2 Método Dijkstra

Esse método é menos usado que o A*, porém ainda assim é outro dos mais usados em jogos. Ele tem um funcionamento semelhante ao A* e ambos utilizam grafos como estrutura de dados.

O funcionamento do Dijkstra se baseia em tentar achar o caminho mais curto entre dois vértices do grafo. Para fazer isso ele monta toda a estrutura do mapa pela descrição geométrica, criando um grafo com o peso de cada aresta representando as distancias. Em seguida, ele através do grafo, calcula a menor distancia que deve ser percorrida para chegar do vértice origem ao vértice destino [Graham et al. 2003].

3. O Algoritmo dos Retângulos

Normalmente em jogos cliente-servidor, o servidor precisa ficar monitorando os passos dos clientes a todo o momento: sempre haverá pessoas com más intenções de trapacear no jogo e cabe ao servidor impedir esse tipo de ação. Algumas maneiras de saber se um jogador movimentou seu personagem de um ponto a outro sem trapacear é: informando sua posição passo a passo ou implementando o próprio algoritmo de *pathfinding* usado no cliente, também no servidor. O primeiro caso exige um grande consumo da banda, já o segundo requer um maior processamento e consumo de memória no servidor. Cabe então ao desenvolvedor escolher a melhor alternativa levando em conta os três fatores: memória utilizada, banda necessária e tempo de processamento.

O Algoritmo dos Retângulos é uma alternativa aos outros algoritmos de *pathfinding* e foi desenvolvido para ter a mínima complexidade computacional com uma boa qualidade de soluções. Com sua complexidade baixa, ele torna possível a sua implementação em servidores sem requerer tanta capacidade de processamento.

Outro fato interessante é que diferentemente da maioria dos algoritmos, ele não acha o melhor caminho sempre, mas sim procurava um bom caminho, de forma similar a como um ser humano procuraria, tornando assim o comportamento obtido mais realista.

3.1 Procurando o Destino

Como foi dito anteriormente, o mapa deve ser divido em regiões (blocos) iguais, e essas regiões são de dois tipos: livres e bloqueadas. A idéia inicial desse algoritmo é conhecer o ponto no mapa aonde se quer chegar. Sabendo isso, o personagem sai de sua posição inicial e segue para o próximo bloco vizinho que seja mais próximo do bloco destino. Esse procedimento é feito sucessivamente até achar alguma barreira.

Quando o próximo bloco vizinho escolhido for um dos considerados bloqueados, o mesmo não poderá fazer parte do caminho. O personagem precisa então, antes de seguir, saber um modo de desviar desse bloqueio. A idéia do algoritmo para saber a melhor direção é “olhar” para os lados e ver qual é o lado mais curto para sair desse bloqueio. No caso da Figura 2, percebe-se claramente que a melhor direção a seguir é para baixo, pois o personagem precisa atravessar dois blocos para baixo, contra seis para cima.

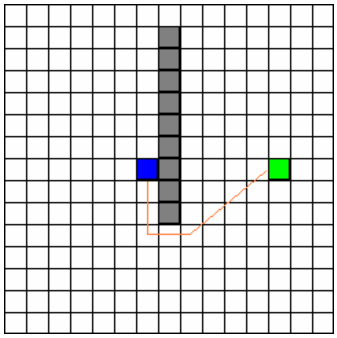

Figura 2: Personagem (azul) tentando chegar ao destino (verde)

3.2 Formando Retângulos

O procedimento descrito anteriormente funciona muito bem, porém em alguns casos ele não vai saber “sair de uma barreira”. Esses casos ocorrem quando tanto um lado como o outro dela não estão livres. Observando a Figura 3 consegue-se notar que ambos os lados da barreira estão bloqueados, logo o personagem não consegue calcular o melhor lado para sair ficando, dessa forma, parado.

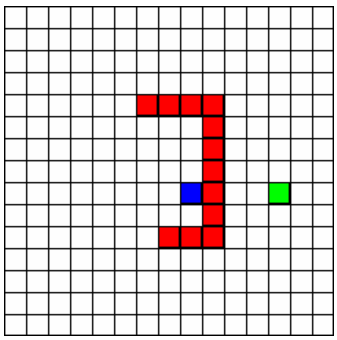

Figura 3: Personagem (azul) tentando chegar ao destino (verde)

Casos como esse devem ter um tratamento diferente: o algoritmo precisa fechar todos os objetos formando retângulos, preenchendo-os internamente e eliminando a possibilidade de o personagem entrar neles. O resultado do procedimento pode ser observado na Figura 4. Nesse caso o personagem não vai nem chegar a entrar no objeto, pois ele fará uma trajetória contornando a parte externa.

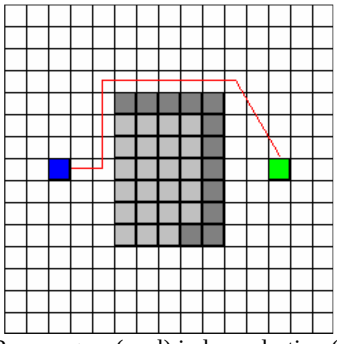

Figura 4: Personagem (azul) indo ao destino (verde)

3.3 Saindo dos Retângulos

Praticamente todos os tipos de mapa podem ser resolvidos até agora, porém existem situações em que, ou o personagem ou o destino estão em uma área que vai ser usada para formar um retângulo. Esses casos devem ser resolvidos encontrando a saída/entrada de cada retângulo. A saída/entrada deve ser o ponto fora do retângulo mais próximo do destino. Como pode ver na Figura 5, o personagem sabe exatamente para onde seguir.

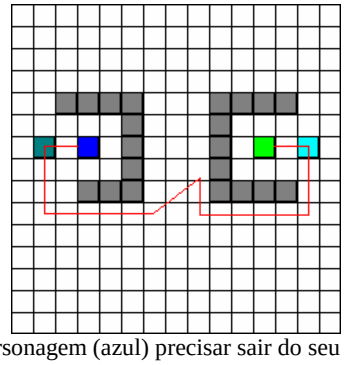

Figura 5: Personagem (azul) precisar sair do seu retângulo para chegar ao destino (verde)

Para evitar cálculos desnecessários, só são calculadas as saídas/entradas dos retângulos em que estão o personagem ou o ponto de chegada. Além disso, quando o personagem estiver na saída do seu retângulo, ele deve ser proibido de voltar a entrar, o que o obriga a dar a volta e ir em direção à entrada do próximo retângulo, pois caso ele entre novamente, ficaria preso.

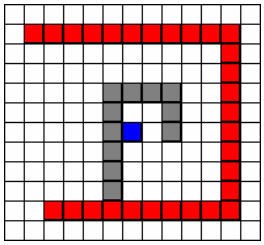

Figura 6: Exemplo de retângulo recursivo

Na Figura 6 é possível observar que o retângulo principal (em vermelho) está encobrindo outro retângulo menor (em cinza). Se o algoritmo fosse feito verificando apenas os pontos adjacentes para formar um retângulo, ele formaria um retângulo só, bem grande, nesse mapa e dependendo da posição do personagem, ele talvez nunca conseguiria chegar ao ponto de destino (em azul). O que deve ser feito é usar esse mesmo algoritmo, porém recursivamente. Assim que achar o primeiro retângulo (em vermelho) elimina-o e faz o mesmo procedimento para os retângulos internos, até não sobrar mais nenhum. No exemplo da imagem, dois pontos de saída/entrada serão encontrados: um referente ao retângulo vermelho e outro referente ao retângulo cinza.

Os pontos de entradas/saídas encontrados devem ser adicionados a uma pilha dependendo da ordem que vão ser buscados. Cada ponto atingido pelo personagem deve ser eliminado da pilha, forçando-o a buscar o próximo ponto. Essa busca só é terminada quando não existem mais pontos de entrada/saída na pilha, lembrando que o último ponto é o destino final do personagem.

3.4 Solucionando os Mapas

De acordo com as técnicas supracitadas, o algoritmo provavelmente seria capaz de solucionar todos os mapas de objetos esparsos. Mapas no estilo labirinto, onde muitas paredes são adjacentes umas às outras, o algoritmo encontraria muitos pontos de entrada/saída que talvez confundissem o personagem. Em outros casos, mesmo mapas grandes, porém com objetos esparsos, como os observados em muitos jogos massivamente multi-usuários, o Algoritmo dos Retângulos resolveria muito bem.

O fato de o Algoritmo dos Retângulos ser recomendado para usar em servidores é que todo o processamento da formação dos retângulos das partes fixas do mapa pode ser feito antes de iniciar a partida. Desta forma, se nem o personagem nem o local destino estiverem dentro de algum retângulo já formado, o algoritmo faria somente uma busca simples. Caso algum deles esteja dentro de algum retângulo, o

procedimento é feito conforme fora explicado, identificando as saídas/entradas recursivamente e colocando-as na pilha.

3.5 Unidades Móveis

O procedimento usado para desviar das unidades móveis do mapa (outros personagens, veículos, etc.) é praticamente o mesmo daquele usado acima nas partes fixas. A principal diferença é que o cálculo das regiões dos retângulos não poderá ser feita com antecedência.

A idéia nesse caso é deixar o personagem andar pelo mapa independente da posição das partes móveis, desviando-se de todos os objetos que tiverem a forma de retângulo, pois os cálculos já foram feitos. Quando encontrar algum objeto que não consiga passar, provavelmente esse objeto é formado por partes móveis. A partir desse momento, pega como referência o ponto de colisão e é feito o mesmo cálculo dos retângulos descrito acima. Dessa forma, os novos pontos de entrada/saída são adicionados à pilha.

Usando essa idéia, o personagem pode adquirir uma postura não muito inteligente, pois quanto às unidades móveis, ele primeiro colide para depois descobrir como desviar. Porém, levando-se em conta que no tipo de jogo em que este algoritmo será empregado, as partes móveis raramente passam de um bloco único no mapa e quando passam ocupam áreas retangulares, o resultado do algoritmo será pouco afetado.

3.5 Outras Soluções

O Algoritmo dos Retângulos foi desenvolvido para ser eficiente e, dessa forma, ser usado além dos clientes, também em servidores, policiando e monitorando as trajetórias de seus usuários, como já foi dito. Nos clientes, algumas abordagens diferentes podem ser usadas. Uma dela é o cálculo dos retângulos sendo feito em tempo real. Às vezes não é vantagem gastar um tempo maior fazendo o pré-processamento do mapa completo, então só é calculada a parte que vai ser usada em determinado momento.

Uma boa alternativa de calculo dos retângulos em tempo real é a “tentativa e erro”. O algoritmo faz uma trajetória pelo mapa até encontrar um obstáculo que não consegue passar. Caso encontre, é feito o cálculo do retângulo que pertence ao ponto que foi colidido. Depois volta até o ponto inicial e refaz a trajetória. Esse procedimento só termina quando a trajetória chega ao ponto final, fazendo assim o caminho correto, fazendo o personagem andar sobre ele.

Um fato interessante a ser observado é o formato da trajetória, uma vez que o personagem somente desvia de um obstáculo quando o toca. Isso às vezes deixa o movimento muito artificial, como pode ser observado na Figura 7.

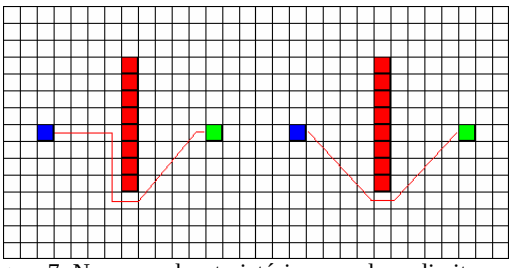

Figura 7: Na esquerda a trajetória normal, na direita a trajetória ideal

Uma solução para esse problema seria fazer a trajetória em ambos os sentidos: do início para o fim e do fim para o início. Posteriormente selecionam-se os pontos mais eficientes de cada uma. Desse jeito teria uma trajetória mais próxima daquela que seria escolhida por um humano.

4. Conclusão

Pôde-se observar que o Algoritmo dos Retângulos é uma alternativa rápida e eficiente se comparada aos algoritmos tradicionais. É fácil observar que seu caso médio vai depender diretamente do mapa usado no jogo. Porém, em muitos casos, será uma busca simples, o que permite sua implementação eficiente nos servidores. Deste modo, será possível controlar passo a passo a movimentação dos personagens, sabendo se houveram fraudes no movimento realizado por eles.

Além disso, o Algoritmo dos Retângulos também se mostrou como uma boa alternativa na movimentação de personagens não controlados por humanos, podendo ser empregado, inclusive, em outros tipos de jogos.

Como futura extensão, espera-se aplicar o algoritmo desenvolvido em um projeto de jogo massivamente multiusuário e estudar os resultados obtidos.

Referências

BUCKLAND, MATH, AI Techniques for Game Programming. Game Developer Series, Premier Press, 2002.

GRAHAM, ROSS, McCABE, HUGH, SHERIDAN, STEPHEN, Pathfinding in Computer Games, ITB Journal, 2003.

LESTER, PATRIC., 2005. A* Pathfinding para Iniciantes. Traduzido por Raynaldo N. Gayoso. Disponível em: <http://www.policyalmanac.org/games/aStarTutorial_port.htm>. Acesso em: 18 AGO. 2008.

RABUSKE, RENATO A., Inteligência Artificial, Editora da UFSC, 1995.

RICH, ELAINE, KNIGHT, KEVIN, Inteligência Artificial, MAKRON BOOKS, 1993.