3º ano do Mestrado Integrado de Engenharia Informática e Computação, 2014/2015

Inteligência Artificial

Box World 2

Relatório Final

Francisco Lopes, 201106912
Miguel Mendes, 201105535

31/05/2015

Objetivo:

O objetivo deste trabalho consiste em desenvolver um programa que demonstre inteligência suficiente para resolver o problema em questão. Focando-nos no jogo "Box World 2", pretendemos que, em vez de ser o jogador a resolver os puzzles propostos, que o computador seja capaz de o fazer autonomamente, encontrando caminhos através dos mesmos, e empurrando quaisquer blocos necessários para que tal seja possível.

Especificação:

1) Análise

O nosso trabalho é baseado no jogo "Box World 2". Neste jogo, o objetivo do jogador é simplesmente chegar à saída, desta forma conseguindo avançar para o nível seguinte. No entanto, os caminhos para a saída poderão estar bloqueados por buracos que impeçam a travessia até ao objetivo. Para resolver isto, existirão vários blocos que podem ser deslocados pelo jogador, e colocados nestes buracos para que chegar ao objetivo seja ultimamente possível. O desafio neste jogo reside no quão complexos poderão ser os movimentos do jogador, em congruência com a colocação destes blocos, para que este consiga resolver os vários puzzles. Além dos blocos mais comuns, que apenas movem uma unidade quando o jogador os empurra, ainda existe uma segunda categoria de blocos, blocos de gelo, que se deslocam ininterruptamente na direção em que são empurrados até que encontrem uma parede ou bloco.

O nosso objetivo consiste em desenvolver inteligência para o programa, de forma a que não seja necessário um jogador, e o computador seja capaz de resolver os vários puzzles apresentados, presumindo que existe solução, respeitando todas as regras implícitas ao correr do jogo.

2) Abordagem

O algoritmo principal no qual o desenvolvimento deste projeto se foca é o A*. Este algoritmo é vastamente utilizado para encontrar caminhos, principalmente em situações onde o deslocamento se efetua em grelhas bidimensionais, como é o caso. Cada nó da grelha, que poderá ser também um nó do algoritmo, tem um valor f(x), resultante da soma de dois outros valores. O primeiro, g(x), será correspondente à distância do nó inicial ao nó que se analisa na altura. O segundo valor, h(x), consiste na função heurística através da qual realizamos uma previsão da distância que será necessário percorrer do nó atual até ao objetivo.

Partindo do nó inicial, calcula-se o f(x) dos seus vizinhos. Aquele que possuir o menor f(x), ou seja, o maior potencial de nos levar pelo caminho mais curto, terá os seus vizinhos analisados, que se juntam ao grupo de nós não selecionados até agora, de forma a que se encontre de novo o nó com o menor valor de f(x), e assim sucessivamente, até o nó selecionado ser o objetivo.

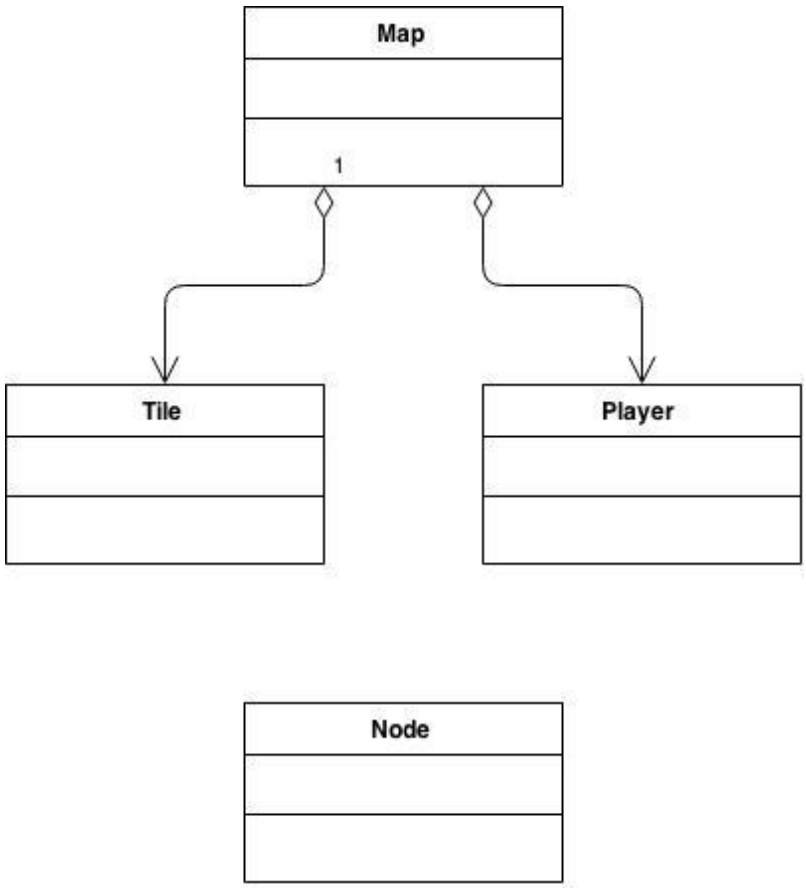
O nosso plano consiste em aplicar o algoritmo A* para que o programa encontre o caminho mais curto para a saída, com o menor número de buracos presente possível nessa tentativa de solução. A partir daí, utilizamos as caixas que nos estão disponíveis, caso sejam suficiente, para preencher os buracos que se apresentam como inevitáveis na nossa tentativa de solução. Assim, após assignar cada caixa ao seu buraco, aplicaremos um A* adaptado a caixas, de forma a colocar cada uma no buraco que lhe foi assignado. Esta versão do algoritmo tem uma restrição adicional, que consiste em que a direção em que a caixa se move deverá ter um espaço vazio na direção contrário, para que o "jogador" se possa colocar de forma a empurrar a caixa pelo caminho ultimamente encontrado pelo algoritmo. Após a colocação das caixas necessárias, aplica-se o algoritmo A* uma última vez para a final deslocação para a saída, agora que todos os buracos necessários de se resolver se encontram preenchidos.

Desenvolvimento:

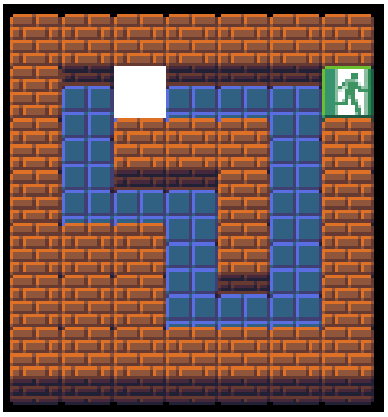
A nossa aplicação foi desenvolvida em C++, em Windows, fazendo uso da biblioteca SFML 2.0 para a criação da interface gráfica.

A maior parte de todo o processo encontra-se concentrado no módulo relativo ao mapa, que alberga não só as estruturas que representam o problema no ecrã, assim como as funcionalidades relacionadas como algoritmo responsável pela resolução do problema. Este módulo contém também o módulo do "jogador", que neste caso permitirá a representação do progresso da inteligência, que tenta resolver o problema proposto. Outro módulo importante será o módulo do nó, estrutura de conhecimento principal utilizada durante o algoritmo para criar a árvore que se torna crucial na procura da solução.

Em seguida apresentamos uma versão simplificada da estrutura geral do nosso programa, necessária para a resolução do problema:



Experiências:



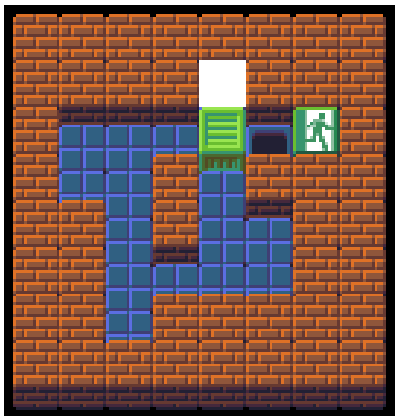
Inicialmente, as experiências, têm que ser básicas, para confirmação da plena funcionalidade do algoritmo A*. Neste exemplo, demonstramos como apresentamos ao programa a possibilidade de perseguir dois caminhos diferentes para chegar ao seu destino. Fora error técnicos iniciais que fariam com que o programa tomasse caminhos inválidos, confirma-se que o programa é capaz de selecionar o caminho mais curto. Neste caso, desloca-se para a direita.

Conclusões:

Através das experiências levadas a cabo, entendemos que o algoritmo implementado de A* é perfeitamente capaz de encontrar o caminho mais curto para o objetivo de qualquer labirinto, posto que este tenha uma solução possível.

Melhoramentos:

Devido a condições fora do nosso controlo, não foi possível implementar mais do que a capacidade do programa de levar a cabo a viagem mais curta entre quaisquer dois pontos selecionados no mapa. Os algoritmos necessários à inclusão de caixas está existente. Apenas seria necessário adaptar a componente gráfica para que esta restante funcionalidade fosse suportada.



Durante a maioria do trabalho foram apenas consideradas situações em que a colocação de caixas era bastante linear. De futuro, seria vantajoso considerar e testar situações em que o movimento de caixas necessita ser algo mais complexo. Como exemplo, temos a situação demonstrada na imagem, na qual seria necessário deslocar a caixa de forma a realizar uma espécie de ciclo, para que o único buraco que bloqueia a saída fosse eliminado.

Recursos:

1) Bibliografia

http://en.wikipedia.org/wiki/A*_search_algorithm

2) Software

Todo o código necessário à realização do projeto foi desenvolvido com a utilização de Visual Studio.

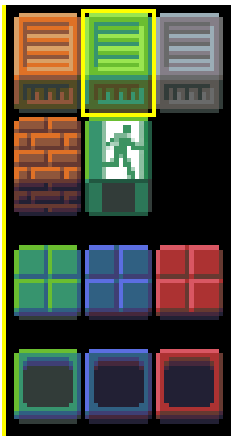
3) Elementos do grupo

Francisco Lopes: 10%

Miguel Mendes: 90%

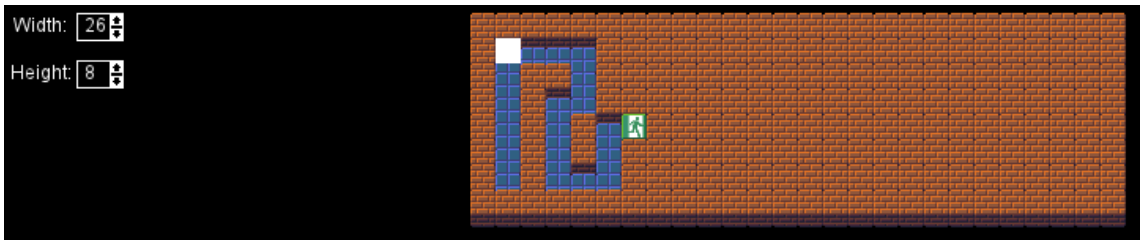
Apêndice:

Aqui, pode aprender a utilizar a aplicação, e a criar o seu próprio labirinto, que o programa será capaz de resolver!



Selecione o elemento que deseja colocar no mapa. Pode colocar uma saída, uma caixa, um pedaço de chão, uma parede, ou um buraco. Apenas selecione o elemento que deseja colocar e clique no local do mapa que pretende alterar.

Se assim o pretender, pode também alterar as dimensões do seu mapa, clicando nas setas que alteram os valores para esse efeito.



Por fim, premindo ‘P’ enquanto clica no mapa colocará o jogador onde deseja.

Para terminar, prima “Enter” quando pensa que tem um mapa completo com uma solução possível, e o labirinto será resolvido.