Ms. PacMan – Abordagem Euler e Giovanni

Giovanni Jakubiak de Albuquerque[1], Euler Giachini[1]

[1]Universidade do Estado de Santa Catarina - UDESC
[1]Departamento Engenharia de Software
Centro de Educação do Alto Vale do Itajaí (CEAVI) – Ibirama, SC – Brazil

giovannijakubiak@hotmail.com, eulergiachini@gmail.com

Abstract. *In this article, we will present the game Ms. PacMan scenario, as well demonstrating the main objectives proposed on the work. We will show the solutions developed by the team to solve de problem, detailing it. We will analyze the obtained results, comparing it with each other and with the existing techniques.*

Resumo. *Neste artigo iremos apresentar o cenário do jogo Ms. PacMan, bem como a demonstração dos objetivos propostos no trabalho. Apresentaremos as soluções desenvolvidas pela equipe para a resolução do problema, detalhando-as. Analisaremos os resultados obtidos, comparando uma com a outra e também com as técnicas já existentes.*

1. Introdução

Este trabalho compõe a nota final da matéria de Inteligência computacional como sendo o trabalho 1 da disciplina. Ele consiste em criar uma forma de aplicar algoritmos de buscas (informadas ou não-informadas) ou metaheurísticas para realizar os comportamentos do personagem principal do jogo, Ms. PacMan.

O cenário do jogo é o típico PacMan, que ao invés de ser o próprio pacman, o ator principal é a Ms. PacMan. O jogo é basicamente igual ao original em todos os aspectos. O jogo consiste em um personagem que come pílulas para ganhar pontos, ao comer todas as pílulas o jogo termina. Outro objetivo é fugir nos *gosths,* que são os inimigos do jogo, ao tocar em um inimigo o jogo acaba. Outro ponto importante são as *powerPills,* que permitem ao personagem principal destruir os inimigos por um tempo determinado.

O framework disponibilizado para a realização do trabalho contém diversos métodos para serem utilizado na construção do comportamento do agente. Os métodos disponibilizados executam consultas no cenário do tipo buscar as *powerPills* mais próximas ou verificar se tem algum *gosth* próximo.

Na segunda seção iremos apresentar as estratégias e algoritmos utilizados para a criação das duas lógicas de jogo. Na terceira iremos mostrar os resultados obtidos, detalhando-os e comparando com as outras implementações já existentes no cenário. E por fim as conclusões obtidas com os experimentos.

2. Estratégias

2.1. Estratégia criada pelo Euler

Este algoritmo possui algumas variáveis para controle dos possíveis caminhos que a nossa Ms. Pac-Man irá fazer. Podemos citar inicialmente a variável ‘move’, onde atribuímos o destino da nossa Ms. Pac-Man a cada ciclo no nosso código. Essa variável é iniciada nula e só lhe é atribuído um valor após a construção da nossa árvore de caminhos possíveis e suas verificações se possui Ghost’s no caminho e outras.

A primeira coisa que fazemos no nosso algoritmo é verificar os Ghost’s e guarda-los em um ArrayList de Ghot’s. Após isso iniciamos a construção da nossa árvore de possíveis caminhos através do método ‘construirArvore’ no qual nos devolve, de forma simplória o atual índice da nossa Ms. Pac-Man.

Para a construção da nossa árvore criamos uma classe de nome ‘No’ com alguns atributos para podermos gerenciá-la melhor. No método ‘construirArvore’ nós fazemos uma busca nos vizinhos de nosso index, que se inicia na atual posição da Ms. Pac-Man e vai adicionando os filhos destes vizinhos em um ArrayList de nós e excluindo os pais desses filhos conforme vai pesquisando os filhos dos filhos em um limite de 1000 interações.

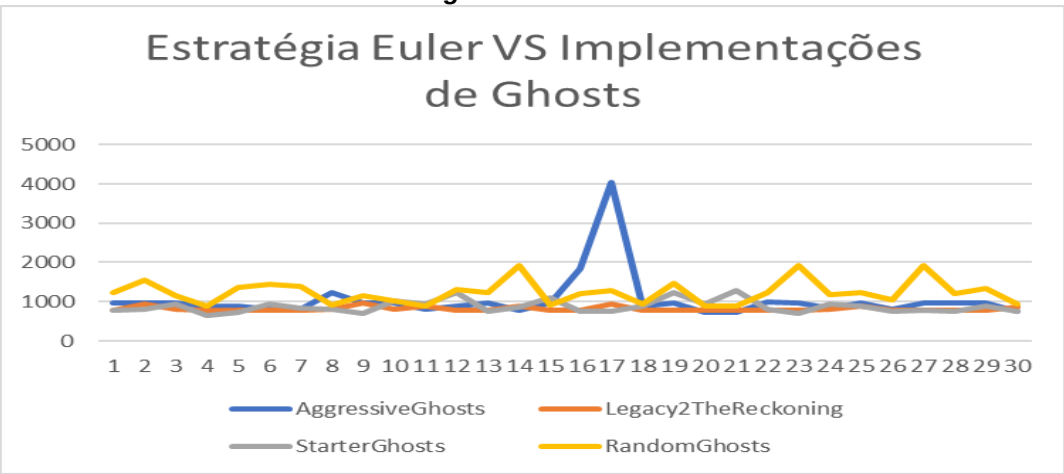
Após a construção da árvore faremos uma verificação dos possíveis caminhos através do método ‘verificaCaminho’. Este método consiste basicamente em utilizar a nossa árvore criada no método ‘construirArvore’ e verificar se existe algum Ghost em uma área previamente determinada.

Caso haja um Ghost dentro da área determinada iremos definir como ‘destino’ da nossa Ms. Pac-Man a PowerPill mais próxima, caso não haja mais PowerPills ativas no nosso jogo o nosso destino será a primeira pill no primeiro índice do nosso vetor de pills. Caso não haja nenhum Ghost na nossa área de busca iremos definir como destino que a Ms. Pac-Man fique coletando as pills seguindo a ordem no vetor de pills.

Após definição do destino da nossa Ms. Pac-Man iremos passar esse comando para a variável ‘move’ para que ela determine o destino da nossa Ms. Pac-Man.

Os resultados obtidos da abordagem contra cada uma das implementações de *Ghosts* seguem no gráfico 1:

Gráfico 1: Comparativo da implementação Euler com todas as implementações de *ghosts.*



Como podemos perceber os resultados obtidos durante 30 partidas de Ms PacMan, cada uma com uma implementação diferente de *Ghosts,* a pontuação ficou com uma média entre 0 pontos e 2000 pontos. Na execução de número 17 obtivemos um resultado recorde de 4030.

2.2. Estratégia criada pelo Giovanni

Esta estratégia consiste basicamente em determinar uma pontuação para o caminho mais distante do *ghost* mais próximo da Ms PacMan, sempre buscando as *pills* e as *powerPills.* Foi utilizada uma busca em largura com uma profundidade pré-definida para determinar o melhor o melhor galho da árvore.

Para determinar a pontuação do caminho, foi pontuado primeiramente cada posição no mapa que o personagem quer acessar. Cada nodo dentro do limite de profundidade tem seus filhos gerados, desprezando o ponto por onde ele já passou. Quando um novo ponto é descoberto, ele é adicionado à arvore da busca como sendo uma entidade do tipo *Node* que possui os atributos: noPai (do tipo Node), noAtual (do tipo int), nosFilhos (do tipo ArrayList), valorCaminhoAteAqui (int) e nível (int).

A pontuação armazenada na variável valorCaminhoAteAqui do nodo na posição correspondente. Para definir a pontuação primeiramente é analisado o quão longe está o *ghost* que está mais próximo do agente. Em seguida verifica-se se na posição tem uma *pill* ou uma *powerPill*. Cada um desses condicionais gera uma pontuação para o nodo que vai influenciar no final de cada execução da busca qual galho vai ser escolhido.

O primeiro condicional para atribuir os pontos, confere a posição de cada *ghost,* compara elas entre si, verifica qual o mais próximo da Ms PacMan e retorna o valor da distância do *ghost* mais próximo. Tendo este valor confere-se dentro de qual *range* de valores ele se encaixa, conforme a tabela 1:

Tabela 1: Valor da pontuação de acordo com a distância do *ghost.*

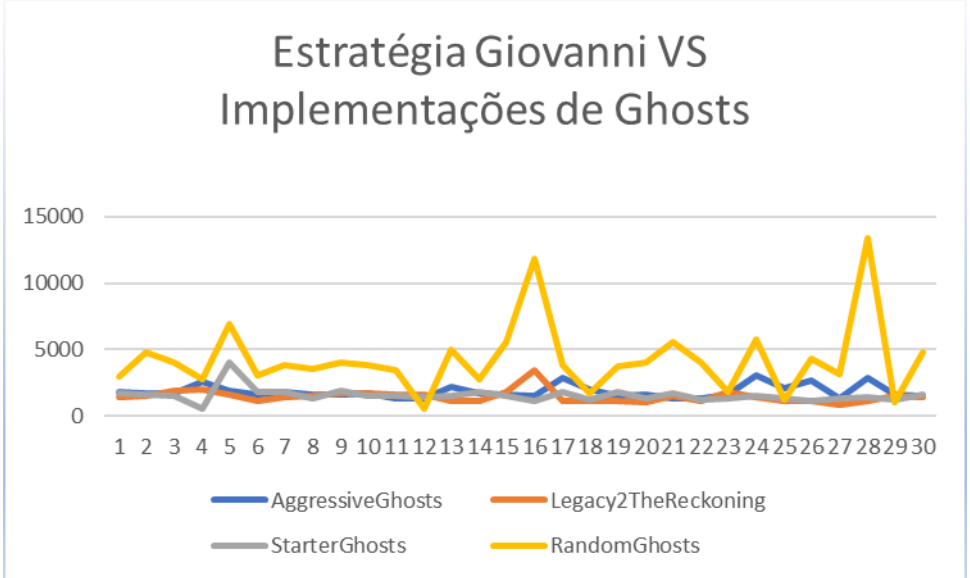
Distância (d)	Valor
d >= 900	300
900 > d > =500	250
500 > d > = 300	200
300 > d > = 150	170
150 > d > = 100	140
100 > d > = 80	120
80 > d > = 60	100
60 > d > = 40	80
40 > d > = 30	60
30 > d > = 20	30
20 > d > = 10	10
d < 10	1

Após atribuir uma pontuação de acordo com a tabela 1, verifica-se se na posicão correspondente se existe uma *pill,* caso exista é somado 320 pontos à pontuação. E então confere-se se há uma *powerPill* na posição, caso haja, atribui-se mais 450 pontos ao nodo. Sabendo que não existe uma p*owerPill* na mesma posicão de uma *pill,* a seleção dos nodos fica sempre balanceada. A definição dos valores foi feita com testes de combinação de valores até encontrar algum que otimizava a pontuação.

A descoberta de novos galhos na arvore de busca encerra quando chega em um limite pré-estabelecido de 110. Quando todos os nodos deste nível limite estão descobertos é verificado qual deles tem o maior valorCaminhoAteAqui. Selecionado o galho, montamos a lista do caminho a ser percorrido, que é simplesmente ir pegando o pai do nodo acima, até que o pai seja nulo.

Executando este algorítmo contra todos as implementações de fantasmas, obtivemos os resultados expressos no gráfico 2:

Gráfico 2: Comparativo da implementação Giovanni com todas as implementações de *ghosts.*

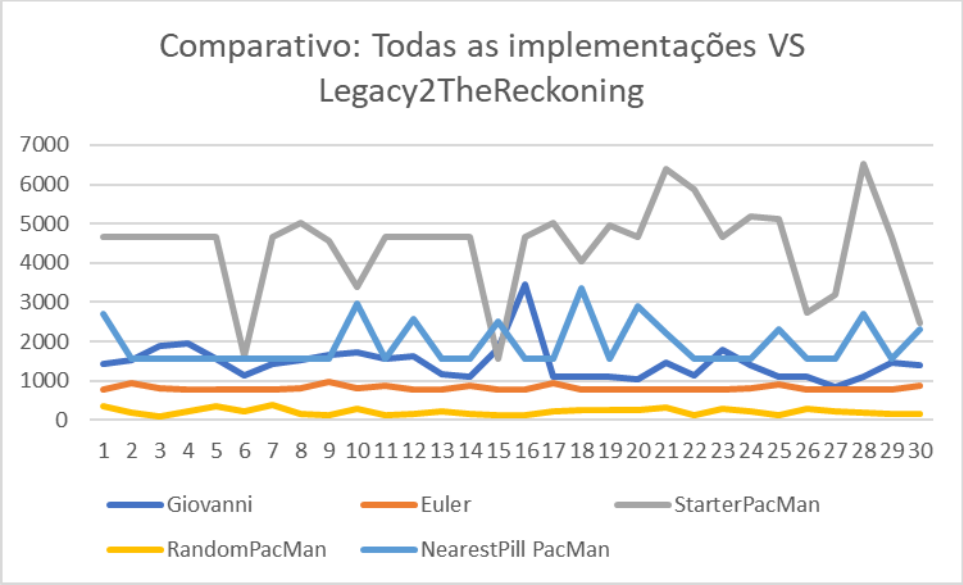


Podemos perceber que a média da pontuação ficou entre 0 e 5000 pontos. A estratégia teve um melhor resultado nas partidas contra os *RandomGhosts*, as quais ficaram com uma pontuação entre exatamente 13400 pontos e 540, o que é um resultado muito interessanta dada a discrepância dos valores.

3. Resultados obtidos

Nesta seção iremos comparer os resultados das estratégias, tanto dos autores quanto as que já vieram de exemplo no próprio código.

Gráfico 3: Comparativo de todas as implementações contra *Legacy2TheReckoning*



No gráfico 3, podemos perceber que a implementação *StarterPacMan* foi a que se saiu melhor contra a implementação de *ghosts, Legacy2TheReckoning.* Foi escolhida esta implementação de *ghosts* pois foi a que gerou menores resultados, nas partidas contra as implementações de Ms PacMan, em comparação com as outras implementações de *ghosts,* logo sendo a mais difícil de disputar.

Calculamos as médias e o desvios padrão de cada implementação. Desvio padrão serve para dizer o quanto os valores dos quais se extraiu a média são próximos ou distantes da própria média (WOLFFENBÜTTEL, 2006). Obtendo a seguinte tabela:

Tabela 2: Comparativo de todas as implementações contra o *Legacy2TheReckoning*. Com medias e desvio padrão

Partida\Implem.	Giovanni	Euler	StarterPacMan	RandomPacMan	NearestPill PacMan
1	1420	790	4650	340	2690
2	1530	940	4650	190	1570
3	1880	800	4650	80	1570
4	1950	790	4650	220	1570
5	1560	790	4650	340	1570
6	1120	790	1630	210	1570
7	1430	790	4650	370	1570
8	1520	810	5020	170	1570
9	1670	970	4550	130	1570
10	1730	810	3380	280	2950
11	1570	890	4650	120	1570
12	1640	790	4660	160	2570
13	1160	790	4650	220	1570
14	1110	890	4650	150	1570
15	1830	790	1560	110	2520
16	3460	790	4650	140	1570
17	1110	940	5020	230	1570
18	1110	790	4050	260	3350
19	1110	790	4940	250	1570
20	1040	790	4650	250	2890
21	1460	790	6400	310	2220
22	1120	790	5880	110	1570
23	1790	790	4650	280	1570
24	1400	810	5170	230	1570
25	1110	900	5130	110	2320
26	1110	790	2730	290	1570
27	840	790	3190	230	1570
28	1110	790	6520	180	2690
29	1460	790	4650	150	1570
30	1390	870	2490	170	2320
Media	1458	821,3333	4425,666667	209,3333333	1930,666667
Desvio padrão	476,7953	55,00679	1153,051972	77,27841887	553,0974431

4. Conclusões

Concluímos que a aplicação de buscas informadas para a descoberta dos melhores caminhos para percorrer, é eficaz. Tudo depende da quantidade e da qualidade de fatores que iremos levar em consideração, porém tudo isso pode influenciar no desempenho da função. A execução do algoritmo do Giovanni é lenta, pois explora a árvore inteira e com uma profundida de 110 com aproximadamente 3700 nodos. Já a estratégia do Euler se limita aos 1000 primeiros nodos filhos sempre excluindo seu nó pai, o que gera uma grande economia de memória e alta velocidade.

References

Wolffenbüttel, Andréa. **O que é? Desvio padrão.** Edição 23, ano 3, 2006 Acesso em: <http://desafios.ipea.gov.br/index.php?option=com_content&view=article&id=2104: catid=28&Itemid=23>