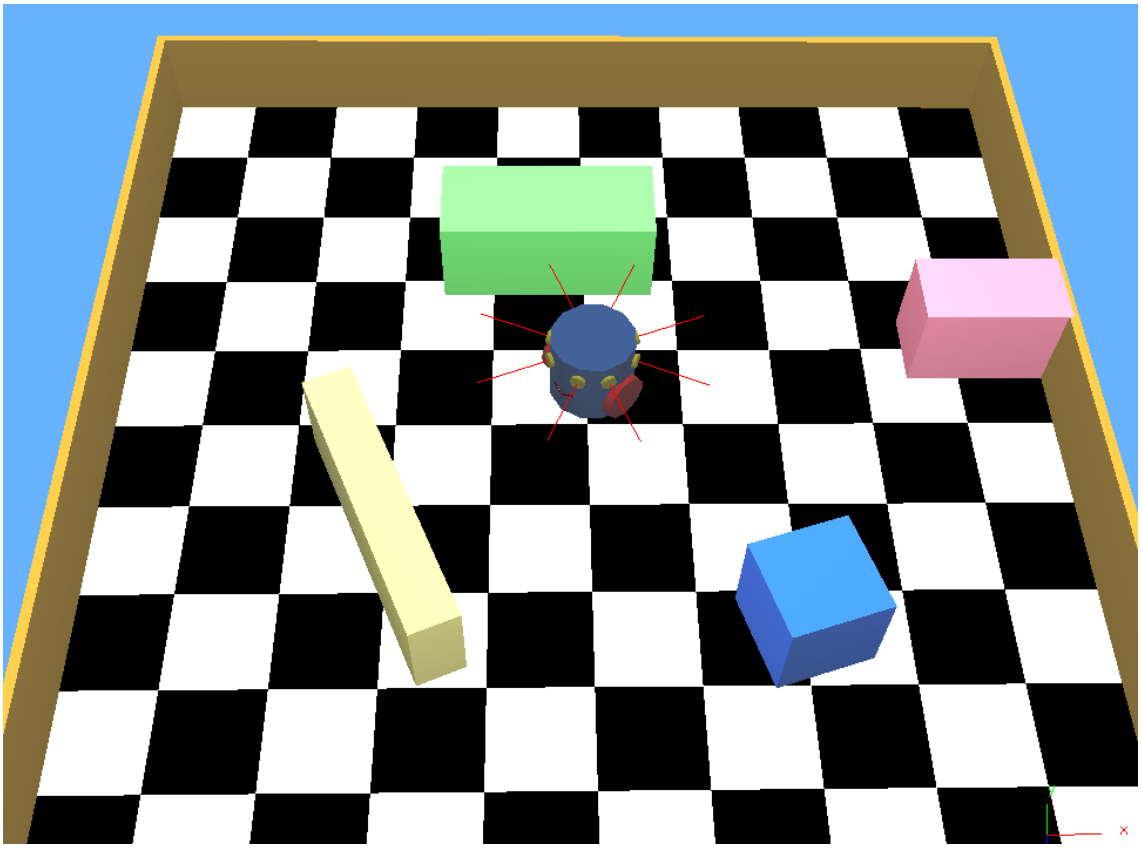
Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Licenciatura em Engenharia Informática

Relatório Nº 1
Introdução à Inteligência Artificial, 2º Semestre

Veículos de Braitenberg

Sense it…



Trabalho realizado por:

- **Ivo Daniel Venhuizen Correia, nº2008110814** **(icorreia@student.dei.uc.pt)**
- **João Pedro Gaioso Barbosa, nº2008111830** **(jpbarb@student.dei.uc.pt)**
- **Alunos da TP3 e PL1**

Coimbra, 14 de Março de 2011

Índice

Uma Breve Introdução

Tratando-se esta da primeira etapa de um trabalho que nos irá ocupar durante algum tempo, tivemos como um dos principais objectivos nesta fase inicial o cuidado de escrever um código genérico e facilmente modificável no que toca a parâmetros, de modo a que as nossas perspectivas actuais possam ser expandidas e rapidamente modificadas se tal se tornar evidente e necessário. Deste modo, privilegiámos a reutilização de código e o uso de herança entre classes para a criação dos diversos sensores entre outros aspectos do código.

Este relatório foi dividido em três partes fundamentais, nomeadamente a discussão do código e alterações que já foram feitas, a esforço exigido a cada membro do grupo e por fim, as perspectivas futuras, levantando um pouco ao véu sobre o que esperamos do trabalho final.

A Nossa Implementação

Modificações ao Código Original

Tal como foi dito na introdução, procurámos redigir código o mais genérico possível, como objectivo de obtermos ficheiros limpos, entendíveis e facilmente modificáveis.

Assim, tínhamos como opção, para a criação dos diversos sensores, copiar simplesmente a classe já implementada para um sensor de luz e apenas modificar a designação e coloração de cada um. Contudo, como é de notar, todos os sensores partilham muita da informação e características. Por isso, definimos uma classe *BraitenbergMainSensor*, que basicamente, alberga todos os métodos do sensor de luz originalmente proposto, exceptuado o método *iterate()*.

Este método é integrado em duas outras classes, *BraitenbergSensor* e *BraitenbergBlockSensor*, classes que derivam de *BraitenbergMainSensor*. *BraitenbergSensor* fica responsável por detectar os objectos de luz, olfacto e som, sendo que para realizar esta distinção recorremos a um método *setType()*. A detecção e reacção perante os emissores são exactamente iguais ao comportamento definido para a luz.

Quanto aos blocos, tivemos de modificar um pouco mais o código, e por isso não podia ser simplesmente identificado como uma variante do sensor original. Como o veículo apenas responde ao bloco que se encontrar mais próximo, iteramos normalmente sobre todos os blocos no campo de visão, mas apenas consideramos o que estiver mais próximo, sendo que esse trabalho é feito pelo registo da menor distância e da força correspondente.

Cenários de Teste

De modo a testar o nosso código, planeámos alguns cenários de teste de modo a garantir que não estamos a cometer nenhum erro. No primeiro, com uma ligação cruzada, definimos uma linha de emissores de som e estando o veículo inicialmente distante desta linha, ganha velocidade e uma vez atingido o primeiro sensor, segue o trilho (figura 1).

Por outro lado, de modo a testar um veículo que se afasta dos emissores, criámos um circuito circular, definido por dois círculos de diferentes raios. O veículo, com posição inicial entre os dois círculos, segue o percurso, sem nunca atravessar os limites impostos (figura 2).

Para concluir, incluímos um cenário com apenas dois blocos. Inicialmente, o veículo avança até se aproximar do bloco direito. Aí vira, para a esquerda e apenas nesse momento passa a considerar o bloco do lado esquerdo, mudando a sua trajectória uma vez mais. Assim,

podemos comprovar que os sensores de proximidade estão a trabalhar adequadamente (figuras 3). Notar que tivemos de baixar a voltagem de 5 para 2 de modo a dar tempo ao veículo de mudar a trajectória quando se dirige para o bloco do lado esquerdo.

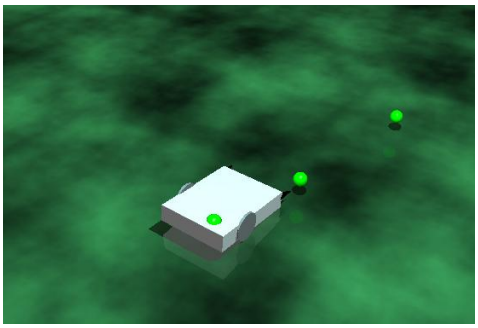


Figura 1. O veículo segue o trilho de som.

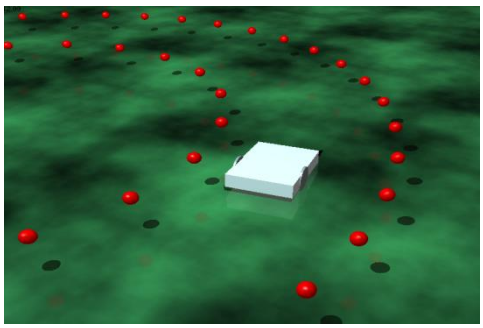


Figura 2. O veículo mantém-se dentro do percurso.

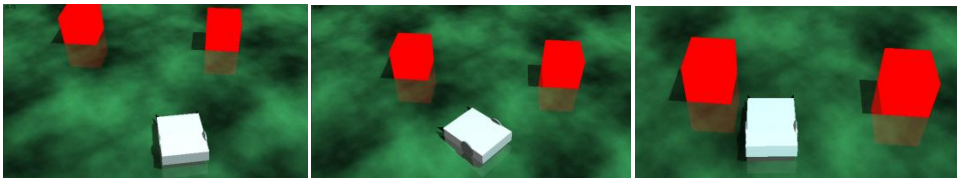


Figura 3. Sequência de imagens do comportamento do veículo perante dois blocos.

Contributos Individuais

Dividimos o trabalho o mais equitativamente possível para que aumentássemos a eficiência e eficácia do nosso trabalho, não só na quantidade e qualidade do trabalho produzido, assim como na mais rápida detecção e correcção de erros e ideias de implementação.

Deste modo, o aluno Ivo Correia ficou encarregue de criar classes genéricas para comportar todos os sensores genéricos e emissores (luz, som, olfacto e blocos) exigidos para o trabalho prático. A redacção final deste relatório também lhe coube em grande parte, sendo que isso não tenha significado que as ideais e o formato do documento tenham sido da sua inteira responsabilidade.

Por seu lado, o aluno João Barbosa teve como missão criar diversos cenários de teste assim como a criação e *setup* do veículo de *Braintenberg* propriamente dito (i.e., colocação das rodas, afinação dos parâmetros de voltagem e reacção e definição dos sensores). Para além disso, tratou da implementação dos sensores de proximidade de blocos.

Sensivelmente, cada aluno gastou 8 horas do seu tempo nesta fase inicial, contando, para além de todo o trabalho necessário para alcançarmos este ponto de situação, o período de habituação ao ambiente *Breve*.

Perspectivas Futuras

Naturalmente, nesta fase, já começámos a formar algumas ideias sobre no que irá consistir o nosso trabalho final. Temos para já duas opções em cima da mesa (podendo aparecer mais se tal se proporcionar), mais concretamente a simulação de uma batalha de

robots recorrendo aos veículos de *Braitenberg*, ou, em alternativa, a geração de labirintos que simulem o famoso jogo *Pacman*. De qualquer uma das maneiras, ambas as ideias envolvem a definição de corredores e pontos de passagem mais estreitos, e daí alguns casos de teste considerados.

Bibliografia

1. *Software Breve*
2. Documentação disponível em '*http://www.spiderland.org*'
3. Imagens:

   Da capa:

   a. http://www.cyberbotics.com

   Outras:

   b. *Screenshots* do ambiente *Breve* em execução para os cenários de teste.