Universidade de Coimbra

Faculdade de Ciências e Tecnologia

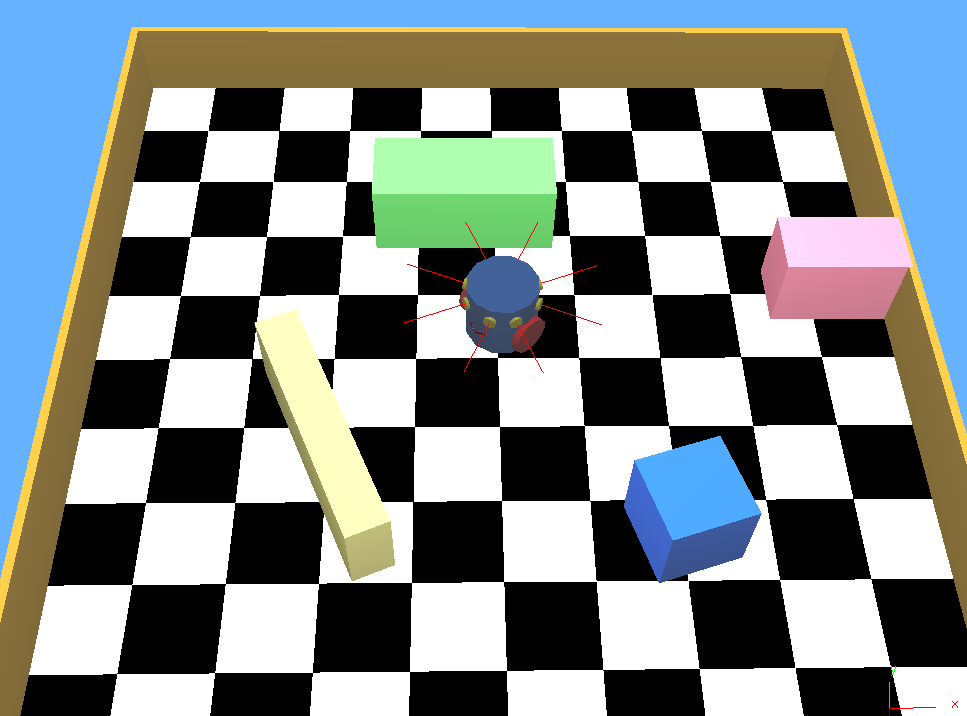
Licenciatura em Engenharia Informática

*Relatório Nº 3*

*Introdução à Inteligência Artificial, 1º Semestre*

**Veículos de Braitenberg**

**…Take it for a Ride!**



**Trabalho realizado por:**

* **Ivo Daniel Venhuizen Correia, nº2008110814**
* **João Pedro Gaioso Barbosa, nº2008111830**

Coimbra, 8 de Abril de 2011

Índice

[Uma Não Tão Breve Introdução 2](#_Toc289814363)

[Sense It – Primeira Meta 4](#_Toc289814364)

[Modificações ao código original 4](#_Toc289814365)

[Cenários de Teste 4](#_Toc289814366)

[Contributos Individuais 5](#_Toc289814367)

[… Tune it & Test it … – Segunda Meta 6](#_Toc289814368)

[As funções de activação 6](#_Toc289814369)

[Os quatro veículos 6](#_Toc289814370)

[Elipser e EightMaker 6](#_Toc289814371)

[3c 6](#_Toc289814372)

[Explorer 7](#_Toc289814373)

[Cenários de Teste 7](#_Toc289814374)

[Contributos Individuais 8](#_Toc289814375)

[… Take it for a Ride! – Terceira Meta 9](#_Toc289814376)

[A nossa ideia: “Pacman” 9](#_Toc289814377)

[O labirinto 10](#_Toc289814378)

[Contributos Individuais 10](#_Toc289814379)

[Conclusão 12](#_Toc289814380)

[Bibliografia 13](#_Toc289814381)

# Uma Não Tão Breve Introdução

Depois de mais de um mês de trabalho, chegámos finalmente à última etapa deste primeiro trabalho prático da cadeira de Introdução à Inteligência Artificial. Fomos desenvolvendo veículos cada vez mais sofisticados e com melhores respostas às modificações ambientais, apresentando muitas vezes comportamentos curiosos e muito intrigantes.

O objectivo deste projecto foi pegar na livraria do *software* Breve, que fornece uma base muito simples para manipulação e teste dos veículos de Braitenberg, e aumentá-la para disponibilizar mais funcionalidades para além das estruturas mais básicas. Assim, começámos por criar novos ‘emissores de sentidos’, sensores para percepcionar a intensidade desses mesmo sentidos e libertar o nosso veículo neste novo ambiente. Este trabalho correspondeu à primeira meta, tendo sido rápida e satisfatoriamente completada.

Com um mundo expandido cheio de oportunidades, tivemos de melhorar a capacidade de resposta e variedade das mesmas dos veículos, implementando novas funções de activação. Isto porque a função lineares, as únicas que tinham sido implementadas de antemão, não servem para todos os casos, sendo que por vezes, tal como acontece no labirinto do nosso ‘Braitenberg Pacman’, queremos que os veículos apenas reajam quando estão a uma determinada distância dos obstáculos ou fontes. Tal comportamento pode ser descrito através de uma função guassiana, justamente uma das funções de activação impostas pela segunda meta. Em outras situações, pretendemos que a aceleração seja maior quanto mais distante os veículos de encontrem do seu objectivo, como é feito pelo ‘Explorador’.

Foi neste contexto que também implementámos os veículos que descrevem elipses e ‘oitos’ em volta de duas fontes, para além do denominado veículo ‘3C’.

Com todos estes elementos juntos, implementados e testados, partimos à aventura para a recta final do nosso projecto. Tal como já foi referido, optámos por uma simulação de um labirinto onde um veículo, representando o ‘Pacman’, tem de fugir de outros dois, os maldosos fantasmas que o perseguem sem tréguas. Durante o seu percurso, o ‘Pacman’ não é simplesmente afastado dos outros dois, mas também é atraído por pequenas fontes que representam o seu alimento.

Este relatório vai seguir um molde um pouco diferente dos anteriores, sendo que o vamos dividir em três grandes secções, cada um correspondente às metas definidas pelos docentes. Deste modo, iremos aproveitar algum do texto redigido nos dois relatórios anteriores, sendo por isso que as duas primeiras partes podem ser vistas quase como versões redefinidas e aumentadas dos trabalhos anteriores, reservando a última para a descrição integral da nossa ideia.

O aumento da documentação relativa às metas anteriores não se prende somente com o facto de este ser um relatório final e por isso, ter de detalhar mais pormenorizadamente cada passo da implementação. Para além de termos um limite mais abrangente no que toca à quantidade de texto, também fomos encontrando diversos erros e pressupostos erróneos depois das respectivas entregas. Como tal, achámos por bem apontar esses aspectos ao longo do relatório, referindo porque estavam errados e como procedemos para a sua correcção. Não o fazemos fazer explicitamente em capítulos específicos, mas vamos incorporar no texto corrido.

Em cada parte, iremos disponibilizar um pequeno texto introdutório para situar o leitor nos parâmetros e tarefas que nos foram exigidas para cada meta, apresentando depois as nossas implementações para atingir tais objectivos, problemas e soluções que encontrámos para cada dificuldade, assim como a definição dos tempos dispendidos por cada elemento do grupo no seu contributo para a conclusão da respectiva meta.

A terceira parte engloba ainda uma visão geral sobre trabalho, assim como uma conclusão acerca dos conhecimentos adquiridos com o desenvolvimento deste projecto deveras desafiante. Esperamos que tenha uma leitura agradável e que todos os aspectos discutidos neste relatório sejam suficientemente explícitos e claros.

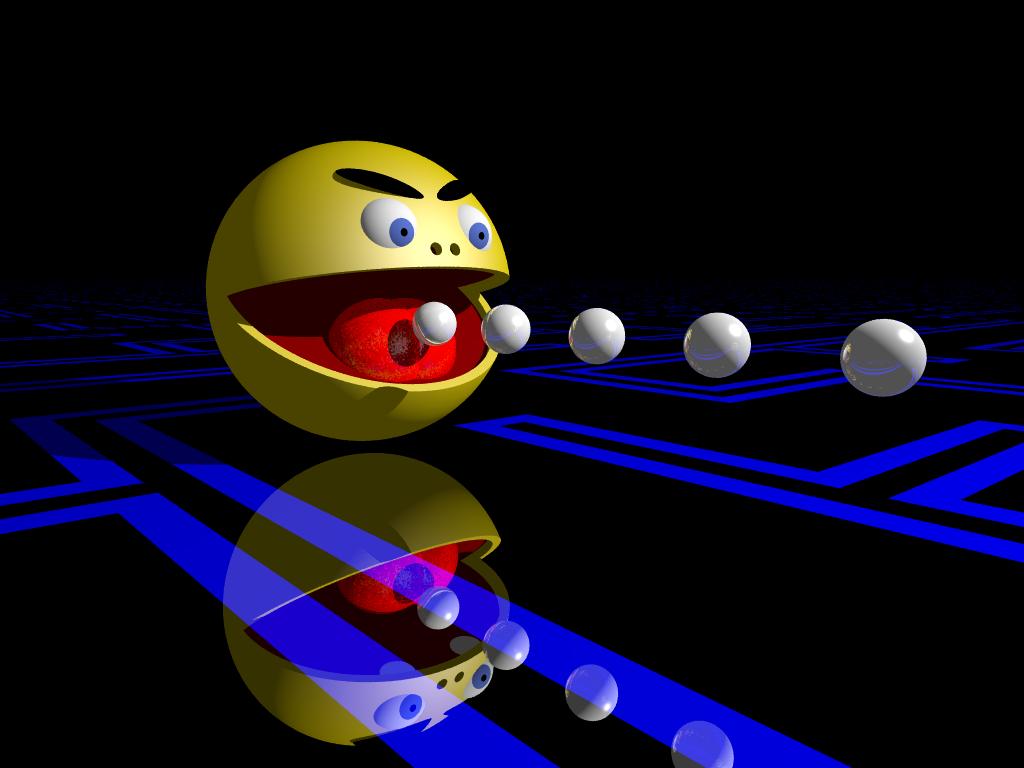


Figura 1. “Pacman”.

# Sense It – Primeira Meta

*O nosso ponto de partida foi a extensão dos objectos e aptidões fornecidos pelo Breve. Tendo sido um trabalho de interpretação e adaptação aos nossos requisitos, não encontrámos dificuldades relevantes. Talvez o maior problema acabou por ser mesmo o ambiente de desenvolvimento disponibilizado pelo Breve, o qual, verdade seja dita, não é dos melhores nem dos mais apelativos…*

## Os Sensores

Desde logo procurámos redigir código o mais genérico possível, como objectivo de obtermos ficheiros limpos, entendíveis e facilmente modificáveis. Tal foi feito com a criação de inúmeras classes e estabelecimento de relações de herança entre elas.

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.

## Ângulo de visão, intensidades e índices de reflexão

Para além de trabalho de volta da interacção entre sensores e fontes de sentidos, para esta primeira meta também nos foi pedido que trabalhássemos com a introdução de métodos para controlar ângulos de visão, intensidades de algumas das fontes (i.e., fontes de luz, som e cheiro) e índices de reflexão dos blocos. Em relação ao trabalho apresentado no relatório da primeira meta, esta foi uma das partes que mereceu uma revisão mais acentuada numa fase posterior.

Em relação aos ângulos de visão, o que simplesmente fazíamos era variar o vector passado à *RevoluteJoint*, sendo que tal procedimento resultava que, passado algum tempo e quando os sensores deixavam de percepcionar fontes, estes começavam a girar descontroladamente. Tal deve-se ao facto da *RevoluteJoint* usar um eixo de rotação e nós não actuarmos directamente sobre esse eixo. Para resolver o problema, mudámos para *FixedJoint*.

Contudo, o ângulo de visão mantinha-se inalterado, porque ao permitir que apontassem para posições mais laterais, não aumentávamos o ângulo de visão, mas sim dirigíamos para uma direcção diferente esse mesmo ângulo. Esta falha apenas foi corrigida quando começámos a precisar de maiores ângulos de visão para a segunda e terceira meta, onde era pedido a criação de veículos que descrevessem percursos em torno de fontes e, no nosso caso, seguissem um trilho dentro de um labirinto. Já tendo a variável de ângulo criada de origem no *Breve*, o nosso único trabalho foi implementar um método de modificação da mesma variável.

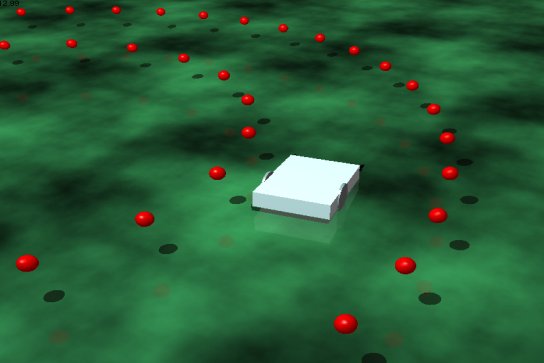
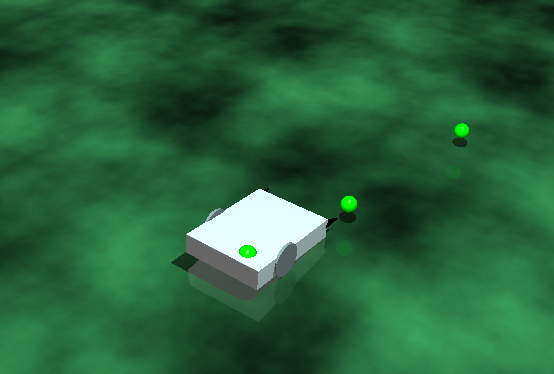
Por fim, temos a intensidade das fontes e os índices de reflexão dos blocos. Estas variáveis são utilizadas de modo a diferenciar fontes (ou blocos) do mesmo tipo, mas que por uma ou outra causa, têm de provocar diferentes estímulos aos veículos de Braitenberg. Deste modo, as intensidades (e para o índice de reflexão, na prática, acaba por ser uma intensidade com um nome distinto) são variáveis que são utilizadas no método *iterate()* do sensores. Como estão a ser multiplicadas pela força, temos que uma intensidade igual a um vai ser a intensidade padrão, e consequentemente, valores inferiores a um (e maiores que zero, deixando de lado os negativos, valores que nunca são por nós utilizados nestas condições) diminuem a intensidade de uma fonte, sendo que valores superiores tornam essa fonte dominante em relação às outras.

## Cenários de Teste

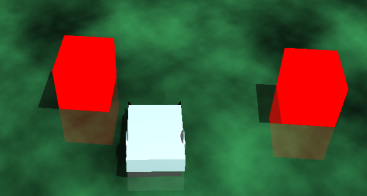
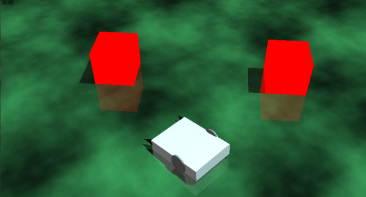
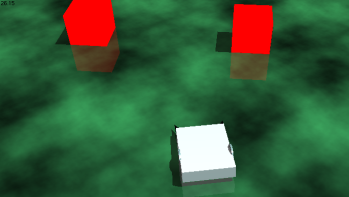
De modo a testar o nosso código, planeámos alguns cenários de teste de modo a garantir que não estamos a cometer nenhum erro. No primeiro, com uma ligação cruzada, definimos uma linha de emissores de som e estando o veículo inicialmente distante desta linha, ganha velocidade e uma vez atingido o primeiro 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 2. O veículo segue o trilho de som. Figura 3. O veículo mantém-se dentro do percurso.**



**Figura 4. 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*.

# … Tune it & Test it … – Segunda Meta

*Ao contrário da primeira parte do trabalho, esta meta ofereceu-nos muitos mais desafios e acesas, saudáveis discussões sobre as metodologias a seguir na abordagem à resolução dos problemas que iam aparecendo. Se por um lado conseguimos definir prontamente quais as funções de activação a utilizar para obter os efeitos desejados, o principal travão ao desenrolar dos acontecimentos de um modo suave e sem sobressaltos foi a afinação dos parâmetros de entrada e saída das funções de activação, assim como localização das fontes e o posicionamento inicial do veículo.*

*Em relação ao trabalho entregue para a segunda meta, tivemos de efectuar algumas alterações, mas concretamente no veículo “3c” e o “Explorador”. O principal aspecto foi a introdução de uma forma diferente de contacto entre o chão e as rodas. Em vez de detectar directamente o contacto entre roda (que na verdade, não é um círculo, mas sim um polígono com muitos lados, o que se traduz em muitos pontos de contacto), envolvemos essa roda numa superfícies invisível esférica (através do método setCollisionShape() ), o que nos permitiu ter apenas um único ponto de contacto, o que aumenta quer a velocidade quer a estabilidade do veículo. Ainda em relação ao “Explorador”, este não se comportava tal e qual a descrição feita por Braitenberg. Tal correcção foi feita mudando as funções de activação.*

## As Funções de Activação

De modo a estender o nosso leque de possibilidades, foi-nos sugerido a implementação de duas novas funções de activação para além da linear, nomeadamente uma função logarítmica e gaussiana. Para além destas, criámos também a função exponencial, sendo que no final ficámos com quatro escolhas diferentes para as funções de activação dos sensores.

De modo a facilitar a manipulação do código, implementámos uma nova classe, chamada de *BraitenbergActivationObject*, que contém atributos relativos ao tipo de função de activação, os limites laterais de activação (*threshold*), assim como o máximo e mínimo admissível da resposta (limites superiores e inferiores). Uma vez definidos estes parâmetros, associamos um objecto desta classe a cada sensor. Esta classe contem ainda um método denominado *activate()* que é invocado, por sua vez, pelo método *iterate()* dos vários sensores. O que este método de activação simplesmente faz é receber um valor de força vindo do *iterate()*, aplicar uma dada função de activação (anteriormente definida aquando da criação do objecto), verificar os limites impostos pelo programador e devolver o valor de saída de acordo com a entrada disponibilizada.

## Os Quatro Veículos

Estes quatro veículos representam cada um dos requerimentos feitos para a segunda meta do trabalho, de modo a testar as várias funções de activação. Tal como já foi sendo referido ao de leve ao longo do texto, procurámos ao máximo reduzir código repetido e redundante. Deste modo, todos estes veículos (tal como o “Pacman” e os seus fantasmas, descritos na secção a seguir) descendem da mesma classe, classe essa que define a base do veículo, para depois ser moldada para cada objectivo nas diversas classes filhas. Assim, sem mais demoras, iremos apresentar os quatros veículos da segunda meta: “Elipser”, “EightMaker”, “3c” e “Explorer”.

### Elipser e EightMaker

O “Elipser” e o “EightMaker” são veículos que foram construídos para descrever formas específicas e repetidas ao longo do tempo. Tal como o nome indica, o “Elipser” descreve uma elipse interior em torno de dois focos, enquanto o “EightMaker” desenha oitos em torno de também duas fontes (idealmente, duas circunferências tangentes).

O “Elipser” foi construído com dois sensores ligeiramente virados para fora, apontando na direcção frontal. Como função de activação, usámos uma gaussiana e uma voltagem positiva. Com uma ligação cruzada e aumento do raio das rodas (de modo a permitir curvas mais acentuadas ao veículo), o “Elipser” produz elipses sem nunca se afastar do trilho definido.

Por seu lado, o “EightMaker” também consegue produzir oitos continuamente sem se afastar da rota. Os sensores foram posicionados de uma forma diferente (colocados à frente, mas perpendicularmente às faces laterais do veículo), sendo que contudo continuam a usar uma função gaussiana e uma voltagem positiva.

Ambos os veículos apresentaram a mesma grande dificuldade, que foi a definição dos parâmetros de entrada da função gaussiana, assim como posicionamento dos sensores, o seu ângulo de visão e delimitação das funções com recurso a limites laterais, inferiores e superiores. Por último, a posição inicial do veículo relativamente à luzes também influencia significativamente o sucesso ou insucesso do objectivo destes dois veículos.

### 3c

O veículo “3c” foi uma fiel reprodução do que era descrito no enunciado do projecto prático. Munido de sensores correctamente definidos, o nosso veículo “3c” foi solto num mundo delimitado por luzes (uma vez que para este tipo de veículo, temos que ele se afasta das luzes com uma voltagem negativa, garantindo deste modo que ele não abandone este mundo quadrangular), dentro do qual se encontravam blocos, cheiros e sons. Tal como esperado, o veículo procura o contacto com os sons, evitando chocar contra os blocos e afastando (se bem que desta vez continuando a sua marcha frontal) dos cheiros.

### Explorer

O veículo “Explorer” (ou o “Explorador”, em português), partilha algumas das características do veículo anterior. Também evita o contacto com os blocos, só que, devido à sua definição, aproxima-se de todos os outros objectos.

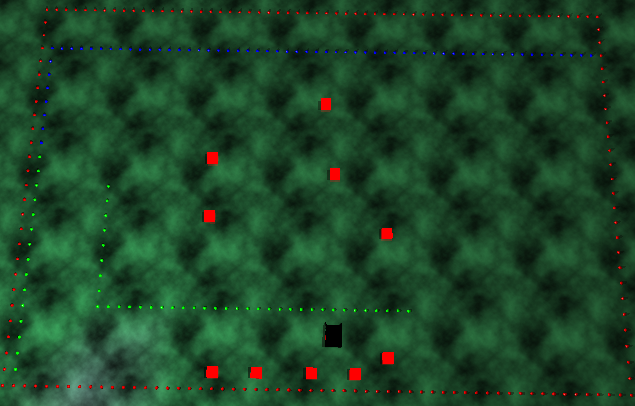
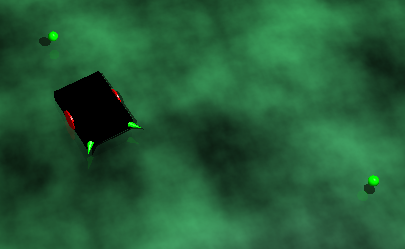
Sendo um explorador, optámos por largá-lo num mundo infinito (ou melhor dizendo, no mundo definido por omissão no *Breve*, o qual ainda apresenta alguma extensão) em que fomos colocando os obstáculo e as fontes atractivas, das quais o veículo se aproxima com mais rapidez, abrandando para à medida que encurta a sua distância para as mesmas. De modo a alcançar tal comportamento, recorremos a uma função gaussiana centrada na origem.

## Cenários de Teste

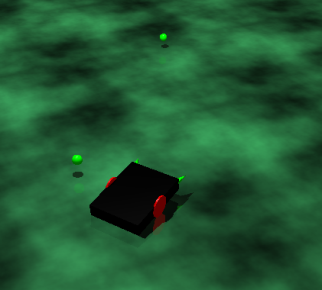
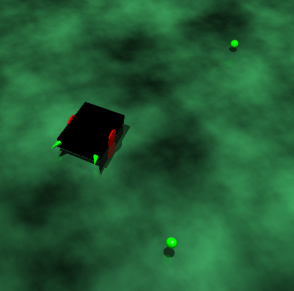
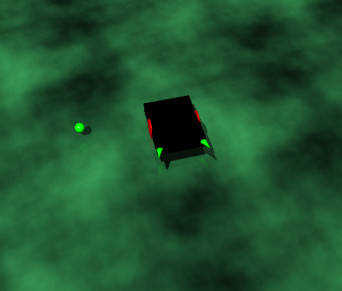
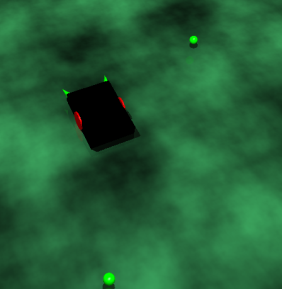
Como cenários de teste para esta meta, começámos pelo cenário onde era exigido que o veículo desenhasse uma elipse em torno de dois focos (figura 2). Chegámos a resultados favoráveis de uma forma relativamente rápida, sendo que o veículo descreve elipses sem nunca se desviar da trajectória inicial.

Quanto ao segundo cenário (figura 3), onde o veículo deveria delinear um 8 em volta de dois focos, tivemos muitas mais dificuldades. Precisámos de muitas afinações e redefinições das posições iniciais dos blocos, do próprio veículo e parâmetros das funções de activação, mas foi com satisfação que conseguimos colocar o veículo em órbita.

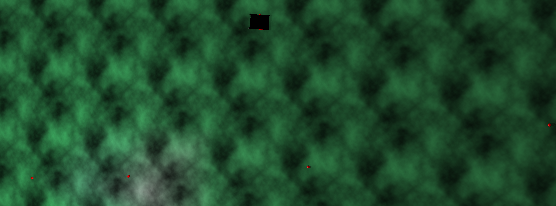
Para finalizar, definimos um cenário onde o nosso veículo ‘3c’ pudesse demonstrar as suas capacidades. Usamos todos os tipos de fontes (luz: vermelho, som: verde, olfacto: azul) e alguns blocos. De modo a facilitar a geração destes cenários, fizemos um pequeno pedaço de código que lê de um ficheiro de texto uma configuração em ASCII e a traduz para o efeito desejado. Para além disso, temos ainda um cenário para o veículo explorador, que procura sempre as luzes, sons ou cheiros mais distantes, procurando desviar-se dos blocos que vai encontrando no seu caminho.

**Figura 5. Um cenário de exploração para o veículo 3c. Figura 6. O veículo realiza uma elipse.**



**Figura 7. Sequência de imagens do ‘circuito octal’ feito pelo veículo.**



**Figura 8. O veículo explorador procura as luzes mais distantes, voltando para trás se nada encontrar à sua frente.**

## Contributos Individuais

Para esta meta, torna-se mais difícil distinguir o trabalho feito por cada elemento do grupo, pois ambos contribuíram para todas as fases exigidas. Contudo, pode-se dizer que o aluno Ivo Correia contribuiu com maior relevância para a redacção do relatório e definição dos veículos Explorador e 3c (assim como os respectivos cenários de teste); enquanto o João Barbosa participou de forma mais activa na definição da classe de funções de activação e ajustamento dos parâmetros para que o veículo padrão estabelecesse uma elipse e um ‘circuito octal’ em torna de dois emissores.

Foram necessárias mais horas de trabalho para esta meta, especialmente porque foi gasto muito tempo na afinação dos parâmetros para obter os comportamentos desejados para cada veículo. Consequentemente, foram gastas aproximadamente 15 horas por cada aluno.

# … Take it for a Ride! – Terceira Meta

*Tendo liberdade total para o desenvolvimento e planeamento da última fase do projecto, foi desta meta que retirámos mais gozo e diversão. Tivemos duas ideias dominantes, uma ‘Arena de Robots’ e um ‘Pacman’, sendo que foi a última que acabou por prevalecer.*

*Tal como na meta anterior, não foi um trabalho fácil, especialmente no que toca ao ajuste dos parâmetros relativos à função de activação. Para além disso, também tivemos de jogar com o alcance e ângulo de visão dos sensores, porque num labirinto de ‘Pacman’, temos ângulos rectos e estando o caminho limitado com um barreira, nas primeiras tentativas, o nosso ‘Braitenberg Pacman’ chocava contra os blocos e bloqueava.*

*Depois disso, ainda tivemos de lidar com a oposição dos fantasmas, uma vez que o ‘Pacman’ tinha de conseguir detectá-los e apenas virar quando tal era possível. De modo a auxiliar todos estes labirintos de testes, fizemos um gerador de cenários a partir de um ficheiro com texto ASCII, ferramenta que nos poupou imenso tempo.*

## A Nossa Ideia: “Pacman”

Apesar do ‘Pacman’ ser um jogo extremamente simples, apercebemo-nos que a sua implementação não era assim tão trivial… De modo a facilitar o progresso do trabalho, dividimos a elaboração do jogo em diferentes fases.

No primeiro momento, pensámos em criar apenas um labirinto onde o ‘Pacman’ se desviava dos blocos que delimitavam o caminho. Contudo, esta abordagem provou ser demasiado exigente quando era requerido ao veículo que descrevesse ângulos de 90º. Por isso, introduzimos uma nova componente de auxílio à navegação.

Assim, criámos um trilho de fontes, o que, para além fazer com que o veículo não embatesse nas paredes, dava um novo realismo ao jogo, uma vez que as fontes representam os pontos que têm de ser comidos pelo ‘Pacman’. Deste modo, sempre que o veículo alcança uma destas fontes, torna-a transparente. Evitámos eliminar totalmente a fonte, porque eles têm de continuar a estar presentes, para que os fantasmas possam também seguir o trilho. Mesmo com um trilho, as curvas continuaram a ser um grande problema, de modo que tivemos de dar mais algum espaço e tornar os cantos mais curvos.

Assim que tivemos um veículo a seguir um trilho e manter-se dentro dos limites, não chocando contras as paredes, partimos para a segunda fase da implementação: a introdução dos fantasmas no labirinto. Se até este ponto o ‘Pacman’ era representando como um veículo simples, apenas com um paralelepípedo como corpo, duas rodas e alguns sensores, com a introdução dos fantasmas, tivemos também de alterar a sua anatomia.

Deste modo, modificámos algumas fontes, nomeadamente as de luz e som, aumentando o seu raio, já que elas eram ambas representadas com esferas. Depois, modificámos a classe de herança destas mesmas fontes, de *Mobile* para *Link*, de modo a permitir que as esferas fossem ligadas aos veículos propriamente ditos. Neste momento, tínhamos esferas em cima dos veículos. Como toque final de fidelidade ao jogo original, tornámos todo o veículo transparente, exceptuando as esferas.

Serviram as fontes colocadas nos veículos apenas para a componente visual? Não de todo. Com a introdução de novos sensores, permitimos que os fantasmas detectem o ‘Pacman’ e vice-versa. Tivemos de limitar o raio de alcance dos sensores e o seu ângulo de visão, para que os veículos apenas ajam quando estão suficientemente perto uns dos outros.

Contudo, quase que tivemos de voltar ao início com a introdução dos fantasmas, porque agora, tínhamos uma nova componente de interferência no ambiente do ‘Pacman’. Para a detecção da oposição, recorremos a um sensor com uma função de activação gaussiana. Para esta função, temos um desvio padrão muito reduzido, para que quando o ‘Pacman’ estiver suficientemente aperto, desvia-se abruptamente do fantasma. Contudo, tivemos o cuidado de não introduzir valores demasiado altos, porque de contrário, ou o ‘Pacman’ virava demasiado e chocava contra as paredes, ou então as rodas ficavam sujeitas a demasiada pressão e acabávamos com um ‘Pacman’ voador…

Tendo os fantasmas e o ‘Pacman’ funcionais, entrámos na última fase. Isto é, produzir um labirinto em que o ‘Pacman’ acabasse por ser apanhado, mas sem dar alguma luta aos fantasmas. Para alcançar tal objectivo, tivemos de definir bem os limites iniciais do ‘Pacman’ e dos fantasmas, para que o veículo ainda se desvie uma vez e permaneça em fuga durante algum tempo. Quanto é atingido, a voltagem associada às rodas do veículo ‘Pacman’ são aumentados para valores muito elevados (500, no nosso caso), sendo que para uma das rodas temos um valor positivo e na outra negativo. Deste modo, a tensão sobre as rodas torna-se enorme e o veículo ‘salta’ fora do cenário, representando o fim do jogo.

É de notar que o nosso ‘Pacman’ tem uma desvantagem em relação aos seus opositores, o que o conduz muitas vezes, a um fim prematuro. Trata-se de o facto de o ‘Pacman’ apenas ter sensores dos fantasmas posicionado na parte da frente. Assim, se encontra oposição à sua frente, faz esforços para a evitar. Contudo, se o fantasma se aproximar por trás, o inocente ‘Pacman’ segue o seu caminho sem preocupações de maior. Por outro lado, os sensores dos fantasmas que os possibilitam seguir o ‘Pacman’ traduzem-se num aumento da voltagem de alimentação das rodas. Consequentemente, vão gradualmente ganhando velocidade e facilmente ganham terreno ao ‘Pacman’.

Analisámos todas estas possibilidades e discutimos diferentes soluções, mas no fim, optámos por beneficiar os fantasmas. Em primeiro lugar, poderíamos colocar sensores na retaguarda do veículo ‘Pacman’, que lhe concedia maior velocidade quando os fantasmas se aproximavam. Contudo, caímos no problema inverso, e agora o ‘Pacman’ dificilmente era apanhado. Por outro lado, esta nova habilidade fazia com que o ‘Pacman’ se tornava altamente instável nas curvas, muitas vezes deixando o trilho (e consequentemente, chocando contra os limites) quando tinha um fantasma no seu encalço.

Por outro lado, poderíamos pensar em limitar a voltagem transmitida às rodas dos fantasmas, quando esta era fornecida pelos sensores de perseguição. Implementámos parcialmente esta opção, para alongar o tempo de vida do ‘Pacman’, mas como contrapartida, o preço a apagar foi que por vezes, os fantasmas não perseguem activamente o ‘Pacman’. Se a velocidade que têm no momento foi demasiado elevada e o ‘Pacman’ estiver demasiado afastado, com sensores mais fracos, os fantasmas acabam por ignorar o ‘Pacman’ e seguir o seu caminho. Contudo, para a simulação criada, tentámos que tal comportamento afectasse o mínimo possível o desempenho dos fantasmas.

De qualquer das formas, não é um ponto totalmente errado, porque como já foi revelado, o sistema de inteligência artificial do ‘Pacman’ original não garante que os fantasmas tomem todas as suas decisões baseadas na direcção do ‘Pacman’. Isto, se o ‘Pacman’ virar à esquerda, o fantasma poderá seguir em frente, se tal se relevar mais próximo da sua estratégia para encurralar o ‘Pacman’.

## O Labirinto

O nosso labirinto é relativamente simples, centrando-se a justificação na questão da eficiência do programa. Com um cenário maior e mais complexo, tínhamos de obrigatoriamente introduzir mais fontes e blocos. Como, por exemplo, no trilho ou nos limites, as fontes e os blocos têm de estar relativamente perto uns dos outros, a quantidade de objectos presentes no ambiente de simulação tornava-se demasiado elevado e consequentemente, a simulação ficava demasiado lenta, aborrecida e irrealista.

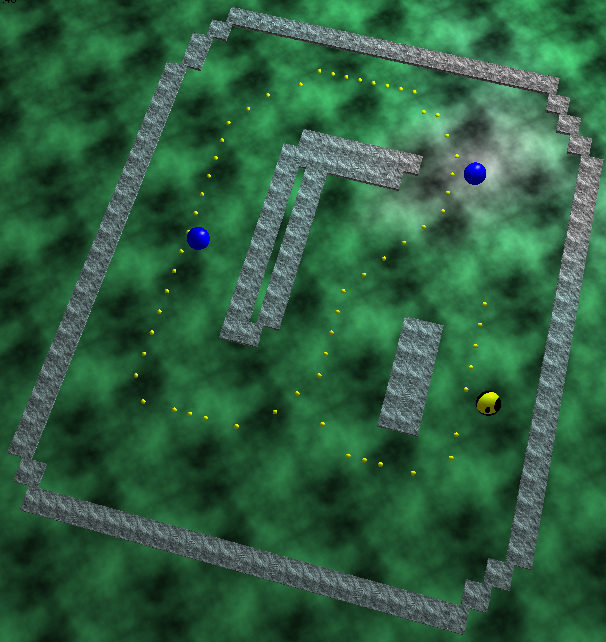


Figura 9. O labirinto do “Pacman”.

Se uma das dificuldades foi mesmo fazer com os veículos descrevessem curvas em ângulos rectos sem chocar contra os limites (tal como se pode ver na fotografia tirada de cima do labirinto, Figura 8, o trilho central ainda está um pouco afastado do muro em redor, especialmente na parte exterior), introduzir mais e mais curvas implicava um cenário exponencialmente maior, pois colocar curvas demasiado próximas umas das outras tinha como resultado os veículos desviarem-se em demasia ao dar uma e já não ter possibilidades de efectuar a segunda em condições.

Para além destes aspectos, também tentámos criar um cenário relativamente pequeno para que o tempo de execução da simulação não se estendesse por demasiado tempo, correndo o risco depois de não a podermos correr na sua totalidade durante a defesa do trabalho.

Com estes ingredientes e limitações, criámos o cenário visível na Figura 8, que pode ser descrito como um quadrado em cujo interior se situam duas ilhas de blocos. Em torno dessas mesmas ilhas, navegam o ‘Pacman’ e os fantasmas, cada um procurando alcançar o seu próprio objectivo.

De notar, tal como já foi referido anteriormente, que este labirinto não foi gerado manualmente. Através de uma notação por nós definida (como por exemplo, ‘\*’ corresponder a um bloco ou ‘l’ a uma luz), descrevemos o cenário num ficheiro de texto e através da leitura desse mesmo ficheiro, o labirinto é gerado automaticamente. Tal ferramenta foi-nos deveras útil para fazer pequenas afinações que, escrevendo directamente, nos iriam consumir muito tempo. Referimo-nos mais concretamente à definição da acentuação das curvas, posicionamento do trilho em relação aos limites, ou distância entre duas curvas consecutivas.

De notar que tal gerador não se restringiu somente a este cenário do ‘Pacman’. Tal como seria de esperar, também o pudemos aplicar a outros cenários de teste (como por exemplo o do ‘Explorador’), o que nos permitiu desenvolver ambientes mais sofisticados num período de tempo mais reduzido.

## Contributos Individuais

Escrever cenas.

# Conclusão

De modo a estender o nosso leque de possibilidades, foi-nos sugerido a implementação de duas

# Bibliografia

1. *Software Breve*
2. Documentação disponível em ‘*http://www.spiderland.org*’
3. Imagens: **Da capa:**
   1. <http://www.cyberbotics.com>

**Outras:**

* 1. *Screenshots* do ambiente *Breve* em execução para os cenários de teste.
  2. *Pacman*, na Introdução:
     1. <http://diboua.webs.com/jogos.htm>