Universidade de Coimbra

Faculdade de Ciências e Tecnologia

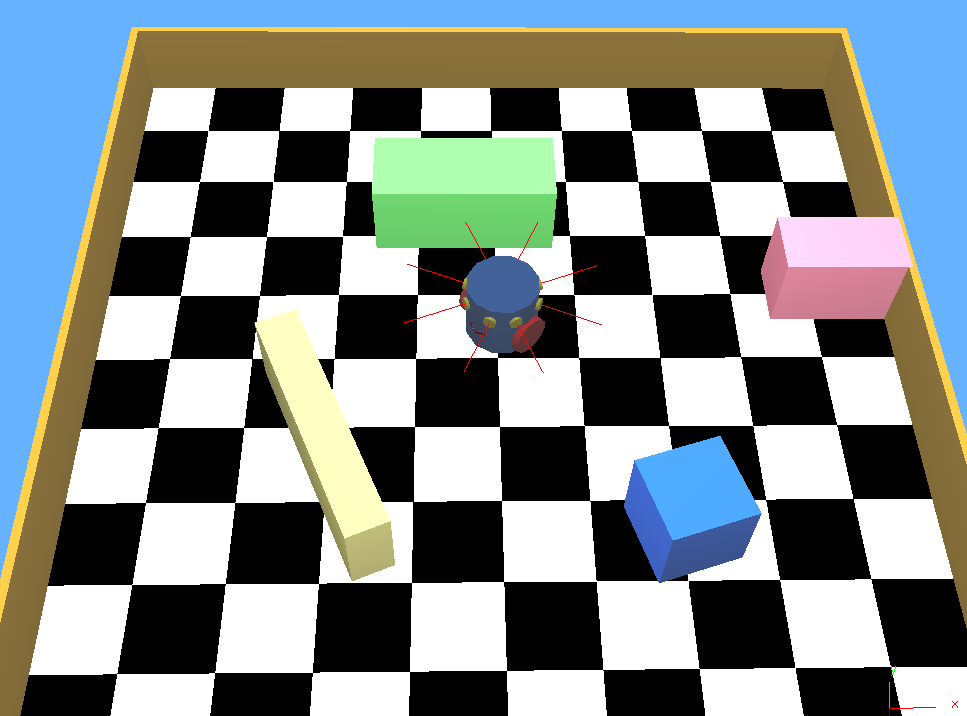
Licenciatura em Engenharia Informática

*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](#_Toc289690686)

[Sense It – Primeira Meta 3](#_Toc289690687)

[Modificações ao código original 3](#_Toc289690688)

[Cenários de Teste 3](#_Toc289690689)

[Contributos Individuais 4](#_Toc289690690)

[… Tune it & Test it … – Segunda Meta 5](#_Toc289690691)

[As funções de activação 5](#_Toc289690692)

[Cenários de Teste 5](#_Toc289690693)

[Contributos Individuais 6](#_Toc289690694)

[… Take it for a Ride! – Terceira Meta 7](#_Toc289690695)

[A nossa ideia: “Pacman” 7](#_Toc289690696)

[O labirinto 7](#_Toc289690697)

[Contributos Individuais 7](#_Toc289690698)

[Conclusão 9](#_Toc289690699)

[Bibliografia 10](#_Toc289690700)

# 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.

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

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.

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.

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.

# 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…*

## Modificações ao código original

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.

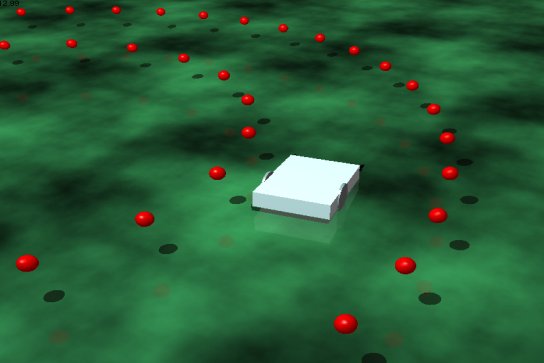
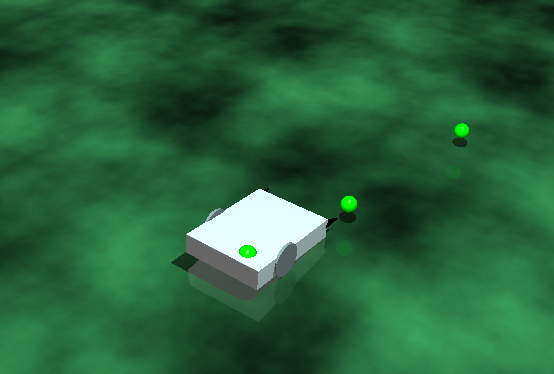
Quanto aos blocos, tivemos de modificar um pouco mais o código, e por isso não podia ser simplesmente identificado como uma variante do sensor original. Como o veículo apenas responde ao bloco que se encontrar mais próximo, iteramos normalmente sobre todos os blocos no campo de visão, mas apenas consideramos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

## Cenários de Teste

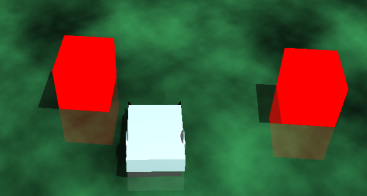
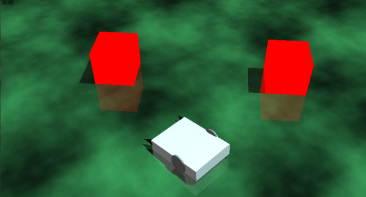
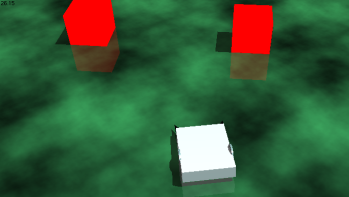
De modo a testar o nosso código, planeámos alguns cenários de teste de modo a garantir que não estamos a cometer nenhum erro. No primeiro, com uma ligação cruzada, definimos uma linha de emissores de som e estando o veículo inicialmente distante desta linha, ganha velocidade e uma vez atingido o primeiro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

# … 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.*

## 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.

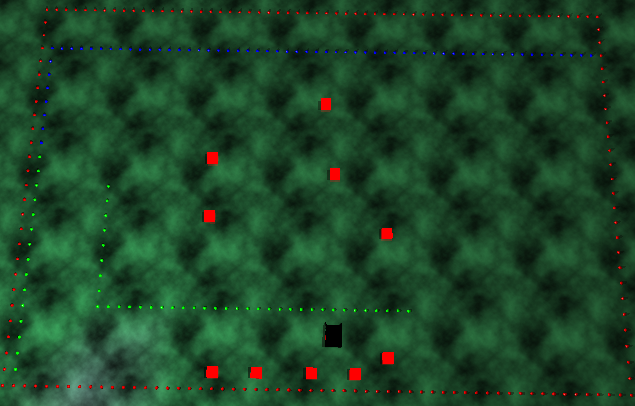
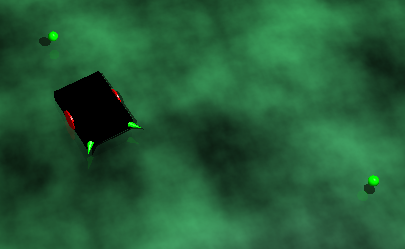
De modo a facilitar a manipulação do código, implementámos uma nova classe, chamada de *BraitenbergActivationObject*, que contém atributos relativos ao tipo de função de activação, os limites laterais de activação (*threshold*), assim como o máximo e mínimo admissível da resposta (limites superiores 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.

## Cenários de Teste

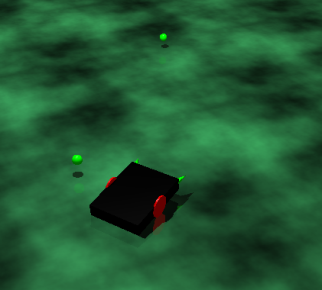
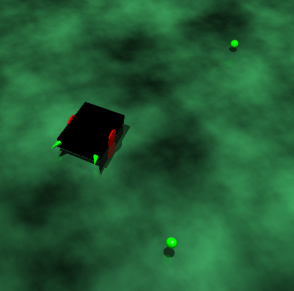
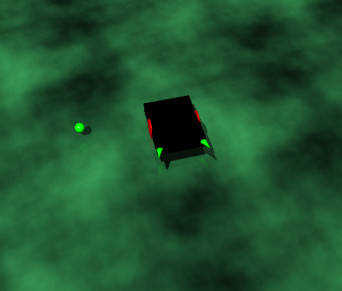
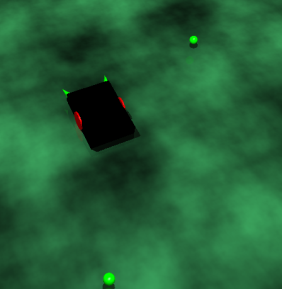
Como cenários de teste para esta meta, começámos pelo cenário onde era exigido que o veículo desenhasse uma elipse em torno de dois focos (figura 2). Chegámos a resultados favoráveis de uma forma relativamente rápida, sendo que o veículo descreve elipses sem nunca se desviar da trajectória inicial.

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.

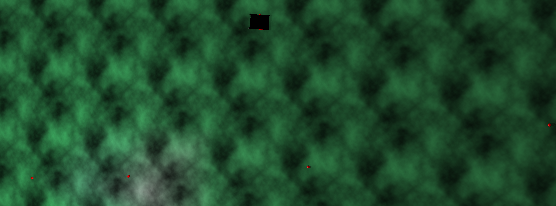
Para finalizar, definimos um cenário onde o nosso veículo ‘3c’ pudesse demonstrar as suas capacidades. Usamos todos os tipos de fontes (luz: vermelho, som: verde, olfacto: azul) e alguns blocos. De modo a facilitar a geração destes cenários, fizemos um pequeno pedaço de código que lê de um ficheiro de 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 mais distantes, voltando para trás caso não tenha mais nenhuma à sua frente.

**Figura 4. Um cenário de exploração para o veículo 3c. Figura 5. O veículo realiza uma elipse.**



**Figura 6. Sequência de imagens do ‘circuito octal’ feito pelo veículo.**



**Figura 7. 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.

## O labirinto

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 mais distantes, voltando para trás caso não tenha mais nenhuma 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.

# 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.