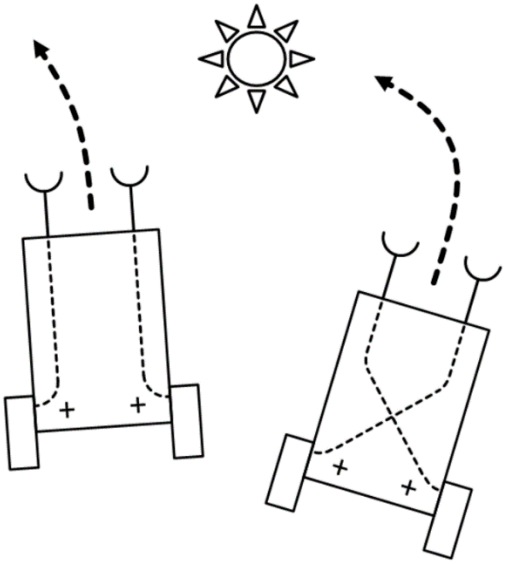
INTELIGÊNCIA ARTIFICIAL

TRABALHO PRÁTICO Nº1

VEÍCULOS DE BRAITENBERG



Nome:
Pedro Manuel Cerveira Andrade

Nº de Estudante:
2014225147

Email:
pmca@student.dei.uc.pt

Turma Prática:
PL1

Nome:
Luis Alberto Pires Amaro

Nº de Estudante:
2014109216

Email:
lapa@student.dei.uc.pt

Turma Prática:
PL2

Nome:
Joel Filipe Rogão Pires

Nº de Estudante:
2014195242

Email:
jfpires@student.dei.uc.pt

Turma Prática:
PL1

Entrega Final

DEI – FCTUC

ÍNDICE

INTRODUÇÃO

Nesta secção pretende-se apresentar sumariamente as experiências que vamos levar a cabo neste trabalho prático.

Trata-se de um trabalho desenvolvido no âmbito da unidade curricular de Introdução à Inteligência Artificial do curso de Engenharia Informática da Universidade de Coimbra e regida por Fernando Machado.

Neste trabalho vamos desenvolver e explorar as capacidades de agentes reativos autónomos. Para isso foram escolhidos Veículos de Braitenberg. Porquê? Porque são robots que podem manifestar comportamentos sofisticados apesar da sua extrema simplicidade. Estes veículos permitem-nos simular diferentes reações de um ser vivo a vários elementos da natureza (como por exemplo luz e obstáculos) através de estímulos sensoriais processados em diferentes lados/hemisférios.

Para desenvolver tal projeto usou-se o motor de jogo Unity e a linguagem de programação C#. O Unity foi-nos útil para poder construir os veículos, construir cenários de teste e visualizar os comportamentos manifestados pelos nossos robots. A linguagem C# deu vida a esses mesmos robots que manifestavam os comportamentos que nós próprios havíamos programado previamente.

IMPLEMENTAÇÃO DE FUNCIONALIDADES

CONSTRUÇÃO DE VEÍCULOS AGRESSIVOS E MEDROSOS À LUZ E AOS BLOCOS

Para a implementação destas funcionalidades nos veículos de Braitenberg tínhamos de ter em conta as seguintes características:

- A saída dos foto-sensores é calculada em função **de todas as fontes de luz dentro do seu ângulo sensorial.**
- Já a saída dos sensores de proximidade deve depender, exclusivamente, do **bloco mais próximo no seu ângulo sensorial.**

❖ SCRIPTS CRIADOS/ALTERADOS:

- ***CarBehaviour.cs –*** este script determina como o nosso veículo se deve movimentar, isto é, que direção ele deve tomar tendo em conta a velocidade que o veículo adquire. Este script já estava previamente implementado pelo que apenas tivemos de declarar as seguintes variáveis:

```
public ObjectDetectorScript LeftOD;
public ObjectDetectorScript RightOD;
```

- ***ObjectDetectorScript.cs –*** criamos este script para programar os nossos sensores de proximidade a blocos. Colocamos 2 sensores de proximidade a obstáculos no nosso veículo: um à direita e outro à esquerda. Tivemos os seguintes cuidados:
  - Todos os nossos blocos possuem a tag "Obstacle" o que nos permite obtê-los no script da seguinte forma:

```
public GameObject[] GetAllObstacles(){
    return GameObject.FindGameObjectsWithTag ("Obstacle");
}
```

  - Na função *Update()*, calculamos constantemente o obstáculo mais próximo e a distância a que ele se encontra:

```
foreach (GameObject obstacle in obstacles) {
    //Debug.DrawLine (transform.position, ClosestObstacle.transform.position, Color.magenta);
    distance = Vector3.Distance (transform.position, obstacle.transform.position);
    if (distance < minDistance) {
        minDistance = distance;
        ClosestObstacle = obstacle;
    }
}
```

  - Calculamos a intensidade do output com base na proximidade a esse obstáculo.

```
strength +=3.0f / ((transform.position - ClosestObstacle.transform.position).sqrMagnitude);
```

- ***ObjectBehaviour.cs –*** este script que herda as propriedades de *carBehaviour.cs* controla o veículo e permite-nos que, antes de começar a cena e recorrendo a dois booleanos, possamos determinar se queremos que o nosso veículo seja agressivo para com a luz, medroso para com os blocos, etc. Quando a flag da agressividade está a "true", então ele será agressivo para com esse estímulo,

caso contrário ele será medroso. Em baixo encontra-se um excerto de código que mostra a série de "if's" encadeados. Mostra também que foi levado em conta que:

- Se queremos um **veículo medroso** a algum elemento, entao os sensores a esse estímulo devem estar ligados directamente aos motores **do mesmo lado**
- Se queremos um **veículo agressivo** a algum elemento, entao os sensores a esse estímulo devem estar ligados directamente aos motores **do lado oposto**.

```
if (LightAgressive && !BlockAgressive) {
    //Calculate target motor values
    m_RightWheelSpeed = leftSensor * MaxSpeed + rightObstacleSensor * MaxSpeed;
    m_LeftWheelSpeed = rightSensor * MaxSpeed + leftObstacleSensor * MaxSpeed;
}
else if (BlockAgressive && !LightAgressive) {
    m_RightWheelSpeed = rightSensor * MaxSpeed + leftObstacleSensor * MaxSpeed;
    m_LeftWheelSpeed = leftSensor * MaxSpeed + rightObstacleSensor * MaxSpeed;
}
else if (LightAgressive && BlockAgressive) {
    m_RightWheelSpeed = leftSensor * MaxSpeed + leftObstacleSensor * MaxSpeed;
    m_LeftWheelSpeed = rightSensor * MaxSpeed + rightObstacleSensor * MaxSpeed;
}
else{
    //Calculate target motor values
    m_RightWheelSpeed = rightSensor * MaxSpeed + rightObstacleSensor * MaxSpeed;
    m_LeftWheelSpeed = leftSensor * MaxSpeed + leftObstacleSensor * MaxSpeed;
}
```

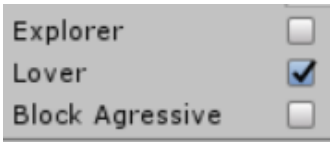
❖ **CENAS ONDE É POSSÍVEL TESTAR ESTAS FUNCIONALIDADES:**

- AgressiveCar
- AgressiveCar 1
- ObjectDetection

CONSTRUÇÃO DE VEÍCULOS APAIXONADOS E EXPLORADORES

❖ **SCRIPTS CRIADOS/ALTERADOS:**

- ***LoveExplorerCars.cs*** **–** este script que herda as propriedades de *carBehaviour.cs* controla o veículo e permite-nos que, antes de começar a cena e recorrendo a três booleanos, possamos determinar se queremos que o nosso veículo seja explorador/apaixonado para com a luz, agressivo/medroso para com os blocos, entre outras combinações. Em baixo encontra-se um excerto de código que mostra a série de "if's" encadeados. Mostra também que foi levado em conta que:



  - Se queremos um **veículo apaixonado** a algum elemento, entao os sensores a esse estímulo devem estar ligados directamente aos motores **do mesmo lado através de inversores.**
  - Se queremos um **veículo agressivo** a algum elemento, entao os sensores a esse estímulo devem estar ligados directamente aos motores **do lado oposto através de inversores.**

```
if (Explorer && !BlockAgressive)
{
    m_RightWheelSpeed = Mathf.Ceil(MaxSpeed / 2) - (leftSensor * MaxSpeed - rightObstacleSensor * MaxSpeed);
    m_LeftWheelSpeed = Mathf.Ceil(MaxSpeed / 2) - (rightSensor * MaxSpeed - leftObstacleSensor * MaxSpeed);
}
else if (Explorer && BlockAgressive)
{
    m_RightWheelSpeed = Mathf.Ceil(MaxSpeed / 2) - (leftSensor * MaxSpeed - leftObstacleSensor * MaxSpeed);
    m_LeftWheelSpeed = Mathf.Ceil(MaxSpeed / 2) - (rightSensor * MaxSpeed - rightObstacleSensor * MaxSpeed);
}
else if (Lover && !BlockAgressive)
{
    m_RightWheelSpeed = Mathf.Ceil(MaxSpeed / 2) - (rightSensor * MaxSpeed - rightObstacleSensor * MaxSpeed);
    m_LeftWheelSpeed = Mathf.Ceil(MaxSpeed / 2) - (leftSensor * MaxSpeed - leftObstacleSensor * MaxSpeed);
}
else if (Lover && BlockAgressive)
{
    m_RightWheelSpeed = Mathf.Ceil(MaxSpeed / 2) - (rightSensor * MaxSpeed - leftObstacleSensor * MaxSpeed);
    m_LeftWheelSpeed = Mathf.Ceil(MaxSpeed / 2) - (leftSensor * MaxSpeed - rightObstacleSensor * MaxSpeed);
}
else
    return;
```

❖ **OBSERVAÇÕES:**

Para criar o veículo que explorasse o meio ambiente sem colidir com os blocos e que fosse agressivo para a luz achamos apropriado usar 4 sensores: 1 sensor de obstáculos e outro de fontes de luz em cada roda traseira do veículo.

❖ **CENAS ONDE É POSSÍVEL TESTAR ESTAS FUNCIONALIDADES:**

- LoverExplorerCars
- LoverExplorerCars 1

CONSTRUÇÃO DE VEÍCULOS COM DIFERENTES FUNÇÕES DE ATIVAÇÃO

Para cada sensor de fontes de luz pode ser especificada:

- A função de ativação desejada (linear ou gaussiana)
- Limiar(threshold) de ativação mínimo e máximo;
- Limite superior e inferior

Lembramos que no caso de a função de ativação dos sensores ser linear, usa-se o seguinte algoritmo (extraído diretamente do enunciado):

1. Calcula a **energia** total recebida pelo sensor:

$$total = \sum_{i=1}^{n} \frac{1}{\frac{distancia(f_i, sensor)^2}{r_i} + 1}$$

onde n é o número de fontes de luz, f_i a posição de cada fonte de luz e r_i o raio (ou alcance) de cada fonte de luz.

2. Divide a **energia** total recebida pelo número de fontes de luz no campo de visão do sensor;

No caso de a função de ativação ser gaussiana, usa-se a seguinte fórmula:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Recorrendo a esta fórmula conseguimos obter comportamentos diferentes se testarmos com valores de media, bias e desvio padrão diferentes.

❖ **SCRIPTS CRIADOS/ALTERADOS:**

- ***LightDetectorMeta2.cs*** **–** a este script, foi adicionada a função *GetGaussianOutput()* onde se a fórmula de gauss com os limiares (que restringem a força) e os limites (que restringem o output) pretendidos. Essa função encontra-se abaixo representada:

```
// Get gaussian output value
public float GetGaussianOutput()
{
    if ( strength > supLimiar ) {
        strength = supLimiar;
    }

    if (strength < infLimiar) {
        strength = infLimiar;
    }

    float a = (float)((1 / (stddev * (Math.Sqrt (2 * Math.PI)))));
    gaussOutput = (float)(a*(Math.Exp (-Math.Pow ((strength - mean), 2)/ (2* Math.Pow (stddev, 2)))));

    // Thresholds
    if (gaussOutput > supLimit) {
        gaussOutput = supLimit;
    }
    if (gaussOutput < infLimit) {
        gaussOutput = infLimit;
    }


    return bias * gaussOutput;
}
```

- ***CarBehaviour.cs*** **–** a este script que serve de base para possíveis comportamentos dos nossos veículos foram adicionados 2 booleanos para que o utilizador possa escolher se pretende que a função de ativação seja gaussiana ou linear.

- ***LoveExplorerCars.cs/CarBehaviou2a.cs/ObjectBehaviou.cs*** **–** estes scripts são usados para determinar comportamentos dos nossos carros relativamente a fontes de luz e, por isso, é nestes que se encontra a decisão se queremos uma ativação linear ou gaussiana. Tal funcionalidade encontra-se retratada a seguir:

```
if (Linear)
{
    leftSensor = LeftLD.GetLinearOutput();
    rightSensor = RightLD.GetLinearOutput();

}
else if (Gaussian)
{
    leftSensor = LeftLD.GetGaussianOutput();
    rightSensor = RightLD.GetGaussianOutput();
}
else
    return;
```

❖ **CENAS ONDE É POSSÍVEL TESTAR ESTAS FUNCIONALIDADES:**

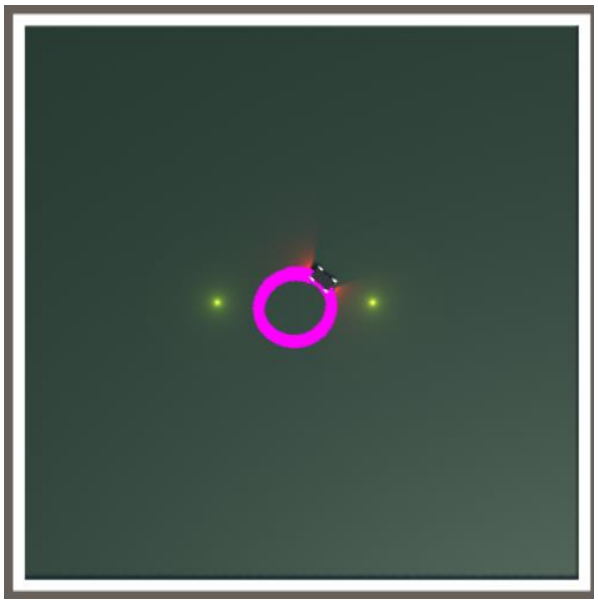
- Em todas incluídas na package.

REPLICAÇÃO DAS TRAJETÓRIAS DO ENUNCIADO

❖ **MOVIMENTO CIRCULAR:**

Foram colocadas 2 fontes de luz à mesma distância e em posições simétricas conforme se pode ver na imagem abaixo. Depois de vários testes com diferentes valores nos diferentes sensores, chegamos ao resultado que queríamos com os seguintes valores retratados na tabela que se segue:

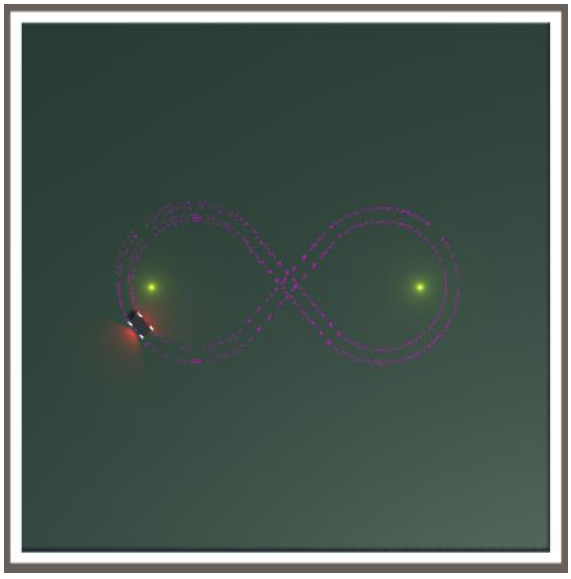
	Sensor da Direita	Sensor da Esquerda
Bias	1	0.3
Angulo do sensor	270	130
μ	0.9	0.9
σ	0.1	0.1
Limite inferior	0.1	0
Limite superior	0.75	1
Limiar inferior	0.1	0
Limiar superior	0.6	0.15



Ainda assim, no o decorrer do tempo verifica-se que o veículo se aproxima mais da fonte de luz do lado direito. Ainda testamos valores diferentes para os limites e limiares de forma a tentar corrigir a situação, mas concluímos que os valores registados são os que se aproximam melhor para uma solução do problema.

❖ **MOVIMENTO EM "8":**

O cenário era idêntico ao anterior, mas desta vez pretendia-se que o veículo seguisse a trajetória de um "8". Mais uma vez testámos com diferentes valores em diferentes sensores e obtivemos sucesso em representar tal trajetória. Os valores usados encontram-se na tabela abaixo:



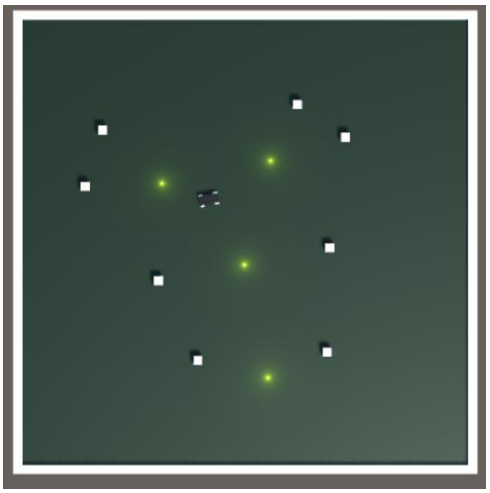
	Sensor da Direita	Sensor da Esquerda
Bias	1	1
Angulo do sensor	115	115
μ	0.9	0.9
σ	0.3	0.3
Limite inferior	0	0
Limite superior	1	1
Limiar inferior	0	0
Limiar superior	1	1

CONCLUSÃO

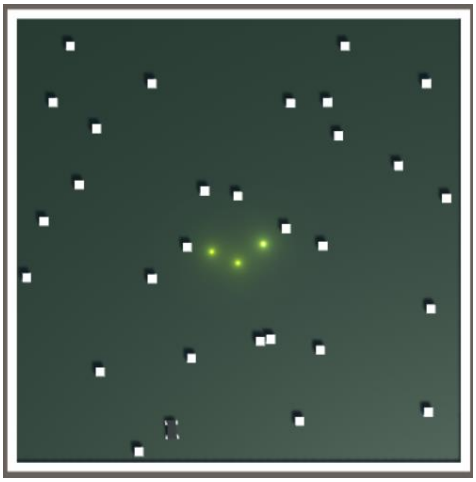
Através deste trabalho pudmos adquirir competências no desenvolvimento de simulações em Unity, nomeadamente ao nível de desenvolvimento e implementação de agentes reativos autónomos. Por fim, foi produtivo porque pudemos familiarizar-nos com o trabalho de Valentino Braitenberg e adquirir conhecimentos nesta área.

ANEXO – A: CENÁRIOS DE TESTE

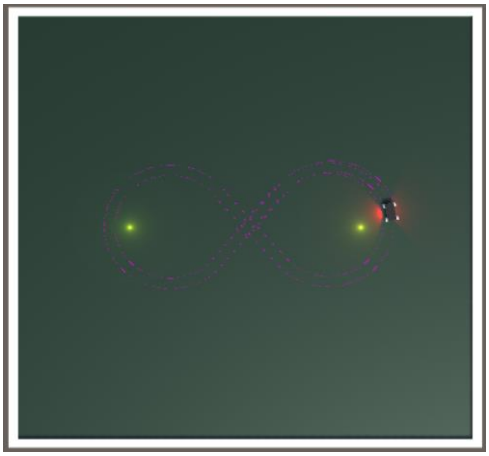
AgressiveCar



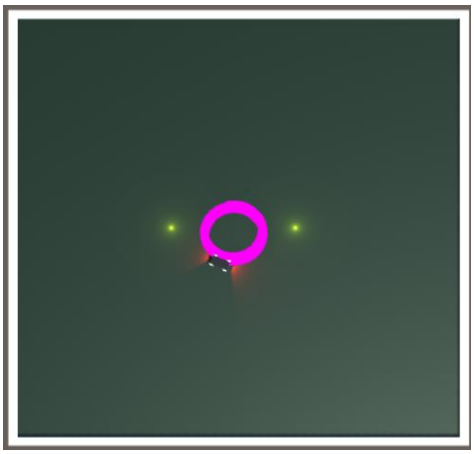
AgressiveCar 1



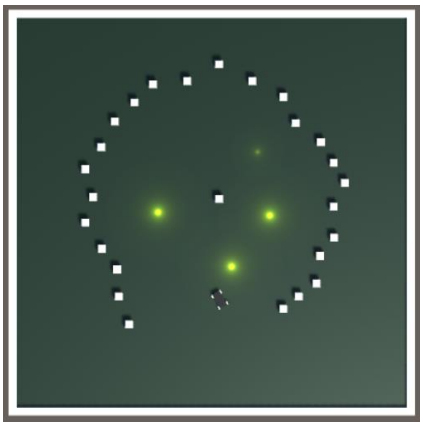
AgressiveCar 8



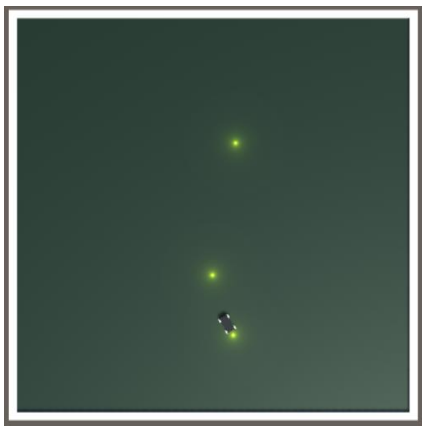
AgressiveCar elipse



LoveExplorerCars



LoveExplorerCars 1



ObjectDetection