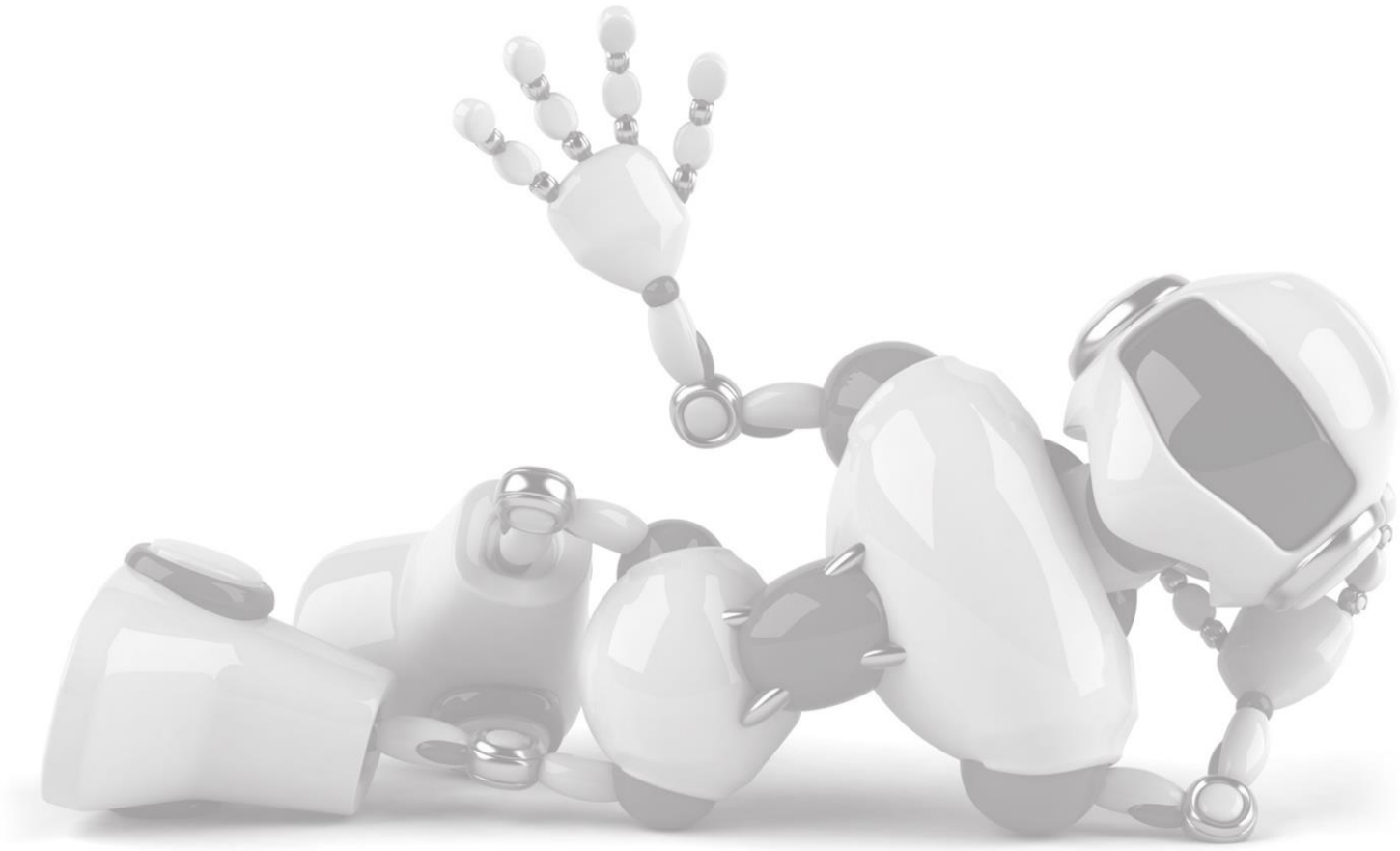
Faculdade de Ciências e Tecnologia da Universidade de Coimbra
Licenciatura em Engenharia Informática
Programação Orientada a Objectos
14-Janeiro-2015

Guuugle – Simulador de Robots Móveis

Relatório do Projecto - 2014/15



Elaborado por:

Ana Guarino 2013131907

Marta Mercier 2013136742

Introdução

No âmbito da disciplina, Programação Orientada a Objectos, foi proposto o desenvolvimento de um simulador de robots móveis.

O programa deve simular o movimento de diferentes robots (agentes) dentro de um ambiente específico, composto por vários objectos. Desta forma, os robots devem mover-se pelo ambiente de maneira a adquirir informação sobre todos os objetos que nele coexistem. Para este efeito, cada robot tem uma memória que lhe permite representar toda a informação adquirida. Possui ainda um campo de visão, o que corresponde a dizer que o agente perceciona apenas os objetos cujas coordenadas se situam dentro do campo de visão. Assim o robot deve escolher o próximo objeto a visitar usando uma de três estratégias: aleatória - o robot visita um objecto escolhido aleatoriamente do campo de visão; Máxima diferença - o robot visita o objecto que mais se diferencia dos que tem em memória; Mais Perto - o robot visita o objecto do campo de visão do qual está mais perto.

Por fim, foi desenvolvida uma interface gráfica que permite a interação com o utilizador de forma a apresentar os resultados da simulação.

Estrutura do Programa

Interface.java(main class)

JFrame Form contendo todas as funcionalidades necessárias para a apresentação da simulação.

Atributos

- *private final* Ambiente A;
- *private* javax.swing.JButton jButton1;
- *private* javax.swing.JButton jButton2;
- *private* javax.swing.JButton jButton3;
- *private* javax.swing.JButton jButton4;
- *private* javax.swing.JComboBox jComboBox1;
- *private* javax.swing.JLabel jLabel1;
- *private* javax.swing.JLabel jLabel2;
- *private* javax.swing.JScrollPane jScrollPane1;
- *private* javax.swing.JTextArea jTextArea1;
- *private int* selectedindex.

Métodos

- ***public* Interface()**

→ Responsável pela criação do Ambiente, execução da simulação e inicialização dos componentes da interface gráfica.

- ***private void* initComponents()**

→ Inicialização dos componentes da interface gráfica(*JLabel, JButton, JComboBox…*).

- ***private void* jButton1ActionPerformed(java.awt.event.ActionEvent evt)**

→ Criação do botão "Imprimir Sequência de Passos".

- ***private void* jButton2ActionPerformed(java.awt.event.ActionEvent evt)**

→ Criação do botão "Imprimir Memória".

- ***private void* jButton3ActionPerformed(java.awt.event.ActionEvent evt)**

→ Criação do botão "Imprimir Percepções".

- ***private void* jButton4ActionPerformed(java.awt.event.ActionEvent evt)**

→ Criação do botão "Imprimir Dados Estatísticos".

- **private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt)**

→ Criação da caixa de escolha do tipo de robot.

- ***public static void* main(String[] args)**

▪ Ambiente.java

Classe responsável pela execução de toda a simulação bem como da leitura e escrita em ficheiros.

○ Atributos

- *private int* largura;
- *private int* comprimento;
- *private int* tempo_vida;
- *private final ArrayList<Objecto>* objectos;
- *private final ArrayList<Agente>* agentes.

○ Construtores

- **Ambiente()**

→ Construtor por omissão da classe.

- **Ambiente(int l, int c, int tv)**

→ Construtor do Ambiente com três parâmetros: largura, comprimento e tempo de vida.

Métodos

- ***public void*** **setLargura(int l)**

→ Método que define a largura.

- ***public void*** **setComprimento(int c)**

→ Método que define o comprimento.

- ***public void*** **setTV(int tv)**

→ Método que define o tempo de vida de um robot.

- **Public ArrayList<Agente> getListaAgentes()**

→ Método que devolve a lista de agentes.

- ***public void*** **run()**

→ Executa toda a simulação.

- ***public void*** **listar_objectos(ArrayList<Objecto> lista)**

→ Lista todos os objectos do Ambiente – utilizado para debug.

- ***public void*** **escrever_ficheiro_objectos(String nome, int tipo)**

→ Escreve em ficheiro os objectos da memória e das percepções de acordo com o tipo introduzido (tipo 0: Memória; tipo 1:Percepções).

- ***public void*** **escrever_ficheiro_coordenadas(String nome)**

→ Escreve em ficheiro as coordenadas por onde o robot passou.

- ***public void*** **escrever_ficheiro_info ()**

  → Escreve no ficheiro as informações estatísticas: distância percorrida, número de objectos aprendidos e número de objectos diferentes aprendidos.

- ***public void*** **ler_ficheiro_config()**

  → Lê do ficheiro as configurações para a criação do ambiente, objectos e agentes.

- ***public boolean*** **verifica_coord(int pos_x, int pos_y)**

  → Verifica se as coordenadas pertencem ao Ambiente.

▪ Entidade.java

Classe que contem os atributos tanto dos agentes como dos obejctos.

○ Atributos

- *protected int* id;
- *protected String* cor;
- *protected String* forma_geometrica;
- *protected Coordenadas* posição.

○ Construtores

- **Entidade(int identificacao, String c, String fg, int x, int y)**

  → Construtor para a classe entidade.

▪ Agente.java

Classe que estende da classe Entidade e é através dela que são criados todos os robots de diferentes tipos.

○ Atributos

- *protected int* tipo*;*
- *protected int* campo_visao*;*
- *protected double* dist_percorrida*;*
- *protected int* n_obj_dif*;*
- *protected int* n_obj*;*
- *protected ArrayList<Objecto>* memoria;
- *protected ArrayList<Coordenadas>* caminho;
- *protected ArrayList<Objecto>* campo_visao_list;
- *protected ArrayList<Objecto>* percecoes.

- **Construtores**
  - **Agente(int identificacao, String c, String fg, int x, int y, int t, int cv)**

    → Construtor da classe agente.

- **Métodos**
  - ***public void*** **adquire_campo_visao(ArrayList<Objecto> objectos, int comprimento, int largura)**

    → Aquisição do campo de visão.

  - ***public void*** **listar_memoria()**

    → Lista todos os objectos existente na memória do agente – utilizado para debug.

  - ***public void*** **calcular_distancia()**

    → Calcula a distância percorrida do agente.

  - ***public void*** **listar_caminho()**

    → Lista as posições por onde o agente passou – utilizado para debug.

  - ***public abstract void*** **mover(ArrayList<Objecto> objectos, int comprimento, int largura)**

  - ***public void*** **mover_aleatorio(ArrayList<Objecto> objectos, int comprimento, int largura)**

    → Move o agente para uma das posições do campo de visão, aleatoriamente.

  - ***public boolean*** **verifica_pos(int pos_x, int pos_y)**

    → Verifica se o agente já esteve nessa posição.

Objecto.java

Classe estendida da classe Entidade que permite a criação de diferentes objectos no ambiente.

- ### Atributos
  - *private final String* tipo;

- ### Construtores
  - **Objecto()**

  → Construtor por omissão da classe objecto.

  - **Objecto(int identificacao, String c, String fg, int x, int y, String t)**

  → Construtor da classe objecto com as características definidas.

- ### Métodos
  - ***public String* getTipo()**

  → Devolve o tipo do objecto.

Coordenadas.java

Classe que permite guardar as posições (x,y) das entidades no Ambiente.

- ### Atributos
  - *private int pos_x;*
  - *private int pos_y.*

- ### Construtores
  - **Coordenadas(int x, int y)**

  → Construtor da classe coordenadas.

- ### Métodos
  - ***public int* getPos_x()**

  → Devolve a coordenada x.

  - ***public void* setPos_x(int x)**

  → Define a coordenada x.

  - ***public int* getPos_y()**

  → Devolve a coordenada y.

- ***public void*** **setPos_y(int y)**

→ Define a coordenada y.

▪ Aleatorio.java

Classe que se estende da classe agente, em que o robot se move para um dos objectos do campo de visão de forma aleatória.

○ Construtores

- **Aleatorio(int identificacao, String c, String fg, int x, int y, int cv)**

→ Construtor da classe Aleatório.

○ Métodos

- ***public void*** **mover(ArrayList<Objecto> objectos, int comprimento, int largura)**

→ Método que move o robot no ambiente.

▪ Mais_perto.java

Classe que se estende da classe agente, em que o robot se move para o objecto mais perto dentro do campo de visão.

○ Atributos

- *private int* dist_obj.

○ Construtores

- **Mais_perto(int identificacao, String c, String fg, int x, int y, int cv)**

→ Construtor da classe Mais_perto.

○ Métodos

- ***public double*** **calcular_dist(int atual_x,int atual_y,int pos_x_obj,int pos_y_obj)**

→ Calcula a distância entre o robot e o objecto em análise.

- ***public void*** **mover(ArrayList<Objecto> objectos, int comprimento, int largura)**

→ Método que move o robot no ambiente.

- ## Max_dif.java

Classe que se estende da classe agente, em que o robot se move para o objecto, do campo de visão, que mais se diferencia daqueles que tem em memória.

Construtores

- **Max_dif (int identificacao, String c, String fg, int x, int y, int cv)**

→ Construtor da classe Max_dif.

Métodos

- ***public void*** **mover(ArrayList<Objecto> objectos, int comprimento, int largura)**

→ Método que move o robot no ambiente.

- ***public int*** **calcular_dif_hamming(Objecto obj)**

→ Calcula a distância de Hamming entre objectos.

- ***public void*** **move_aleatoriamente()**

→ Move o agente, aleatoriamente, para a posição de um dos objectos no campo de visão.

Ficheiros de texto

- ## Config.txt

O ficheiro das configurações, designado por config.txt é composto pelos dados essenciais à criação do Ambiente bem como os dados dos agentes e dos objectos.

- ## Memoria.txt

O ficheiro memória.txt guarda a memória de cada tipo de robot. Para tal é escrito no ficheiro as caracteristicas que identificam cada um dos objectos, que existem na memória de cada robot, no fim da simulação.

- **Percepcoes.txt**

O ficheiro percepcoes.txt guarda os objectivos percepcionados por cada tipo de agente, ao longo da simulação. Como no ficheiro de memória, são escritas as caracteristicas que identificam cada um dos objectos.

- **Coordenadas.txt**

No ficheiro coordenadas.txt são guardadas as sucessivas coordenadas que cada agente ocupa ao longo da sua simulação.

- **Info.txt**

Ao longo da simulação são contabilizados o número de objectos aprendidos e o número de objectos diferentes aprendidos, sendo estes dados escritos em ficheiro juntamente com a distância percorrida, calculada no fim da simulação.

Considerações

Ao longo do projecto, em relação a alguns pontos não esclarecidos no enunciado, considerámos os pontos apresentados em seguida, com o objectivo de clarificar a elaboração do projecto.

- Campo de visão é quadrado e o seu valor representa o número de quadrículas atingidas nas diferentes direcções (como é mostrado na imagem 1);
- As percepções guardam o campo de visão que o agente tem em cada movimento;
- Cada agente tem um tipo (tipo=0 - Aleatório, tipo=1-Mais Perto, tipo=2-Máxima Diferença) definido aquando da sua criação;
- Quando o campo de visão está vazio, o agente move-se para uma posição aleatória dentro do campo de visão;
- O agente Máxima Diferença, quando tem a memória vazia, não tem objectos para a comparação, logo move-se, aleatoriamente, para a posição de um dos objectos no seu campo de visão.
- Caso o agente esteja sobre (na mesma posição) um objecto não o considera no seu campo de visão.
- Os agentes não visitam duas vezes a mesma posição. Caso haja possibilidade de acontecer:

    - Aleatório – move-se para uma posição aleatória dentro do campo de visão;

    - Mais perto – escolhe o próximo objecto mais perto. Caso já tenha visitado todos os objectos do campo de visão, move-se para uma posição aleatória dentro do campo de visão;

- Máxima Diferença – escolhe outro objecto dentro das condições estabelecidas e caso já tenha visitado todos os objectos do campo de visão move-se para uma posição aleatória do mesmo.

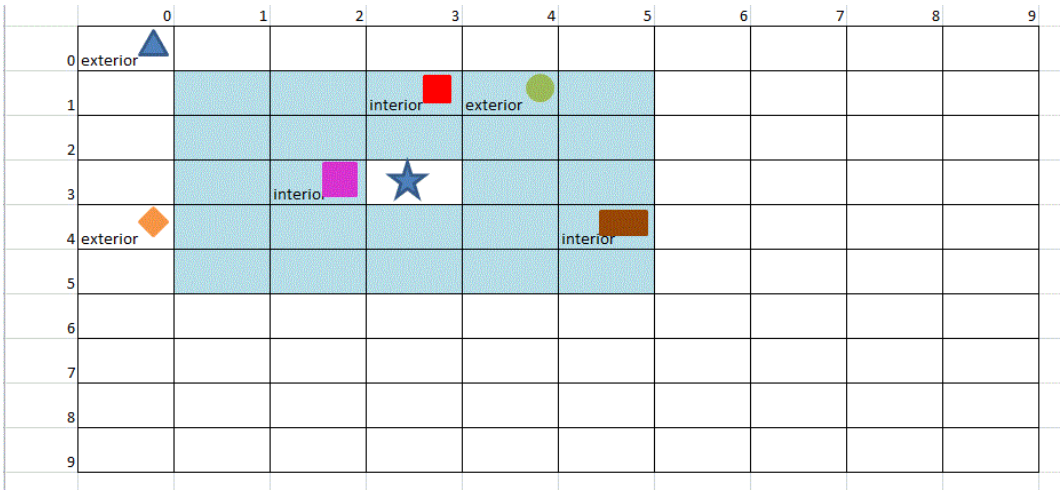
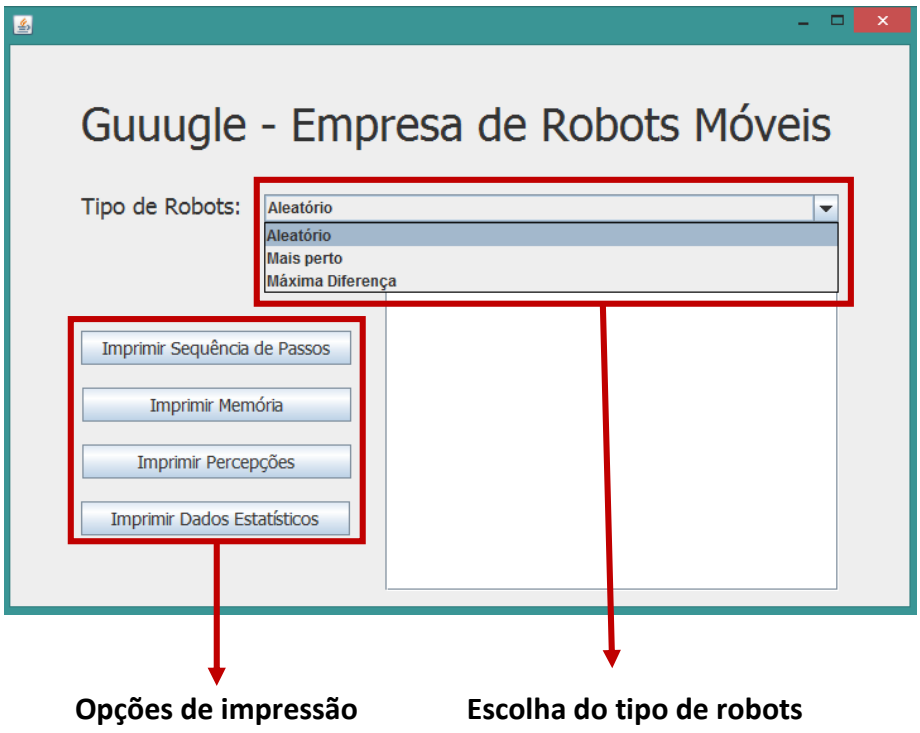


Imagem 1 - Exemplo para campo de visão com valor 2

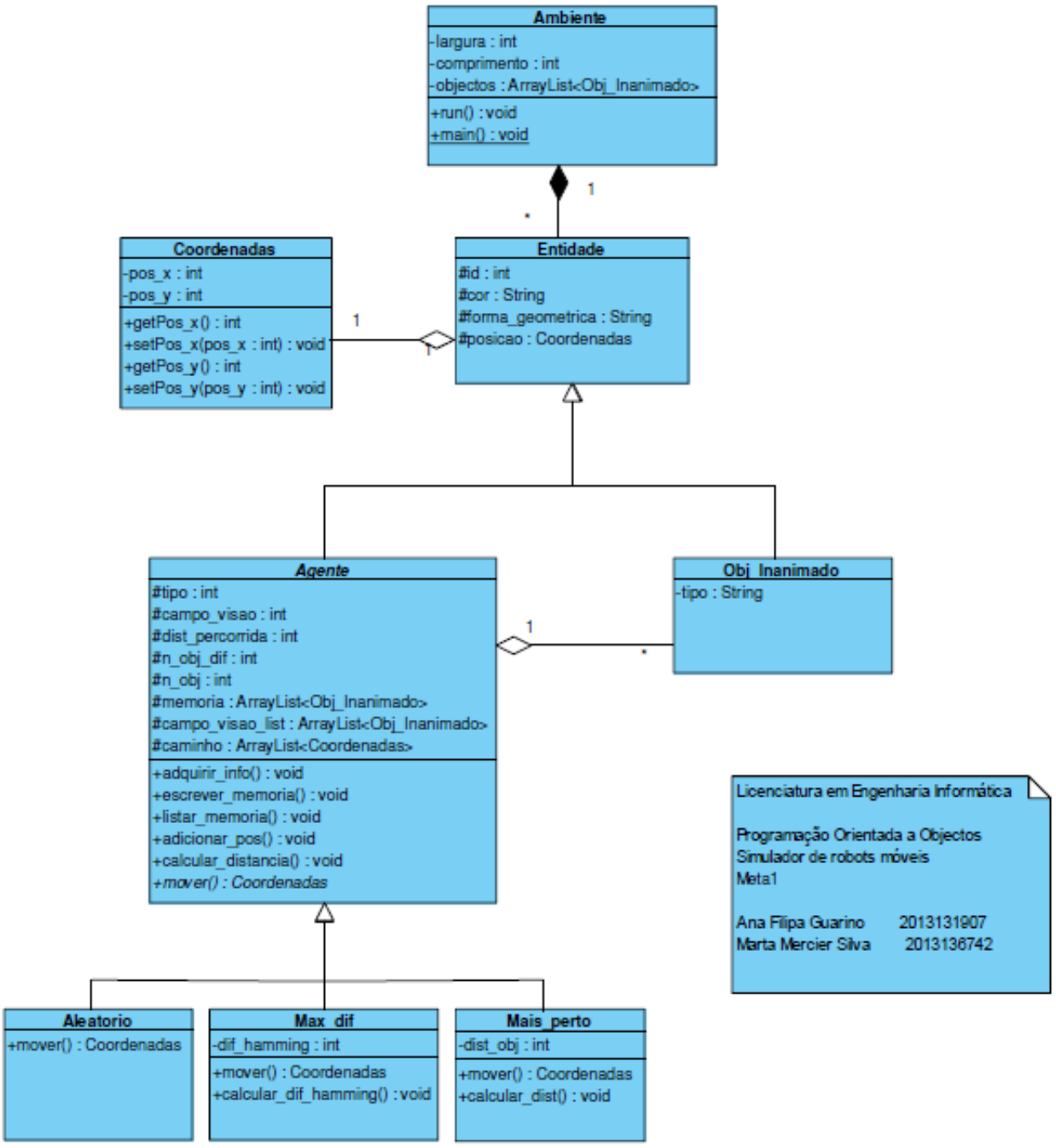
Funcionamento

De forma a apresentar o resultado ao utilizador, foi criado a seguinte interface gráfica. Na interface existe uma secção que permite escolher o tipo de robots da qual queremos imprimir os resultados e existe outra que permite escolher o que queremos imprimir.

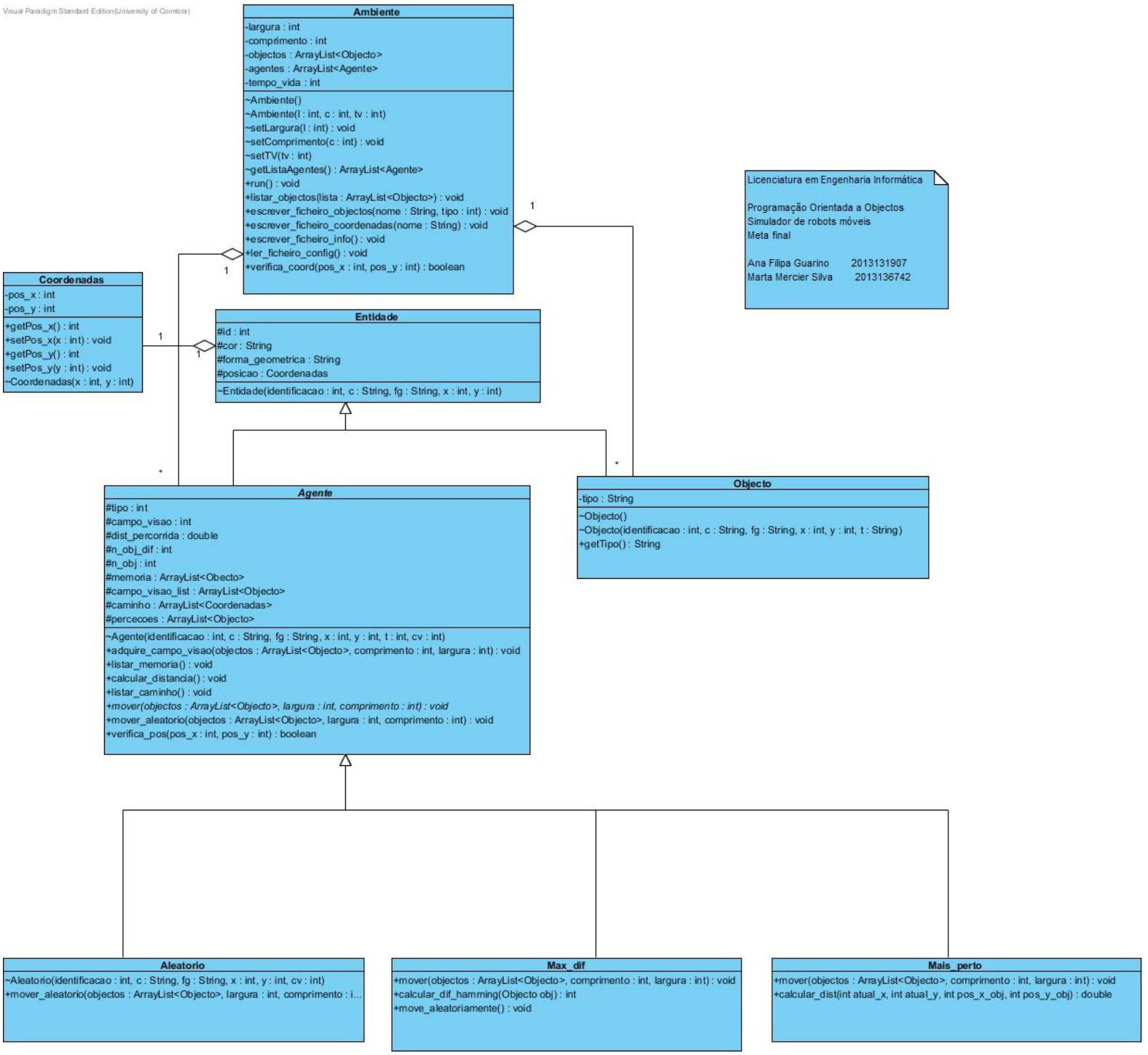


Diagramas de Classes

- **Fase inicial**



- **Fase final**


Visual Paradigm Standard Edition(University of Coimbra)
Ambiente
-largura : int
-comprimento : int
-objectos : ArrayList<Objecto>
-agentes : ArrayList<Agente>
-tempo_vida : int
~Ambiente()
~Ambiente(l : int, c : int, tv : int)
~setLargura(l : int) : void
~setComprimento(c : int) : void
~setTV(tv : int)
~getListaAgentes() : ArrayList<Agente>
+run() : void
+listar_objectos(lista : ArrayList<Objecto>) : void
+escrever_ficheiro_objectos(nome : String, tipo : int) : void
+escrever_ficheiro_coordenadas(nome : String) : void
+escrever_ficheiro_info() : void
+ler_ficheiro_config() : void
+verifica_coord(pos_x : int, pos_y : int) : boolean
Licenciatura em Engenharia Informática
Programação Orientada a Objectos
Simulador de robots móveis
Meta final
Ana Filipa Guarino 2013131907
Marta Mercier Silva 2013136742
Coordenadas
-pos_x : int
-pos_y : int
+getPos_x() : int
+setPos_x(x : int) : void
+getPos_y() : int
+setPos_y(y : int) : void
~Coordenadas(x : int, y : int)
Entidade
#id : int
#cor : String
#forma_geometrica : String
#posicao : Coordenadas
~Entidade(identificacao : int, c : String, fg : String, x : int, y : int)
Agente
#tipo : int
#campo_visao : int
#dist_percorrida : double
#n_obj_dif : int
#n_obj : int
#memoria : ArrayList<Obecto>
#campo_visao_list : ArrayList<Objecto>
#caminho : ArrayList<Coordenadas>
#percecoes : ArrayList<Objecto>
~Agente(identificacao : int, c : String, fg : String, x : int, y : int, t : int, cv : int)
+adquire_campo_visao(objectos : ArrayList<Objecto>, comprimento : int, largura : int) : void
+listar_memoria() : void
+calcular_distancia() : void
+listar_caminho() : void
+mover(objectos : ArrayList<Objecto>, largura : int, comprimento : int) : void
+mover_aleatorio(objectos : ArrayList<Objecto>, largura : int, comprimento : int) : void
+verifica_pos(pos_x : int, pos_y : int) : boolean
Objecto
-tipo : String
~Objecto()
~Objecto(identificacao : int, c : String, fg : String, x : int, y : int, t : String)
+getTipo() : String
Aleatorio
~Aleatorio(identificacao : int, c : String, fg : String, x : int, y : int, cv : int)
+mover_aleatorio(objectos : ArrayList<Objecto>, largura : int, comprimento : i...
Max_dif
+mover(objectos : ArrayList<Objecto>, comprimento : int, largura : int) : void
+calcular_dif_hamming(Objecto obj) : int
+move_aleatoriamente() : void
Mais_perto
+mover(objectos : ArrayList<Objecto>, comprimento : int, largura : int) : void
+calcular_dist(int atual_x, int atual_y, int pos_x_obj, int pos_y_obj) : double
1
1
1
1
*
*