O Visualizador do Ambiente de Simulação Ciber-Rato

António Neves, João Figueiredo, Nuno Lau, Artur Pereira e Andreia Melo

Resumo **– Este artigo apresenta uma visão geral do visualizador, construído para o ambiente de simulação Ciber-Rato, desde a sua especificação até à sua implementação prática. O ambiente Ciber-Rato consiste num conjunto de aplicações que constroem um ambiente virtual e funcionam como um sistema distribuído.**

Abstract **– This paper gives an overview of the viewer aplication developed for the simulation environment Ciber-Rato from its specifications to its pratical implementation. The Ciber-Rato environment consists of a set of applications that simulate a virtual environment and work as a distributed system.**

I. Introdução

O concurso Micro-Rato, organizado pela Universidade de Aveiro desde 1995 [1], consiste numa competição entre pequenos robots móveis e autónomos que têm como objectivo percorrer um labirinto e imobilizarem-se numa área alvo. Desde a edição de 2001 que existe a modalidade Ciber-Rato [2], destinada a concorrentes que apenas se queiram envolver nos aspectos algorítmicos do controlo de robots. Desenrola-se num ambiente virtual suportado por uma arquitectura distribuída de simulação.

Esta arquitectura de simulação é constituída por um simulador, um visualizador e vários agentes (fig. 1). O simulador é responsável pela criação de todo o ambiente virtual, incluindo o labirinto e o hardware dos robots. Os agentes são peças de software construídas pelos concorrentes que controlam os robots virtuais durante uma prova. O visualizador é responsável pela apresentação visual da prova de uma forma clara e atractiva.

O mecanismo de comunicação entre as várias aplicações assenta no protocolo UDP/IP visto este ser um protocolo simples orientado à mensagem. Quanto ao formato das mensagens, optou-se pela utilização de um modelo em *eXtensible Markup Language* (XML) porque este é um *standard* já muito difundido. O XML é um formato textual orientado à descrição de objectos e para o qual já existem actualmente disponíveis bibliotecas de manipulação, simplificando a tarefa de implementação.

Neste artigo descrevem-se as várias fases de desenvolvimento do visualizador, desde a especificação até à implementação. Apresentam-se também os resultados obtidos, com destaque para a sua utilização no âmbito do concurso Ciber-Rato 2002.

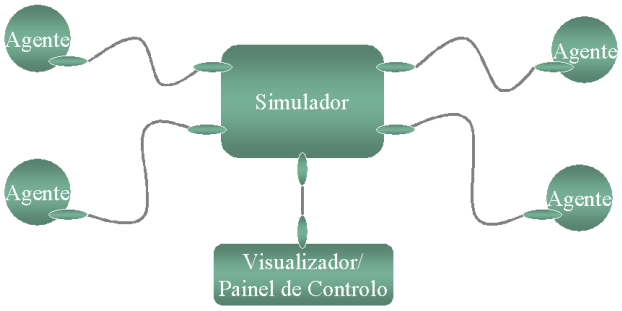

Figura 1 - Arquitectura do Ciber-Rato.

O desenvolvimento desta aplicação fez parte do projecto de fim de curso da Licenciatura em Engenharia Electrónica e Telecomunicações dos dois primeiros autores deste artigo, sob orientação dos restantes [3].

II. Especificações

Para a construção de uma aplicação que permitisse representar visualmente o ambiente de simulação do Ciber-Rato, teve-se em conta que seria necessário que essa aplicação gerasse gráficos e comunicasse com o resto do sistema.

Construiu-se uma aplicação gráfica e, para o efeito, foram seleccionadas as seguintes funcionalidades como as mais importantes para a concretização do projecto:

- Visualização de uma prova em tempo real com a informação fornecida pelo simulador. O simulador é uma aplicação *time-driven* que envia mensagens para o visualizador com uma cadência periódica. Isto implica a necessidade de um programa rápido a desenhar as eventuais alterações ocorridas durante um ciclo de simulação;
- Representação das paredes com possibilidade de escolha da cor dado que há a possibilidade de existirem paredes com alturas diferentes sendo necessário distingui-las num labirinto a duas dimensões e, para efeitos de projecção, poderá ser necessário escolher as cores que melhor se adaptem ao local da apresentação;
- Representação de cada robot através de um ícone diferente por forma a tornar mais atractiva a visualização;
- Suporte para comunicação através de socket UDP para interligação com o simulador;
- Interpretação e geração de mensagens utilizando o formato XML como método de comunicação com o simulador;
- Painel de controlo para iniciar/reiniciar a prova, pará-la e remover robots "mal comportados";

- Apresentação de um *placard* com a evolução das pontuações, do estado e do número de colisões dos robots no decorrer da prova.

III. Modelação

O desenvolvimento do visualizador começou pela sua modelação recorrendo à utilização da *Unified Modeling Language* (UML). Esta linguagem de modelação possibilita a esquematização de uma aplicação recorrendo a diversos diagramas, sem a necessidade de escrever código, permitindo assim a participação de pessoas que não estejam directamente ligadas à programação.

A representação do labirinto no visualizador é implementada com dois níveis de abstracção (fig. 2).

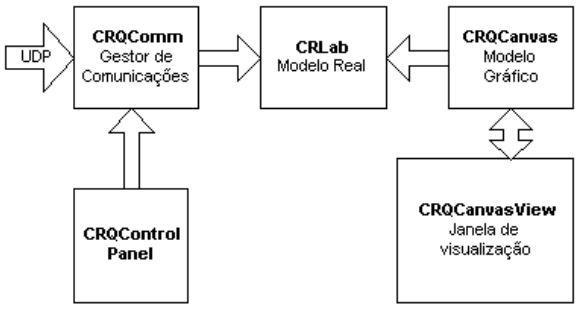

Figura 2 - Modelo simplificado do visualizador.

A um nível, um objecto (CRLab) armazena toda a informação relativa aos elementos do labirinto (robots, paredes, etc), tal como ela chega do simulador, em coordenadas reais. Num segundo nível existe um objecto (CRQCanvas) onde cada elemento do modelo real é convertido num elemento gráfico com coordenadas em píxeis para posterior apresentação. Finalmente a informação gráfica é tornada visível através de um objecto, que implementa a janela (CRQCanvasView), que desenha no ecrã todos os elementos gráficos (fig. 3).

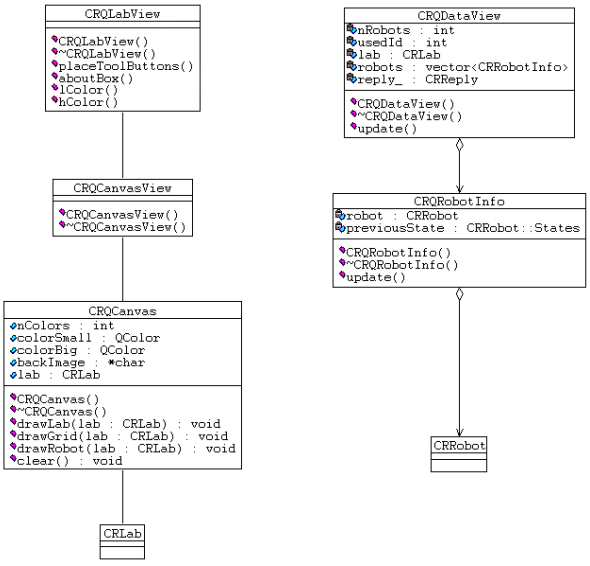

Figura 3 - Diagrama de classes em UML que implementam os vários níveis de abstracção.

Também os robots são implementados com dois níveis de abstracção, tal como o labirinto. Existe um objecto para cada robot participante que contém toda a informação do estado desse robot e que está ligado ao CRLab. No nível gráfico existe um objecto pertencente ao CRQCanvas que representa o robot na forma de um ícone com possibilidade de se deslocar e rodar no labirinto.

Como existe a necessidade de controlar o decorrer de uma prova, definiu-se um outro objecto que, ligado ao CRQComm, envia mensagens de controlo para o simulador.

A comunicação entre o visualizador e o simulador é feita recorrendo a mensagens textuais em XML, havendo a necessidade de troca das mesmas nos dois sentidos.

Mensagens do visualizador para o simulador:

- Pedido de registo do visualizador no simulador;
- Pedido do labirinto da prova;
- Pedido da grelha de partida do labirinto;
- Pedido de início/recomeço da prova;
- Pedido de paragem da prova;
- Pedido de remoção de um robot específico.

Mensagens do simulador para o visualizador:

- Resposta de aceitação ou recusa ao pedido de registo do visualizador;
- Descrição do labirinto da prova;
- Descrição da grelha de partida do labirinto da prova;
- Mensagens de estado dos robots.

Sendo as mensagens transmitidas através do protocolo UDP, poderá eventualmente existir perda de informação. No caso de mensagens de estado de um robot esta situação não é crítica, pois cada mensagem contém a informação total/absoluta sobre o estado do robot, sendo a informação perdida reposta numa outra mensagem posterior.

IV. Implementação

Na implementação referem-se os pormenores relativos à construção de classes utilizando a linguagem de programação C++ e bibliotecas de programação, bem como uma referência aos testes realizados ao visualizador mesmo antes da conclusão do simulador.

A. Classes

Na implementação do visualizador foi utilizada a linguagem de programação C++ em ambiente Linux com o auxílio das bibliotecas Qt [4]. As bibliotecas de programação Qt são construídas em C++ e disponibilizam funcionalidades gráficas, de comunicação e de manipulação de dados XML, tendo sido essenciais no desenvolvimento da interface gráfica e de comunicação do visualizador.

Foram implementadas classes por forma a disponibilizarem as funcionalidades inicialmente especificadas.

Para suporte do modelo real do labirinto destacam-se:

- CRLab — Classe base do labirinto;
- CRWall — Classe que implementa as paredes;

- CRGrid — Classe que implementa a grelha de partida;
- CRBeacon — Classe que implementa o farol;
- CRRobot — Classe que implementa o estado de um robot;
- CRTarget — Classe que implementa a área de chegada do labirinto.

Para o suporte do modelo gráfico, utilizaram-se classes Qt orientadas para desenho de objectos a duas dimensões, as quais foram redefinidas conforme a necessidade:

- CRQCanvas — Classe derivada da QCanvas que implementa a área de desenho a duas dimensões;
- CRQCanvasView — Classe derivada da QCanvasView que constrói uma janela para visualizar todos os elementos gráficos do CRQCanvas;
- CRQDataView — Classe derivada da QWidget que apresenta as pontuações, as penalizações e os estados dos robots durante o decorrer de uma prova;
- CRQControPanel — Classe derivada da QWidget que implementa o painel de controlo que disponibiliza o envio de comandos para o simulador.

Na implementação do gestor de comunicações foi construída uma classe (CRQComm) que realiza todo o processo de comunicação com o simulador.

Para a manipulação e interpretação das mensagens em XML foram utilizadas classes Qt que realizam esse processo com base no SAX2 [5], *API* para XML originalmente desenvolvido para JAVA.

Apresentam-se abaixo exemplos de dois tipos de mensagens em XML recebidas pelo visualizador, nomeadamente a mensagem de resposta a um pedido de registo e uma mensagem que representa a informação referente a um robot:

```
<Reply Status="Ok">
  <Parameters SimTime="1800" CycleTime="80"
   CompassNoise="5" BeaconNoise="5" ObstacleNoise="0.25"
   MotorsNoise="1"/>
</Reply>

<Robot Name="GUISample" Id="3" Time="636" Score="636"
 Collisions="0" State="Stopped">
  <Position X="12.0028" Y="11.019" Dir="135.489"/>
</Robot>
```

B. Testes

A utilização do XML para o formato das mensagens obrigou a adaptações no simulador que até então funcionava com um protocolo binário. Previamente à conclusão destas adaptações houve a necessidade de se realizarem testes para comprovar o funcionamento do visualizador e corrigir eventuais erros que ainda existissem. Para o efeito foi construída uma aplicação simples de comunicações em UDP, utilizando a linguagem C, que simulava as mensagens de registo do simulador para o visualizador e permitia enviar mensagens XML previamente definidas.

Após o simulador estar concluído, foram efectuados testes que incluíram as duas aplicações, simulador e visualizador, e ainda agentes utilizando os dois formatos de mensagens, caracterizados por um comportamento simples e reactivo, que posteriormente foram disponibilizados como exemplos aos concorrentes.

V. Resultados

Como resultado do trabalho desenvolvido, foi definido um pacote de software, denominado ciberTools [6], e geradas duas versões. Uma compilada estaticamente (*static link*) não sendo necessário, para a utilizar, ter instaladas as bibliotecas Qt; outra com uma compilação utilizando o formato partilhado (*shared link*) dessas bibliotecas.

O pacote de software ciberTools contém todo o software necessário para os concorrentes poderem desenvolver os seus agentes de controlo dos robots e realizarem os respectivos testes. O seu conteúdo é o seguinte:

- simulador;
- visualizador;
- código fonte de um robot para demonstração;
- diversos labirintos e grelhas de partida;
- scripts diversos para lançamento de provas.

As duas versões do pacote ciberTools foram disponibilizadas aos concorrentes antes da realização do concurso Ciber-Rato 2002 [6].

O visualizador, como parte integrante do pacote de software do ambiente de simulação, foi utilizado com sucesso no concurso Ciber-Rato 2002, inserido no encontro nacional de robótica ROBOTICA 2002, realizado na Universidade de Aveiro [1]. Na figura 4 apresentam-se as três janelas do visualizador. Na figura 5 estão representados os labirintos utilizados nas diversas mangas do Ciber-Rato 2002.

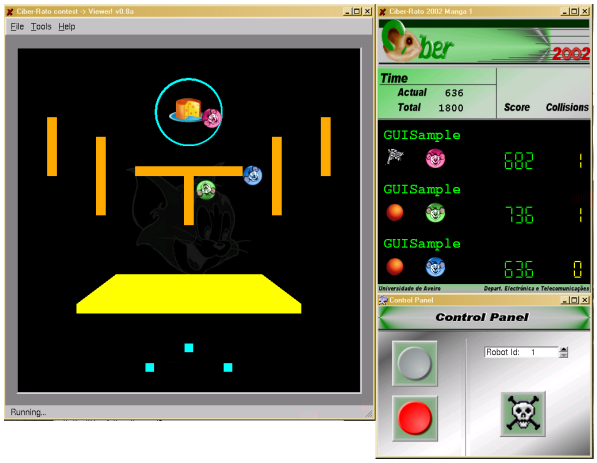

Figura 4 - Aspecto do Visualizador.

No decorrer do Ciber-Rato 2002 concorrentes e público teceram comentários favoráveis ao novo visualizador pela sua apresentação ao nível das cores dos objectos gráficos representantes dos vários elementos do labirinto.

A modalidade Ciber-Rato foi também apresentada no Dia Aberto da Universidade de Aveiro como forma de cativar os estudantes visitantes para a aprendizagem

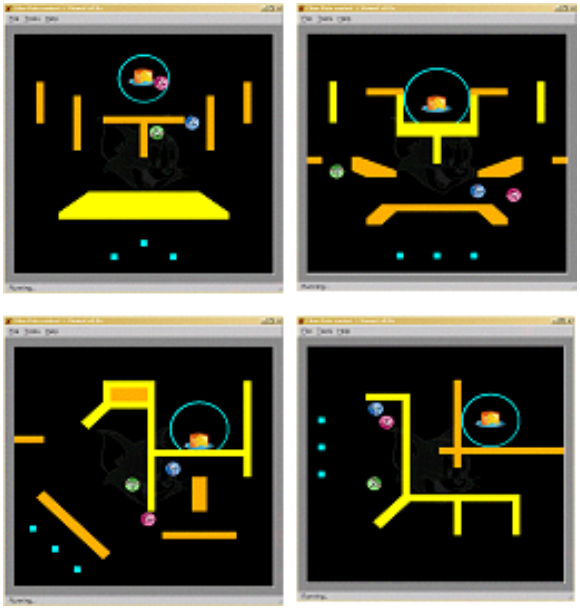

Figura 5 - Labirintos das mangas 1, 2, 3 e final do concurso Ciber-Rato 2002.

da programação em particular do desenvolvimento de algoritmos de Inteligência Artificial.

VI. Conclusões

Foi desenvolvida uma aplicação de visualização, parte integrante do software de suporte ao concurso Ciber-Rato. Esta aplicação cumpre as especificações definidas inicialmente.

O visualizador foi utilizado no decorrer do concurso Ciber-Rato 2002, recebendo comentários genericamente positivos quer dos concorrentes, quer do público. Podemos referir, porém, alguns pontos menos conseguidos que foram detectados no dia da prova, como é o caso de algum desajuste na dimensão da janela do painel de controlo.

Posteriormente à prova foi pedido aos concorrentes que enviassem por correio-electrónico críticas, comentários e sugestões para trabalho futuro por forma a tornar a modalidade mais aliciante e competitiva. Das contribuições recebidas destacam-se as seguintes relativamente ao visualizador:

- possibilidade das equipas disponibilizarem um ícone para o robot podendo ser ainda criado um prémio alternativo para o ícone mais "bonito";
- transformar as três janelas do visualizador numa única janela;
- realização de uma prova apenas com um robot no labirinto de forma a reduzir o factor sorte;

Uma das funcionalidades desejadas pelos concorrentes é a de poderem fornecer o seu próprio ícone para o robot. Isto implicará a especificação das dimensões e formato da imagem a utilizar, a definição do protocolo a utilizar para a transferência de ícones e a adaptação do visualizador para que aceite e manipule estes ícones.

Como trabalho futuro, no âmbito das ferramentas de apoio ao concurso Ciber-Rato, há a salientar o desenvolvimento de vários componentes. Construção de um *log player* com o qual seja possível reproduzir partes ou a totalidade de uma prova, a partir de um ficheiro gerado pelo simulador. A versão actual do visualizador permite a reprodução desde que o *log player* se comporte de forma idêntica ao simulador. No caso de se pretender a navegação livre no tempo virtual, será necessário redefinir o painel de controlo e criar novas mensagens de comando (por exemplo: *Step-back, Step-forward, Goto-time*).

O facto dos labirintos serem descritos utilizando um formato textual XML permite a sua definição usando um simples editor de texto. No entanto, a existência de um editor gráfico de labirintos permitiria maior flexibilidade e conforto nessa tarefa.

As regras da modalidade Micro-Rato têm evoluído, impondo novos desafios aos concorrentes, nomeadamente labirintos com maiores dimensões, regresso ao ponto de partida, paredes superiores ao plano do farol que originam zonas de sombra. Para que a modalidade Ciber-Rato acompanhe esta evolução há necessidade de se proceder a algumas alterações nos vários componentes do sistema de simulação.

Referências

[1] "Concurso Micro-Rato", http://microrato.ua.pt.

[2] Nuno Lau, Artur Pereira, Andreia Melo, António Neves e João Figueiredo, "Ciber-Rato: Um Ambiente de Simulação de Robots Móveis e Autónomos", *Revista do Departamento de Electrónica e Telecomunicações*, 2002.

[3] António José Neves e João Pedro Figueiredo, "Ferramentas de apoio ao concurso ciber-rato - Relatório da disciplina de Projecto da Licenciatura em Engenharia Electrónica e de Telecomunicações", Tech. Rep., Universidade de Aveiro, 2002.

[4] "Trolltech: Creators of Qt, the cross-platform C++ GUI toolkit", http://www.trolltech.com.

[5] "The SAX homepage", http://www.saxproject.org/.

[6] "Arquivo Técnico do Ciber-Rato", http://microrato.ua.pt/main/documentos.htm#ArqTecCR.