CiberPlayer3D – Visualizador 3D para LOGs do Ciber-Rato

Mário Rui Luzeiro

***Resumo* – No âmbito da disciplina de Computação Gráfica da LEET foi desenvolvida uma aplicação 3D, para demonstração das capacidades básicas das bibliotecas OpenGL e GLUT, que permite visualizar / reproduzir em 3D os ficheiros das provas (LOGs) do concurso Ciber-Rato.**

***Abstract* – A 3D application, to demonstrate the basic features of the OpenGL and GLUT libraries, was developed within the Computer Graphics course of the LEET: it is a 3D player / viewer of LOG files from the Ciber-Rato competition.**

I. INTRODUÇÃO

O ambiente do concurso Ciber-Rato, modalidade do Micro-Rato [1], consiste num conjunto de aplicações que constroem um ambiente virtual e funcionam em sistema distribuído. Este conjunto de aplicações é constituído por um simulador, agentes virtuais e um visualizador / painel de controlo.

A competição está estruturada em quatro mangas onde os robots competem de acordo com as regras definidas. Cada manga está dividida em provas, nas quais competem três robots de cada vez. Todas estas provas são registadas pelo simulador num ficheiro LOG, para que mais tarde possam ser novamente visualizadas ou analisadas.

O visualizador actualmente disponibilizado representa a simulação numa vista de cima (2D) e tem vindo a sofrer alguns melhoramentos desde a sua primeira versão [2]. No entanto uma representação 2D está longe de reproduzir o verdadeiro ambiente do concurso original, o Micro-Rato, onde tudo é verdadeiramente real: robots, cenário, concorrentes, público e apresentador que cria um elo e relação de intimidade com todo o espaço envolvente.

Assim, e a fim de melhorar a capacidade apelativa do Ciber-Rato e cativar ainda mais a atenção do público e concorrentes, um visualizador 3D seria um possível próximo módulo a desenvolver e a acrescentar ao conjunto de aplicações actual.

Neste artigo descreve-se o trabalho desenvolvido no âmbito da disciplina de Computação Gráfica (LEET, opção de 5º ano): um visualizador / reprodutor dos ficheiros LOG das provas do concurso Ciber-Rato, ao qual foi dado o nome de CiberPlayer3D.

O CiberPlayer3D foi implementado usando as bibliotecas gráficas OpenGL e GLUT, assim como outras bibliotecas adicionais que serão referidas posteriormente, e o ambiente de desenvolvimento Visual C++ em linguagem C/C++.

Apresenta-se também uma visão crítica do resultado obtido e uma análise ao que futuramente poderá ser feito para desenvolver um visualizador 3D com características visuais mais motivantes e atractivas.

II. DESCRIÇÃO DO CIBERPLAYER3D

O CiberPlayer3D é um programa independente que faz a leitura de LOGs das provas do concurso Ciber-Rato e as visualiza num cenário tridimensional. Assim, não é actualmente um módulo que possa ser interligado com as outras ferramentas do conjunto de aplicações disponibilizado pela organização.

Na versão actual só são lidos os LOGs do concurso de 2005, pois estes estão completamente descritos e contêm, ao contrário dos anteriores, informação do labirinto e grelha de partida. Para ser possível visualizar os LOGs anteriores a 2005, será necessário inserir manualmente no ficheiro LOG o conteúdo dos ficheiros do labirinto e grelha de partida correspondentes.

Durante a visualização poderão ser escolhidas várias posições de observação (i.e., da câmara), usando para isso o botão direito do rato; usando as teclas de cursor do teclado poderá mover-se a posição da câmara.

Na janela de visualização é mostrado, no canto superior esquerdo, o labirinto em 2D bem como a grelha de partida, o farol e a posição actual dos robots. No canto direito podem ser visualizados os nomes dos três robots em prova assim como a sua pontuação actual.

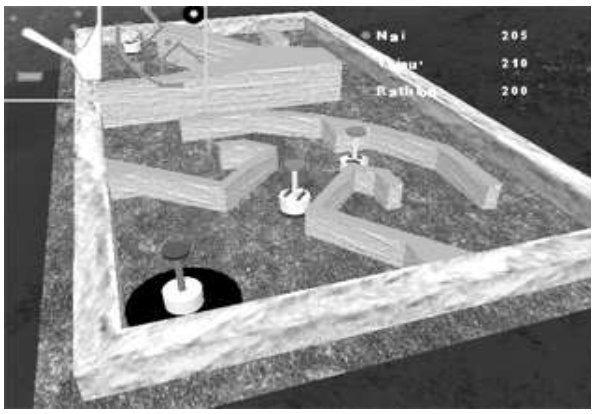

Figura 1: Ciber-Rato 2004 – Final

O cenário tridimensional consiste nos seguintes elementos: chão do labirinto com uma textura de relva, as paredes com uma textura de mármore e pintadas, por cima, com as cores respeitantes à sua altura. O labirinto é composto por um ambiente envolvente com montanhas, céu, mar e horizonte. As figuras 1 e 2 mostram o aspecto do CiberPlayer3D.

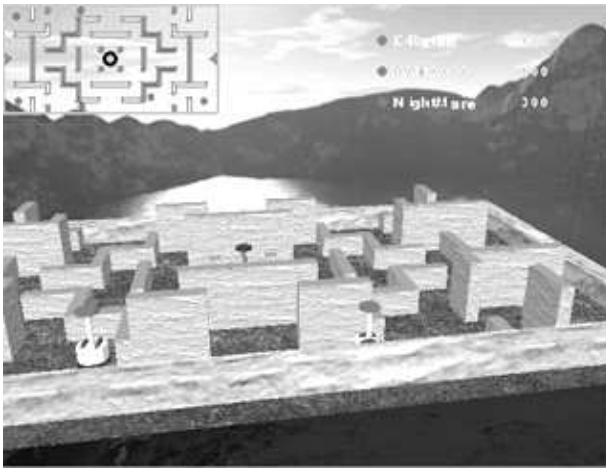

Figura 2: Ciber-Rato 2005 – Final1

III. Descrição dos ficheiros de LOG do Ciber-Rato

O ambiente de simulação do Ciber-Rato é constituído por várias aplicações, nomeadamente, simulador, visualizador e vários agentes. Entre estas aplicações existe um sistema de comunicação assente no protocolo UDP/IP usando um formato de mensagens modelado em XML (*eXtensible Markup Language*), um formato *standard* bastante difundido.

O XML é um formato baseado em simples caracteres textuais, o que permite uma sua fácil visualização e alteração com qualquer simples editor de texto. Esta simplicidade, aliada a uma grande versatilidade, permite o seu uso em sistemas bastante complexos e é, por isso, usada nas mais diversas aplicações e fins. A sua fácil manipulação e popularidade fazem com que seja possível encontrar na Web várias bibliotecas de manipulação, simplificando assim a implementação e tratamento dos ficheiros XML.

Os ficheiros de LOG do Ciber-Rato (versão 2005) contêm a informação completa de uma prova: configurações de simulação, descrição do labirinto, grelha de partida e todas as acções e estados de cada robot em prova durante o tempo de simulação.

Informações detalhadas sobre estas componentes podem ser obtidas na documentação referente ao concurso Ciber-Rato [3].

Em seguida apresentam-se excertos do LOG da prova Ciber2005_Final1. Os excertos de informação com maior interesse são os da descrição do labirinto, posição do farol e paredes:

```
<Lab Name="tos challenge" Height="14" Width="28">
 <Beacon X="14" Y="7" Height="2"/>
 <Target X="14" Y="7" Radius="1"/>
 <Wall Height="1">
  <Corner X="27" Y="7"/>
  <Corner X="28" Y="6"/>
  <Corner X="28" Y="8"/>
 </Wall>
(...continua...)
</Lab>
```

Grelha de Partida:

```
<Grid>
 <Position X="2" Y="12.5" Dir="-15"/>
 <Position X="1.75" Y="1.5" Dir="15"/>
 <Position X="26.25" Y="12.5" Dir="-165"/>
</Grid>
```

LOG Temporal:

```
<LogInfo Time="0">
 <Robot Name="ElRaton" Id="1" State="Running">
 <Pos X="2" Y="12.5" Dir="-15"/>
 <Scores       Score="0"
               ArrivalTime="0"
               ReturningTime="0"
               Collisions="0"
               Collision="False"
 />
 <Action       LeftMotor="-0.15"
               RightMotor="0.15"
               EndLed="Off"
               ReturningLed="Off"
 />
 </Robot>
  (...)
</LogInfo>
```

Para tratamento dos ficheiros de LOG em formato XML, recorreu-se à biblioteca ezXML - XML *Parsing C Library* de autoria de Aaron Voisine [4]. Trata-se de uma biblioteca bastante fácil de usar e com apenas três das várias funções disponíveis foi possível fazer a abertura do ficheiro de LOG, a navegação entre *tags* e o acesso a atributos.

IV. Construção do Labirinto

O labirinto é descrito pelas dimensões das suas paredes externas (altura e largura), a posição do farol, o chão preto e as paredes internas. Os valores de altura e largura são usadas para desenhar as paredes externas que limitam o labirinto, assim como para definir a área do chão do labirinto.

As paredes internas resultam dos obstáculos do labirinto. Cada obstáculo é descrito por um conjunto de pontos que definem a sua planta, não estando, contudo, definido um sentido de orientação para as arestas, portanto, estas tanto podem ter uma orientação no sentido directo (CW) ou inverso (CCW). Esta indefinição cria um problema ao desenhar as faces das paredes nas bibliotecas gráficas 3D que utilizem ou tenham activado a opção de *Back-Face Culling* (remoção de faces ocultas), como é o caso do OpenGL. Caso uma face tenha sido definida na orientação contrária ao definido pela biblioteca gráfica, não será visualizada por estar supostamente virada para o lado contrário à câmara. Para resolver este problema, as faces terão de ser desenhadas sem o uso de *Back-Face Culling:* isto é feito desactivando a opção GL_CULL_FACE, no caso do OpenGL.

Outro problema resultante da orientação reside no cálculo das normais às faces que são usadas para cálculos de iluminação. Devido à orientação do contorno, o vector normal poderá ficar a apontar para dentro da parede e esta ser, por isso, iluminada incorrectamente. Para resolver este problema, o vector normal deverá ser sempre considerado como virado para o lado exterior e por isso invertido se for necessário. Em OpenGL, para resolver este problema, foi activado o modo GL_LIGHT_MODEL_TWO_SIDE que inverte as normais antes do processo de iluminação, caso a face esteja definida em sentido contrário, permitindo assim uma correcta iluminação da face em relação ao ponto de observação.

Menos trivial é desenhar a parte superior das paredes dos obstáculos. Como anteriormente dito, estas são definidas através de pontos que caracterizam o seu contorno, que na maior parte das vezes representa um polígono não convexo. Um polígono é convexo se cada um dos ângulos internos medir no máximo 180 graus, assim como também se verifica que qualquer segmento de recta definido entre dois vértices do polígono não sai fora deste.

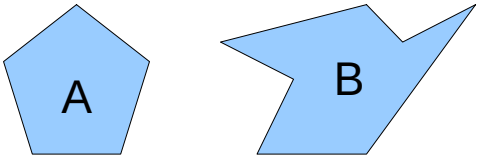

Figura 3: Exemplos de Polígonos:
A - Convexo; B - Não Convexo

A dificuldade está em preencher a área definida pelo contorno pois, usando bibliotecas gráficas como o OpenGL, temos habitualmente como unidade elementar de superfície o triângulo. Analisando o caso de um polígono convexo, exemplo da figura 3A, ao ser traçado um segmento de recta entre dois quaisquer pontos do contorno, este ainda se encontrará no interior do contorno sem intersectar nenhuma das suas arestas. No caso de um polígono não convexo, exemplo da figura 3B, esta condição não se verifica.

Nos polígonos convexos, a solução para preenchimento da área com recurso a triângulos poderia ser simplesmente determinar o centro do polígono e, para cada lado, desenhar um triângulo com um vértice no centro e os outros dois vértices sendo os pontos extremos da aresta. Esta simples solução não pode ser aplicada no caso dos polígonos não convexos. Para resolver este problema, é necessário subdividir o polígono num conjunto de triângulos de modo a que possa ser desenhado usando as funcionalidades do OpenGL.

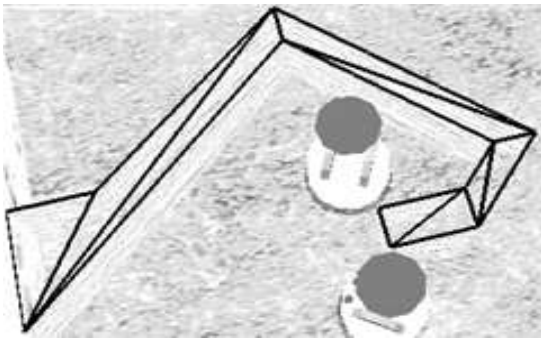

Figura 4: Triangulação da parte superior dos labirintos.

Para subdividir os polígonos existem vários algoritmos disponíveis. Um algoritmo de subdivisão de superfícies, como o de triangulação, cria uma lista de pontos que definem triângulos que cobrem toda a superfície, podendo ser usado no caso de o polígono ser ou não convexo.

Como no âmbito deste trabalho não se pretendia desenvolver um algoritmo deste género, optou-se por usar uma pequena biblioteca de funções criada por John W. Ratcliff e disponibilizada na Web [5], permitindo assim um fácil tratamento do problema e a determinação dos vértices dos triângulos do topo dos obstáculos [Figura 4].

V. Construção dos Restantes Elementos

Os modelos dos agentes robóticos e do farol foram criados usando um programa de modelação 3D externo e exportados para o formato OBJ. Este formato [6], desenvolvido inicialmente pela *Silicon Graphics*, foi escolhido por já ter sido usado nas aulas práticas da cadeira e ser de fácil tratamento. Foi acrescentada a capacidade de identificar grupos de objectos definidos no ficheiro para que pudessem ser distinguidos elementos do modelo (roda, base, botões, etc.) No entanto, este formato mostrou-se extremamente limitado para possíveis melhorias futuras, como será discutido posteriormente.

Para melhoramento do aspecto visual é desenhado, como adição ao cenário normal do ambiente do concurso Micro-Rato, uma *skybox* que facilmente recria um ambiente de fundo, dando a ilusão de um mundo envolvente infinito.

Na figura 2 é perceptível a sensação de envolvente criada pelas montanhas, lago e nuvens em fundo.

A sua facilidade de implementação deve-se ao facto de apenas ser preciso desenhar um cubo texturado para recriar o efeito.

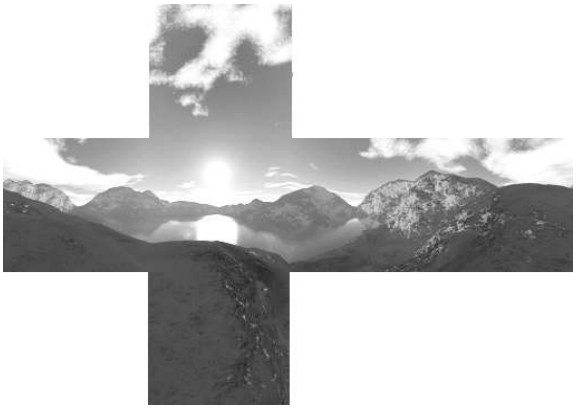

Figura 5: Planificação da textura do cubo usada numa *skybox*

As texturas (Figura 5) são criadas usando outros programas de *rendering* 3D avançados e podem ser facilmente obtidas livremente na Web [7].

VI. Conclusões

Com este pequeno projecto, não só se desenvolveu uma aplicação 3D que fizesse uso das capacidades básicas das bibliotecas OpenGL e GLUT, mas também se demonstrou a importância e impacto que se obtém com uma simples visualização 3D de um ambiente de simulação de robots virtuais. Muito mais poderia ter sido ainda feito e melhorado mas, devido ao âmbito da cadeira em que se inseria, não foram exploradas mais capacidades. São, no entanto, sugeridas na parte final deste artigo ideias futuras a ter em consideração numa nova abordagem a um visualizador 3D.

O formato XML revelou-se bastante simples de ser trabalhado recorrendo a bibliotecas externas de fácil uso; a possibilidade da sua visualização usando um editor normal de texto permitiu uma fácil análise da informação contida nos ficheiros de LOG.

O formato OBJ apesar de bastante fácil de ser tratado apresentou capacidades diminutas e foi pouco eficaz. As características que apresenta não são muito atraentes e, se quisermos armazenar informação mais complexa no ficheiro do modelo, teremos de recorrer a outro formato mais robusto e com capacidades adicionais.

As bibliotecas OpenGL e GLUT mostraram-se bastante capazes, com uma utilização muito simples mas de grandes capacidades e elevado potencial. A grande popularidade e divulgação destas bibliotecas permitiram facilmente encontrar boa documentação e programas exemplificativos por toda a Web, fazendo assim com que a sua aprendizagem se tornasse bastante fácil e agradável. Estas bibliotecas possuem uma elevada facilidade de portabilidade que é um importante aspecto a ter em conta numa futura abordagem ao visualizador.

VII. Sugestões para futuras abordagens

Apesar da simplicidade do CiberPlayer3D, este abre o apetite para a criação de um projecto mais sério de um visualizador 3D a integrar no conjunto de programas do Ciber-Rato. Mesmo com o pouco tempo despendido na criação deste programa conseguiu-se, com o resultado final, perceber o impacto causado por um visualizador 3D e que, portanto, esta ideia deverá ser levada em frente e melhor trabalhada.

Em relação às características actuais das aplicações de Computação Gráfica, a implementação apresentada fica muito longe do realismo que se consegue obter com as actuais placas gráficas de baixa gama. Hoje em dia, com o recurso a *shaders* por *hardware* é possível visualizar em tempo real efeitos de difusão, mapeamento de textura, reflexão e refracção, sombras e outros efeitos de pós-processamento.

Para além de um forte apelo visual ao nível geral, desejável a uma aplicação deste fim para cativação do público, ao nível do utilizador do programa deveria ser criado uma interface e um sistema configurável de visualização. Seria desejável escolher e adicionar temas gráficos no cenário e labirinto, bem como a possibilidade de visualizar outra informação disponível, tal como a velocidade aplicada nos motores e outra informação técnica detalhada, útil em tempo de desenvolvimento dos agentes. Informação útil adicional contida no LOG, que não está a ser usada no CiberPlayer3D e que deverá também ser mostrada, é a indicação de ocorrência de uma colisão e o estado dos *LEDs* do robot.

VIII. Agradecimentos

Agradece-se aos Professores Artur Pereira, Joaquim Madeira e Nuno Lau pelas ideias e incentivos para a realização do trabalho e a escrita deste artigo.

Referências

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

[2] António Neves, João Figueiredo, Nuno Lau, Artur Pereira e Andreia Melo, "O Visualizador do Ambiente de Simulação Ciber-Rato" - Revista do DETUA, Vol. 3, Nº 7, Setembro 2002

[3] Concurso Ciber Rato: Regras e Especificações Técnicas da Modalidade Ciber Rato. Disponível em [1].

[4] ezXML - XML Parsing C Library - Aaron Voisine aaron@voisine.org http://ezxml.sourceforge.net/

[5] Biblioteca de funções para triangulação, autoria de John W. Ratcliff: http://www.flipcode.org/cgi-bin/fcarticles.cgi?show=63943

[6] Especificações do Formato OBJ: http://www.csit.fsu.edu/~burkardt/data/obj/obj_format.txt

[7] Amostras grátis de texturas para skyboxes: http://www.scentednectar.com/skyboxes/