Visualizador 3D para o Ciber-Rato

Joaquim Fonseca, Flávio Fonseca, Artur Pereira, Paulo Dias,

Resumo - O Ciber-Rato é uma competição de robótica, em ambiente simulado, que decorre anualmente na Universidade de Aveiro. O módulo de visualização do ambiente permite actualmente uma visualização em duas dimensões, nascendo por isso a vontade de criar uma ferramenta que permita visualizar a competição em 3 dimensões. Esta ferramenta foi desenvolvida no âmbito da disciplina de Modelação e Visualização 3D com base na biblioteca VTK (Visualization Toolkit).

I. INTRODUÇÃO

A unidade de visualização da competição Ciber-Rato é actualmente um ambiente 2D surgindo de forma natural a necessidade de criar um visualizador mais avançado que permita uma visualização 3D com vários pontos de vista. Esta ferramenta foi desenvolvida no âmbito da disciplina de Modelação e Visualização 3D (opção de 5º ano para os alunos do Mestrado Integrado em Engenharia Computadores e Telemática), sendo o objectivo demonstrar as capacidades de modelação e visualização da biblioteca gráfica VTK (Visualization Toolkit) [4]. Esta ferramenta não inclui ainda o módulo que permite a comunicação com o simulador, e responde actualmente a movimentos do teclado. No futuro, o visualizador 3D poderá vir a ser integrado com o simulador, para que a competição possa ser realizada em ambientes 3D.

II. CIBER-RATO

O Ciber-Rato [1, 2] é uma competição de robótica em ambiente simulado, que surgiu em 2001, integrada no concurso Micro-Rato [1, 3], complementando a competição que até aí se fazia usando robôs reais. Trata-se de um concurso de robótica onde vários robôs autónomos se movimentam num recinto povoado de obstáculos com um duplo objectivo: primeiro têm de localizar e alcançar um farol; de seguida têm de regressar ao seu ponto de partida. No Ciber-Rato os robôs são criados virtualmente por um simulador e têm todos as mesmas capacidades, isto é, têm todos os mesmos órgãos sensoriais e de actuação. Os concorrentes têm que desenvolver os agentes (software) que comandam esses robôs. O simulador envia a cada agente os valores medidos pelos sensores, recebe dele os comandos sobre os "motores" e actualiza em conformidade a posição do robô no recinto. Os agentes têm de, com base nos dados sensoriais recebidos do simulador, decidir como comandar o seu robô no sentido de cumprir da melhor forma os objectivos estabelecidos nas regras da competição.

O Ciber-Rato já extravasou os limites do Micro-Rato. Desde 2006 que uma competição inspirada no Ciber-Rato e usando as mesmas ferramentas, o CiberMouse@RTSS, tem sido realizada no âmbito do *Student Contest* do *Real-Time Systems Symposium*.

A. O ambiente de simulação

O ambiente de simulação, tal como ilustrado na figura 1, é constituído por um simulador, um visualizador, vários (tipicamente 3) agentes robóticos e 3 ficheiros de configuração. A comunicação entre o simulador e os agentes e o visualizador é feita através de mensagens XML, suportadas por ligações TCP/UDP. Os ficheiros de configuração, também usando o formato XML, definem o recinto de competição, a grelha de partida (posição inicial dos robôs em prova) e alguns parâmetros de simulação, como por exemplo nível de ruído dos sensores e actuadores. A descrição do recinto de competição (labirinto) contém as dimensões exteriores do recinto, a localização e configuração dos obstáculos e a localização dos faróis.

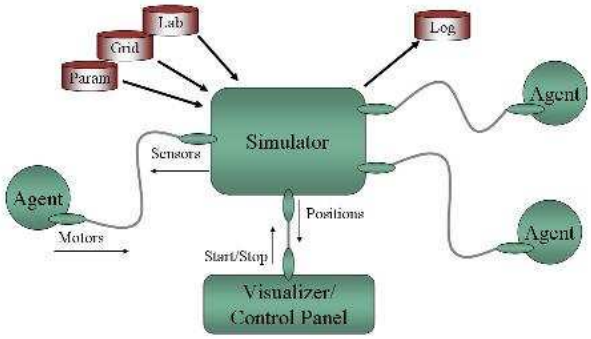

Figura 1: O ambiente de simulação do Ciber-Rato

No início de uma simulação o simulador carrega as descrições do labirinto e grelha e envia-os ao visualizador, não lhe impondo qualquer restrição na forma como o cenário de competição deve ser apresentado ao espectador. A construção de um visualizador 3D torna-se assim possível sem qualquer alteração no simulador.

Após o arranque o simulador envia ao visualizador em cada ciclo o estado da prova. Entre outros elementos, este estado define a localização e orientação de cada um dos robôs em prova.

B. A visualização

Embora o simulador seja a peça mais importante do pacote de ferramentas de suporte ao Ciber-Rato, no âmbito do trabalho apresentado neste artigo é o visualizador que interessa. A aplicação de visualização que actualmente é usada modela em duas dimensões o labirinto virtual, usando para isso uma única vista situada num ponto fixo por cima do labirinto, tal como se ilustra na figura 2.

Figura 2: Visualizador 2D do Ciber-Rato

A criação de uma ferramenta 3D torna possível visualizar o labirinto de qualquer ponto do espaço, oferecendo uma visualização mais realista a todos os níveis. Pretende-se ainda que, além da janela de visualização principal, exista uma janela secundária que corresponda à vista de um qualquer robô seleccionado, dando-se assim ao espectador a noção do que existe à frente desse robô em cada instante.

III. REQUISITOS DO VISUALIZADOR

O objectivo fundamental desta ferramenta é a construção de uma representação 3D dos vários elementos envolvidos numa prova da competição Ciber-Rato. Estes elementos são: um recinto rectangular limitado por paredes exteriores, obstáculos tridimensionais espalhados dentro do recinto, um ou mais faróis, uma ou mais áreas-alvo, uma grelha de partida e vários robôs. Com excepção destes últimos, que se podem deslocar ao longo do recinto, os restantes elementos são estáticos.

A ferramenta deve , a partir dos ficheiros XML com a descrição dos labirintos e sem qualquer tipo de informação da estrutura, ser capaz de proceder à sua representação. Para isso é necessário construir um reconhecedor (*parser*) dos ficheiros de descrição, uma vez que o formato desses ficheiros está pré-definido. Desde o início foi definido como requisito a capacidade de poder alterar a cada momento o ponto de vista do observador do recinto.

Como referido atrás os robôs são elementos móveis. Assim, a ferramenta deve permitir que com eles se interaja no sentido de mudar a sua posição e orientação. Numa versão final da ferramenta, já integrada no pacote Ciber-Rato, o movimento é determinado pelas mensagens de estado enviadas pelo simulador. No âmbito deste trabalho optou-se por comandar os robôs através do teclado.

Finalmente, definiu-se que deveria existir uma janela de visualização secundária que mostrasse o recinto de competição do ponto de vista de um dos robôs. Esta vista deve mostrar o que uma câmara montada no robô veria.

Todos estes objectivos foram alcançados. A integração no pacote Ciber-Rato fica apenas pendente da realização de um sistema de comunicação com o simulador da competição para permitir a utilização do programa com a aplicação real.

IV. VTK (VISUALIZATION TOOLKIT)

O VTK é uma biblioteca de modelação 3D orientada a objectos muito utilizada pelas suas capacidades na área de visualização e processamento de imagem. Pode ser usada em várias linguagens tais como TCL, python, Java ou C++, sendo esta última a escolhida para desenvolver esta aplicação.

O VTK admite várias formas de representar a informação, incluindo conjuntos desorganizados de pontos, malhas poligonais, imagens, volumes e grelhas, estruturadas ou não. Disponibiliza também filtros e algoritmos que permitem operar sobre todas estas representações [4,5].

Apesar da sua complexidade e dimensão, o VTK torna-se bastante intuitivo após alguma familiarização. A principal limitação está na falta de documentação e informação.

A. vtkPolyData

O VTK é por natureza uma linguagem orientada a objectos, e como tal existem várias classes com características muito próprias em função das necessidades de cada representação, e de acordo com os dados a visualizar. A informação a representar neste projecto consiste sobretudo em malhas poligonais. Tendo como base as aulas da disciplina de Modelação e Visualização 3D, optámos pela utilização de uma determinada classe: *vtkPolyData*.

Esta classe da biblioteca VTK, é apropriada para construir polígonos com uma lista de pontos. Para a sua correcta utilização foi associado ao objecto *vtkPolyData* uma lista dos vértices que irão compor o modelo 3D, assim como uma lista de índices destes vértices para construir cada face, para que o visualizador saiba como organizar os vértices.

B. Pipeline de Visualização

A visualização de modelos na biblioteca VTK é feita através de um *pipeline*, criando inicialmente primitivas, às quais irão ser aplicados filtros para que possam finalmente ser visualizados. Neste caso específico, e como irá ser explicado de seguida, as primitivas são do tipo *vtkPolyData* que podem ser compreendidos como uma

simples lista de pontos. Este objecto terá que ser mapeado tendo em conta os vértices e a sua ordem para poder construir uma forma geométrica, recorrendo para isto á classe *vtkPolyDataMapper*. De seguida, é necessário juntar a informação do objecto numa representação simples que possa ser tratada pela biblioteca como um objecto na cena a modelar. Para satisfazer esta necessidade, recorremos á classe *vtkActor* que, tal como o nome indica, torna o objecto numa representação abstracta possível de colocar numa dada posição na cena. Esta *pipeline* de visualização aparece de uma forma resumida na figura que se segue.

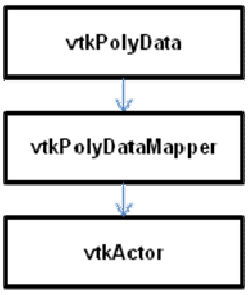

Figura 3: Pipeline de visualização

Existe no caso concreto deste problema, uma excepção a este fluxo, devido à representação de polígonos côncavos (um dos ângulos internos é superior a 180º). A representação de um qualquer polígono côncavo pelo VTK apresenta erros, uma vez que aparecem zonas "fantasma" que não pertencem de facto ao polígono em causa. Este problema foi resolvido recorrendo a um filtro que efectua a triangulação do polígono usando o algoritmo de triangulação de Delaunay [4,8], conseguindo-se assim efectuar a correcta visualização do polígono, quer este seja câncavo ou não. Este fluxo secundário é apresentado na figura 4.

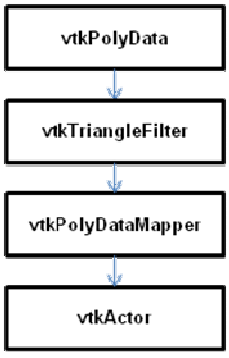

Figura 4: Pipeline de visualização para polígonos côncavos

C. vtkInteractor

Associada à modelação de um determinado ambiente está a sua correcta visualização, que deverá permitir ao utilizador algum tipo de manipulação e interacção. É para satisfazer esta necessidade que a biblioteca VTK disponibiliza a classe *vtkInteractor*. Esta classe define a interacção entre utilizador e a cena, permitindo mudar o ponto de vista, rodar a cena, visualizar os objectos com vários tipos de representações (*solid*, *wireframe*), assim como aproximar ou afastar o ponto de vista dos objectos.

D. vtkVRMLImporter

Existe para cada tipo de objecto constituinte do labirinto uma representação geométrica específica, podendo ser usados conjuntos de polígonos ou esferas para os representar, exceptuando o caso dos agentes robóticos que não tem um representação específica. Neste caso decidimos utilizar uma representação dos robôs que competem na competição Micro-Rato, já referida anteriormente. Este modelo foi-nos fornecidos em formato VRML, que é um formato reconhecido pelo VTK. A importação destes modelos foi feita criando um objecto do tipo *vtkVRMLImporter* que armazena a informação relativa aos mesmos, e que possibilita a posterior transformação deste objecto para um objecto do tipo vtkPolyData. O restante processo de visualização é exactamente análogo aos outros objectos do mesmo tipo.

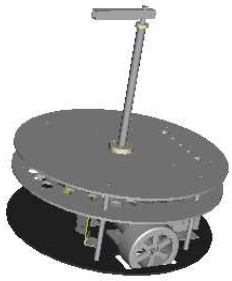

Figura 5: Vista do modelo VRML do robô Micro-Rato

E. Visualização e interacção

O visualizador desenvolvido permite uma visualização em duas janelas distintas. Na primeira a câmara encontra-se inicialmente numa posição que permite visualizar todo o labirinto. Como esta janela está associada a um *interactor* é possível interagir com a visualização através do rato.

A segunda janela tem a câmara posicionada no topo do robô seleccionado de modo a simular o campo de visão do mesmo, dando assim noção das dificuldades que o robô tem de ultrapassar para chegar ao farol. A esta janela também está associado um *interactor* que permite ao utilizador mudar o ângulo de visualização. Para fazer a leitura dos eventos do teclado e actualizar a posição dos robôs, existe uma função que lida com estes mesmos eventos e, executa a acção correspondente a cada tecla premida. Esta função é nomeada de *callback*, e estando associada ao *interactor* é evocada sempre que existe algum tipo de acção no mesmo. Esta *callback* permite que, ao premirmos a tecla correspondente a uma direcção, a

posição do robô seja actualizada assim como a iluminação e a câmara a ele associados.

V. MODELAÇÃO

No início do desenvolvimento desta aplicação foi feita uma divisão clara entre cada conjunto de funcionalidades para que estas pudessem ser desenvolvidas de forma faseada e separada. A realização da aplicação resumiu-se assim ao desenvolvimento e teste de cada parte, para posteriormente se iniciar o desenvolvimento da parte seguinte. Pode-se assim definir o *workflow* desta ferramenta de uma forma simples em 4 partes:

- leitura dos ficheiros XML e preenchimento das estruturas de dados,
- construção dos modelos a partir dos dados das estruturas,
- importação de modelos vrml
- finalmente visualização da cena.

A. Leitura dos ficheiros XML

Tal como já foi referido anteriormente, a descrição da posição e estrutura dos vários objectos no labirinto é feita a partir de dois ficheiros XML. O requisito inicial desta aplicação resume-se à leitura e interpretação destes mesmos ficheiros e preenchimento das estruturas respectivas que vão servir de base à criação dos vários objectos.

A leitura destes ficheiros foi feita de um forma muito simples utilizando a estrutura pré-definida dos dados e recorrendo a um comando simples para leitura de ficheiros, tendo apenas em conta a formatação inicial de cada linha para identificar o tipo de informação. Este método embora se tenha demonstrado bastante fiável, pode ser futuramente melhorado recorrendo a um *parser* mais elaborado, para poder fazer a leitura de ficheiros com outro tipo de estrutura.

Quanto ao armazenamento destes dados, grande parte das estruturas de dados por nós utilizada, foi herdada do código fonte do visualizador 2D já existente. Este código é *openSource* e está disponível na página da competição [1]. Uma vez que os dados a guardar são os mesmos, decidimos que seria viável utilizarmos algumas das estruturas de dados, ainda que com algumas modificações. Apenas para os robôs foi criada uma estrutura de dados para guardar a informação dos mesmos, uma vez que estes serão os únicos objectos que irão mudar de posição, e necessitam por isso de outras formas de acesso mais fácil e rápido.

B. Criação dos modelos

Existem neste ambiente dois tipos de objectos: os que têm uma estrutura bem conhecida e os cuja estrutura é definida pela lista de vértices que os compõem. Na primeira categoria encontram-se os robôs, a área-alvo e o farol. Têm estruturas bem definidas e como tal a criação de modelos que os representem é uma tarefa relativamente directa. A representação destes objectos necessita apenas da leitura dos seus atributos da estrutura de dados e da criação do objecto correspondente, seja uma esfera no caso do farol ou um cilindro no caso da área-alvo.

No ficheiro de descrição, um obstáculo é definido por uma altura e por uma sequência de vértices. Por um lado, esta sequência representa as faces superiores e inferiores do obstáculo. Por outro lado, cada par adjacente de vértices representa uma face vertical. A criação dos obstáculos torna-se assim uma tarefa mais complexa uma vez que é necessário criar uma representação tridimensional a partir de uma lista de vértices em duas dimensões. Uma sequência de *n* vértices dá origem a *n+2* polígonos: *n* faces verticais, uma horizontal por altura do chão e uma horizontal pela altura do obstáculo.

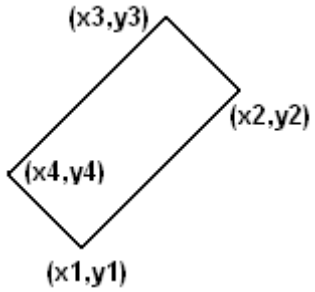

Figura 6: Obstáculo em 2D

De uma forma muito simples este processo resume-se a transformar pares de vértices em grupos de 4 vértices, ou seja, transformar segmentos de recta em polígonos. Os segmentos de recta que representam cada face do obstáculo, irão ser transformados portanto em polígonos para poder obter uma representação tridimensional destas mesmas faces.

A estratégia adoptada para conseguir uma boa representação destes objectos, consistiu na criação de apenas um objecto do tipo *vtkPolyData* que irá representar as faces laterais de todos os obstáculos do labirinto, objecto este que, como já foi referido, é constituído por uma lista de pontos e uma lista de índices destes pontos que irá servir para os agrupar em polígonos. Por cada par de vértices na estrutura, é necessário introduzir no *vtkPolyData* dois pares de vértices, um no plano z=0 e outro no plano z que representa a altura do obstáculo, segmentos estes que definem entre si um polígono. As Figuras 6 e 7 ilustram como são representados os obstáculos em 2D e posteriormente em 3D.

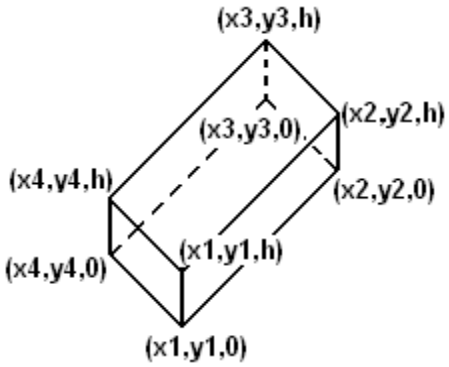

Figura 7: Obstáculo em 3D

Com este método foi possível representar todas as faces laterais dos obstáculos. Mas, faltam ainda as faces superiores. Contrariamente às faces laterais, que são sempre constituídas por 4 vértices, as faces superiores podem ser constituídas por um número desconhecido de vértices, podendo, em consequência ocorrer polígonos côncavos. Devido ao problema, já referido anteriormente, da ineficiência na representação de polígonos côncavos foi necessário recorrer a uma triangulação usando o filtro *vtkTriangleFilter*. Neste caso criámos um objecto do tipo *vtkPolyData* para cada face a representar. As faces inferiores podem ser ignoradas porque estão em contacto com o chão do labirinto.

C. Importação de modelos externos

Para a construção do visualizador foi apenas necessário importar do exterior os modelos VRML correspondentes à representação tridimensional do robô. Este modelo é importado através do objecto *vtkVRMLImporter*, tal como já foi referido, mas é necessário um certo processamento auxiliar para garantir que este modelo é representado correctamente. Ao importar de um ficheiro VRML para o VTK, o elemento é internamente representado por vários objectos *vtkPolyData* (no caso específico do robô eram 70), o que resulta numa visualização ineficiente. A solução passou pela implementação de uma função que dado um determinado objecto do tipo *vtkVRMLImporter* devolve um outro objecto *vtkAppendPolyData* que resulta da fusão de todos os objectos *vtkPolyData* que o constituem. Este processo permite criar uma abstração das partes que constituem o objecto importado, para que possa ser "entendido" pelo VTK como um só.

D. Visualização

A visualização no VTK é feita de uma forma muito simples, que se resume na transformação de primitivas em objectos que podem ser entendidos pelo VTK como actores da cena a modelar. Mais especificamente, no caso dos objectos do tipo *vtkPolyData*, o processo de visualização necessita que estes objectos sejam mapeados para um objecto do tipo *vtkPolyDataMapper* e posteriormente para um objecto do tipo *vtkActor*. Neste ponto é apenas necessário criar um *render vtkRenderer*, adicionando-lhe todos os actores, e criando posteriormente um janela de visualização que terá como objecto de modelação este mesmo *render*. A este *render* poderemos também adicionar luzes ou câmaras, tal como é exemplificado na figura seguinte.

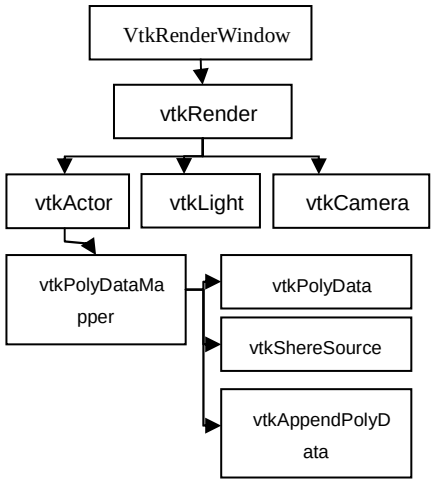

Figura 8: Diagrama de visualização do VTK

VI. RESULTADOS

Tal como já foi mencionado anteriormente, este trabalho foi efectuado no âmbito da disciplina de Modelação e Visualização 3D, e tinha como principal finalidade a representação e modelação de modelos tridimensionais. Como tal, esta ferramenta é, por enquanto, uma aplicação isolada com função de visualização do labirinto, podendo os robôs ser apenas controlados a partir do teclado. Não está ainda integrada com o simulador da competição mas tal poderá ser facilmente conseguido.

As funcionalidades implementadas foram as que se encontravam descritas nos requisitos iniciais, sendo assim a ferramenta capaz de representar qualquer labirinto da competição, desde os ficheiros XML estejam de acordo com o formato estabelecido. A visualização é complementada com uma segunda janela em que a câmara se refere ao ponto de vista de um qualquer robô. As figuras 9 e 10 mostram o resultado final, sendo a pequena janela no canto superior direito a vista na perspectiva de um robô.

O utilizador da ferramenta pode interagir no sentido de mudar o ângulo de visão da vista global e de seleccionar o robô associado à janela secundária.

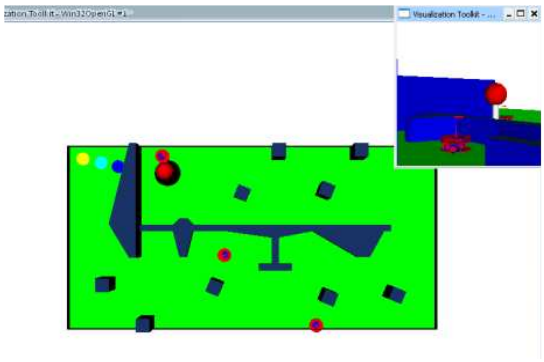

Figura 9: Vista global superior do recinto de jogo

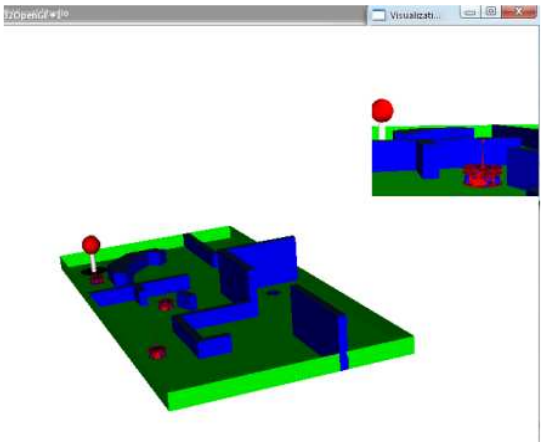

Figura 10: Vista global lateral do recinto de jogo

VII. CONCLUSÕES E TRABALHO FUTURO

No final do desenvolvimento desta ferramenta, podemos concluir que foi bastante interessante e educativo no que toca à visualização e modelação 3D, assim como bastante compensador pensar que esta aplicação poderá servir como base de trabalho futuro para desenvolver um visualizador real.

O VTK revelou-se uma linguagem bastante atractiva e poderosa quanto à modelação e visualização de modelos tridimensionais, sendo na nossa opinião uma biblioteca excelente para dar os primeiros passos na visualização e modelação 3D, uma vez que sendo orientada por objectos torna fácil criar os ambientes desejados.

Quanto ao trabalho futuro, a primeira possibilidade consiste em substituir o visualizador 2D do simulador do Ciber-Rato, faltando ainda a parte de integração funcional com o mesmo. Numa segunda fase, poderá até servir como princípio de novas competições, usando a informação cromática para substituir as informações sensórias tradicionais, tais como infravermelhos.

REFERÊNCIAS

[1] http://microrato.ua.pt

[2] Nuno Lau *et al* (2004) "CIBER-RATO: Uma Competição Robótica Num Ambiente Virtual", in Games2004: Workshop Entretenimento Digital e Jogos Interactivos.

[3] Luís Almeida *et al* (2006) "Micro-Rato Robotics Contest: Technical Problems and Solutions", in Controlo'2006: The 7th Portuguese Conference on Automatic Control.

[4] Will Schroeder et al. (1998) "The Visualization Toolkit – 2nd edition". Prentice Hall.

[5] Lisa S. Ávila et al. (2004) "The *VTK* User's Guide". Kitware.

[6] http://www.vtk.org/pipermail/vtkusers/2007 - January/089212.html

[7] Aulas práticas de MV3D

[8] http://en.wikipedia.org/wiki/Delaunay_triangulation