Jogos de Guerra e Paz

Helder Batista
IST/INESC
Lisboa
hjpb@rnl.ist.utl.pt

Vasco Costa
IST/INESC
Lisboa
vasc@rnl.ist.utl.pt

João Pereira
IST/INESC
Lisboa
jap@inesc.pt

Sumário
A simulação de ambientes virtuais realistas de grande complexidade e interactividade é um problema complexo a vários níveis. Na parte de visualização as dificuldades prendem-se com as limitações do hardware para apresentar uma cena complexa a ritmos interactivos. Na parte das comunicações o desafio coloca-se na gestão de um número muito elevado de participantes mantendo a coerência do ambiente sem sacrificar o desempenho da simulação. Propomos de seguida soluções que procuram tornar possível a realização de simulações de elevado realismo em equipamentos comuns, de baixo custo, disponíveis actualmente no mercado.

Palavras-chave
Simulação, Visualização, HLA, Ambientes Virtuais, Distribuição, Nível de Detalhe Contínuo.

1. INTRODUÇÃO

O desenvolvimento de simulações em larga escala iniciou-se nos meios militares para facilitar o treino e simulação de situações de combate. Exemplo disto são esforços pioneiros como a rede SIMNET [Pope89] e o projecto NPSNET [Macedonia95a, Macedonia95b].

Para a sua realização eram, e ainda são hoje em dia, usados equipamentos de topo de gama em termos de capacidade gráfica e redes dedicadas - que obviamente estão ao alcance apenas de instituições com elevados recursos financeiros.

Vamos usar uma aplicação do tipo simulador de combate aéreo para explorar as soluções disponíveis para resolver os problemas específicos relacionados com este tipo de aplicações. As principais dificuldades a resolver são:

- A visualização de um terreno de grandes dimensões em tempo real.
- A visualização de um grande número de objectos presentes na cena.
- A gestão da participação de um grande número de utilizadores na simulação.

Recentemente foi estabelecido pelo Departamento de Defesa dos E.U.A. (DoD) um novo padrão para a arquitectura em rede de simulações. A *High Level Architecture* (*HLA*) [DMSO98]. Neste momento todas as simulações do DoD são realizadas sobre essa arquitectura e todas as aplicações antigas estão a ser convertidas para o novo sistema.

Neste artigo primeiro descrevemos muito sucintamente a arquitectura da aplicação. Em seguida mostramos em mais detalhe como é efectuada a visualização do mundo. Depois descrevemos em pormenor como foi feita a distribuição dos dados entre os vários utilizadores. Por fim apresentamos as nossas conclusões e ideias para trabalho futuro.

2. ARQUITECTURA DA APLICAÇÃO

A aplicação é composta por vários módulos: (**Figura 1**)

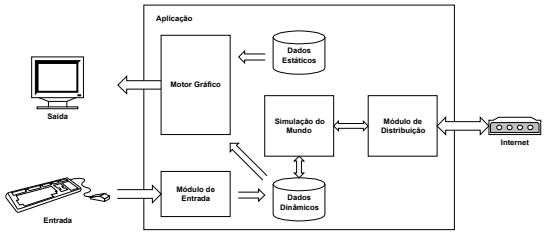

Figura 1 – Arquitectura da aplicação.

Motor Gráfico	responsável pela visualização do mundo e pela detecção de colisões entre os vários objectos.
Módulo de Entrada	interpreta os comandos do utilizador e executa as alterações necessárias no mundo.
Simulador do Mundo	responsável pela simulação de certos fenómenos do mundo como a física newtoniana dos objectos, meteorologia, etc.

Módulo de Distribuição responsável pela distribuição dos dados relevantes para manter a coerência do mundo visto pelos participantes na simulação.

Escolhemos separar os dados estáticos dos dinâmicos já que os dados estáticos são tipicamente mais volumosos. Assim a aplicação é instalada na máquina junto com os dados estáticos e estes não têm de ser transmitidos pela rede. Exemplos de dados estáticos incluem: as malhas de triângulos dos caças e do terreno, texturas, etc. Os dados dinâmicos incluem a posição dos caças, mísseis, etc.

3. VISUALIZAÇÃO

Para a visualização interactiva de cenas complexas o ideal é poder adaptar o nível de detalhe da cena de acordo com alguns parâmetros, como por exemplo, a distância de um objecto ao observador e o poder computacional da máquina.

Existem alguns métodos, designados por de nível de detalhe (*level of detail* ou LOD), que consistem em armazenar várias representações de um mesmo objecto com níveis de complexidade diferentes, usando o nível mais apropriado consoante desejamos mais ou menos detalhe.

Esta aproximação, embora de implementação simples, tem algumas desvantagens, pois conduz ao armazenamento de várias cópias do mesmo objecto com o consequente desperdício de memória.

A transição entre níveis de detalhe é muito brusca e portanto desagradável para o observador (um efeito vulgarmente designado na literatura por *popping*). A sua aplicação a terrenos não é viável devido às dimensões dos dados envolvidos.

Existem algoritmos específicos para a visualização de terrenos de grandes dimensões, tais como algoritmos baseados em árvores quaternárias [Lindstrom96], o ROAM [Duchaineau97] e outros [Garland97], mas não resolvem o problema para os modelos tridimensionais em geral. Um bom resumo do trabalho na área da simplificação de objectos é [Heckbert97].

O algoritmo desenvolvido por Hugues Hoppe, por nós designado como de malhas progressivas (*progressive meshes*) [Hoppe96], foi utilizado no projecto. Este algoritmo tem várias características que o tornam interessante para a realização deste tipo de aplicações:

- Nível de detalhe contínuo (CLOD), ou seja, os modelos podem ser alterados em apenas um vértice de cada vez, dessa forma tornando as mudanças no detalhe mais suaves.
- Pode ser aplicado de forma eficiente tanto a modelos tridimensionais comuns como ao caso específico dos terrenos que, neste tipo de simulação, é uma parte importante quer do ponto de vista do realismo quer do desempenho.
- Suporta refinamento selectivo, ou seja, o aumento ou diminuição do detalhe de zonas particulares do modelo - essencial no caso dos terrenos em que geralmente a área visível é muito inferior às dimensões totais do modelo.
- Subdivisão irregular da malha de polígonos - o que resulta em malhas com menor número de polígonos e com menor erro de aproximação.
- As fases computacionalmente complexas do algoritmo são efectuadas numa fase de pré-processamento o que permite um alto desempenho em tempo-real.
- Suporta a criação de *geomorphs*, ou seja, a interpolação das transformações efectuadas no modelo de forma a suavizar ainda mais os efeitos de *popping*.
- Suporta a preservação de características visuais importantes, como as fronteiras entre materiais.

3.1 Malhas Progressivas

Os detalhes deste algoritmo podem ser consultados em [Hoppe96, Hoppe97 e Hoppe98]. No entanto dado que as variações na implementação podem ter grande influência na qualidade da malha simplificada vamos abordar sucintamente os aspectos mais importantes da implementação.

O algoritmo é aplicado de igual forma tanto ao modelo do terreno como aos restantes modelos na simulação.

Neste algoritmo, um modelo tridimensional é transformado numa árvore binária em que cada nó contém um vértice da malha de polígonos.

No entanto o terreno dispensa o armazenamento de certos atributos como materiais, as normais e as respectivas considerações para a medição do erro das transformações. A árvore do terreno é também muito mais sensível em relação ao uso de memória já que tipicamente o modelo do terreno tem números muito elevados de polígonos (tipicamente mais de 1 milhão de polígonos).

A construção da árvore é efectuada durante a fase de pré-processamento, pelo que o tempo de construção não é crítico, logo é preferível usar uma medida de erro que permita obter malhas de boa qualidade.

Pode facilmente constatar-se que tipicamente não é necessário armazenar todos os vértices originais de um modelo para obter o mesmo nível de detalhe. Esse facto é explorado, sendo possível indicar um nível de erro que é desprezável, dessa forma eliminando vértices e ficando com um modelo que ocupa muito menos espaço que a malha completa sem um sacrifício perceptível da qualidade.

O factor crítico, para a qualidade das malhas de polígonos simplificadas, é a correcta medição da distorção visual introduzida pela eliminação de um vértice. No entanto, para a medição do erro introduzido tem de haver um compromisso entre tempo de execução e complexidade de implementação e fidelidade da medida. Na prática é mais correcto designar esta medição por heu-

rística já que é bastante complexo implementar uma medida que reflicta fielmente a distorção visual introduzida por uma transformação na malha.

A função por nós construída tem a seguinte forma:

$$erro = \sum_i \left[\left(\left\| N(g_i) - N(f_i) \right\| + \frac{\sum_j A(g_j)}{\sum_k A(f_k)} \right) \times A(g_i) \right]$$

$$\forall i : f_i \in F, g_i \in G$$

Onde:

F	é o conjunto das faces originais.
G	é o conjunto das faces modificadas pela transformação.
A(x)	é a área da face *x*.
N(x)	é a normal da face *x*.

Pode dividir-se a fórmula em três partes:

- $\left\| N(g_i) - N(f_i) \right\|$ - Medida da variação da orientação das faces afectadas: contribui para a preservação da aparência da superfície.
- $\frac{\sum_j A(g_j)}{\sum_k A(f_k)}$ - Medida da variação da área: contribui para a preservação do volume do modelo.
- $A(g_i)$ - Peso de cada face: determina que alterações em faces maiores introduzem mais erro

Para os modelos com vários materiais, a fórmula é idêntica, mas os conjuntos de faces são separados por material de forma a preservar as fronteiras de descontinuidade entre faces de materiais diferentes.

Pode ver-se que os resultados desta heurística são aceitáveis, obtendo-se representações com boa qualidade mesmo com um número de polígonos muito inferior ao original, tanto para o terreno como para modelos genéricos (ver **Figura A** em apêndice).

Dado que numa situação normal apenas está visível uma parte do terreno, é natural que apenas se introduza detalhe nessa zona, para isso é necessário usar técnicas de *view frustum culling*, ou seja, testar os vértices do terreno para determinar se estão no volume visível, e ajustar o detalhe da malha de acordo com esse resultado e proporcionalmente à distância do observador (os objectos mais distantes devem ter menos detalhe).

Neste momento não foram efectuadas optimizações neste teste que é uma parte importante para o desempenho. Poderia por exemplo, ser efectuada uma divisão do terreno em quadrículas, determinando-se rapidamente se um conjunto grande de vértices está dentro do volume de visualização, reduzindo consideravelmente o tempo gasto nestes testes [Assarsson99], [Assarsson00].

Para aumentar a performance, repartimos este trabalho por vários quadros (*frames*), reduzindo a carga, especialmente se usarmos um número de polígonos muito elevado. Esta técnica no entanto introduz um certo atraso no refinamento adequado da malha o que pode aumentar um pouco o efeito de *popping,* no entanto, ao permitir usar um número mais elevado de polígonos com mais *performance*, esse efeito é minimizado.

Para obter um ritmo constante, é usado um mecanismo de ajuste dinâmico do erro permitido, de forma a manter um número constante de polígonos na cena.

Em tempo real a lista de vértices e faces activas é a única estrutura actualizada e é muito menor que o número total de vértices do modelo. Desta forma uma das vantagens deste algoritmo é o seu desempenho ser independente do tamanho total do modelo.

A estimativa do valor do erro aceitável é determinada pelo teste que indica se o vértice está visível ou não e pela distância ao observador.

Mesmo com o *hardware* de hoje em dia, o suporte de terrenos exremamente grandes é problemático, pois não é possível mantê-lo completamente em memória.

Para o suporte de terrenos extremamente grandes a estrutura da malha progressiva está desenhada de forma que torna possível a divisão do terreno em zonas, que podem ser construídas separadamente, acelerando o processo de pré-processamento e que podem ser cobertas com mosaicos da textura maior (*texture tiles*), desta forma, ultrapassando as limitações das dimensões máximas da textura da placa gráfica, isto porque as zonas individuais do terreno mantêm a forma quadrada ou rectangular. Em tempo real, estas zonas podem ser carregadas ou retiradas de memória conforme o necessário.

Outra das optimizações efectuadas foi a implementação de *geomorphs*. Este mecanismo consiste na interpolação ao longo do tempo das alterações na malha, desta forma tornando o efeito de *popping* ainda mais difícil de detectar.

Como foi dito anteriormente, o objectivo é a manutenção de um ritmo elevado mesmo quando há muitos objectos visíveis na cena. Portanto o número de triângulos não pode aumentar linearmente com o número de objectos. Também não é conveniente mantê-lo constante pois iria reduzir o desempenho desnecessariamente quando há poucos objectos em cena e ter pouca qualidade com muitos objectos visíveis. Portanto usamos uma heurística para aumentar o número de triângulos de forma não linear. Por cada duplicação do número de objectos visíveis (neste caso os objectos são os caças) aumentamos o número de polígonos em 1/4. Esta aproximação é suficiente para manter uma qualidade gráfica aceitável, já que o detalhe dos caças é também ajustado conforme a sua distância ao observador. A qualidade gráfica global pode ser ajustada mudando o número de triângulos usa-

dos para desenhar um único caça (os triângulos *base*). Esta fórmula pode ser aproximada por:

$$\sqrt[3]{n} \times base; n = \text{número de objectos}$$

3.2 Resultados

Os testes foram efectuados num AMD Duron 800 MHz com uma placa gráfica NVIDIA GeForce 2 MX 32 MB que é neste momento um sistema de baixo custo. O protótipo corre sobre o sistema operativo Windows 2000 utilizando a biblioteca 3D OpenGL [ARB99]. A cena foi visualizada numa janela 800x400.

Os resultados de desempenho apresentados na **Figura 2** são para a navegação num terreno gerado a partir de um mapa de elevação de 2045x2045 pontos representando uma área de 120x120 km^2, que foi dividido em 16 zonas de 512x512 pontos (ver **Figura B** em apêndice).

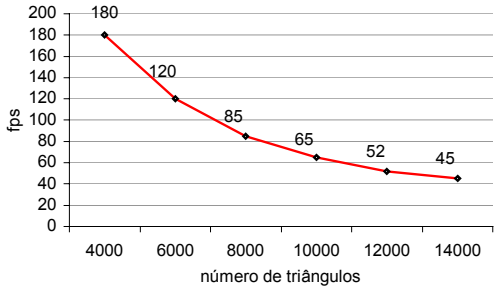

Figura 2 – Variação do desempenho consoante o número de triângulos no terreno.

Na **Figura 3** está um exemplo da progressão do número de polígonos com o aumento do número de caças, usando diferentes número de triângulos *base* (repare-se que a escala do número de caças é logarítmica).

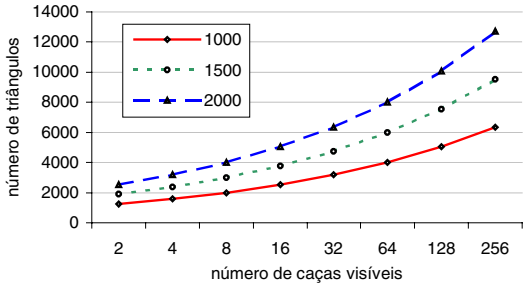

Figura 3 – Variação do número de triângulos consoante o número de caças visíveis.

Na **Figura 4** mostra-se o desempenho numa cena com um terreno com 6000 triângulos e vários números de caças visíveis, com 1500 triângulos *base* (ver **Figura C** em apêndice).

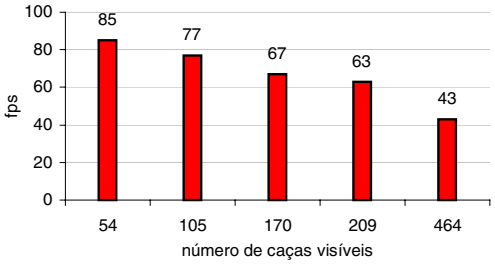

Figura 4 – Variação do desempenho consoante o número de caças visíveis.

4. DISTRIBUIÇÃO

Os problemas de comunicação podem ser resumidos a apenas dois: largura de banda e latência.

Nós desejamos suportar comunicações de longa distância em equipamento de consumo. Temos de suportar os modems de 56Kbps já que são o principal meio de ligação à Internet hoje em dia. As tecnologias de banda larga tais como o ADSL e os modems por cabo ainda não estão suficientemente divulgadas.

Estudos feitos em aplicações multimédia indicam que os seres humanos ficam desconfortáveis com inconsistências na ordem dos 100 ms. Estas inconsistências tornam-se intoleráveis quando na ordem dos 200 ms [Singhal97]. Em ligações transatlânticas sobre a Internet podemos esperar latências entre 200 e 400 ms.

Mesmo que a latência devido ao software e hardware possa ser reduzida estamos limitados pela velocidade da luz quando comunicamos sobre vastas distâncias.

De modo a que os seres humanos não se sintam perturbados é necessário esconder de algum modo a latência induzida pela rede. Também temos de conseguir suportar modems de 56Kbps ao simular centenas de objectos.

Jogos de computador tradicionais como o *Dogfight* e o *DOOM* utilizaram a abordagem ingénua de enviar a posição de todos os objectos simulados uma vez por quadro [Zyda99]. Apesar de esta abordagem funcionar sobre redes locais, devida à elevada largura de banda e baixa latência destas, sobre a Internet apenas a elevada latência é suficiente para arruinar a experiência do utilizador.

Os esforços pioneiros SIMNET e NPSNET aliviaram estes problemas com a introdução de técnicas como a estimativa (*dead-reckoning*) [Aronson97, Zyda99] e a difusão em grupo (*multicasting*) [Deering89].

4.1 Difusão em Grupo

A difusão em grupo (*multicasting*) é utilizada em comunicações entre grupos de máquinas. Cada máquina encontra-se registada num grupo de difusão que possui um dado endereço IP. Os encaminhadores (*routers*) da rede encarregam-se de entregar aos vários membros os pacotes endereçados ao grupo. Utilizando a difusão em grupo podemos reduzir o tráfego necessário para comunicar um evento a várias máquinas.

4.2 Estimativa

As técnicas de estimativa (*dead-reckoning*) trocam alguma consistência na simulação para conseguir fingir uma latência mais baixa e reduzir os requisitos de largura de banda. Elas conseguem atingir estes resultados utilizando predição. Os antigos marinheiros conseguiam empiricamente localizar com algum grau de precisão a sua posição num mapa dado o ponto de partida, a orientação do navio (utilizando uma bússola), a velocidade (medida em nós) e o tempo passado desde que saíram do porto. Utilizando técnicas semelhantes podemos prever a localização de outros objectos. No nosso caso dos caças e dos mísseis. Para isso basta lembrarmo-nos da física elementar:

$$\vec{x} = \vec{x_0} + \vec{v_0}t + \frac{1}{2}\vec{a}t^2$$

Esta equação como é sabido permite descrever o movimento de um corpo com aceleração constante. Sabendo os dados da posição inicial, a velocidade inicial, a aceleração e o tempo decorrido podemos saber onde nos encontramos num dado momento.

A utilização de um protocolo de estimativa permite maiores ganhos em objectos e eventos mais previsíveis. Quando um objecto é completamente controlado por um humano, o seu comportamento é eventualmente imprevisível. No entanto pode usar-se vária informação para prever mais fielmente o comportamento futuro de um objecto. Por exemplo, num caça há limites dinâmicos que têm de ser respeitados, por isso pode ser relativamente fácil prever a sua posição mesmo quando controlado por um ser humano.

4.3 HLA

A HLA [DMSO98] define o modo como deve ser realizada uma simulação desde a documentação dos dados até à implementação e forma de comunicação entre os utilizadores. A HLA define também uma biblioteca (libRTI) [DMSO00] que suporta a construção e operação de simulações distribuídas. Esta biblioteca está num nível de abstracção acima dos protocolos de comunicação usados e está disponível para várias linguagens de programação.

É definido um modelo de objectos e um modelo de interacção comum a todos os participantes (Federados) numa simulação (Federação), o que possibilita a troca de informação de forma uniforme.

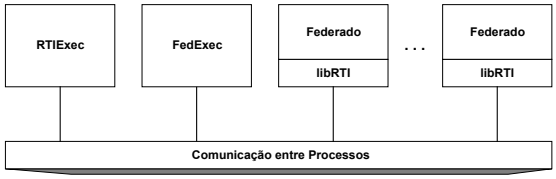

Figura 5 – Componentes da RTI.

A interface de comunicação do HLA (*Run-Time Interface* ou RTI) (ver **Figura 5**) faz com que uma simulação se apresente a um participante como uma entidade única, embora se encontre distribuída pelos vários participantes, não sendo necessária nenhuma componente centralizada. Aliás a centralização é desencorajada.

Para efectuar com eficiência a comunicação de mensagens entre vários participantes a RTI pode usar difusão em grupo.

4.4 Implementação

Decidimos aplicar carimbos temporais às actualizações da posição tal como no protocolo PHBDR [Singhal95, Singhal97]. Isto reduz o erro entre a posição real e a posição mostrada remotamente mas requer que os computadores tenham os relógios sincronizados. Felizmente é fácil de cumprir este requisito utilizando o *Network Time Protocol* (NTP) [Mills92]. O sistema operativo Windows 2000 possui de origem uma implementação do *Simple Network Time Protocol* (SNTP) [Mills96] que é adequada às nossas necessidades.

Para reduzir o erro é enviado um pacote sempre que a distância entre a posição real e a posição prevista exceda um determinado valor. Quanto mais elevado for o valor escolhido, menos pacotes serão enviados pela rede, mas valores elevados causam movimentos de correcção da trajectória mais bruscos e portanto mais desagradáveis.

Para suavizar este movimento movemos o objecto suavemente da posição anterior para a nova posição utilizando um algoritmo de convergência. Foi implementada uma forma de convergência linear para suavizar o movimento.

A informação da orientação do objecto é enviada junto com a informação da posição.

4.5 Resultados

Testamos o protocolo de estimativa por nós desenvolvido na trajectória de um caça representada na **Figura 6**.

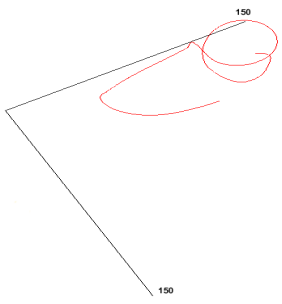

Figura 6 – Trajectória de um caça.

O caça esteve parado aproximadamente durante 1 segundo antes de iniciar o movimento.

O número de pacotes enviados utilizando a estimativa é uma ordem de magnitude inferior ao número de pacotes enviados com a abordagem ingénua como podemos ver na **Figura 7**. Todavia o erro na posição do caça vista remotamente é ligeiramente superior visto que apenas enviamos um pacote quando a posição actual e a posição prevista diferem mais que um certo valor que não é nulo. Este problema pode ser observado na **Figura 8.1**.

A convergência linear também aumenta o erro visto que o caça demora mais tempo a voltar à trajectória real como pode ser visto na **Figura 8.2**.

A introdução de latência como esperado também aumenta o erro. Todavia uma vez que o pacote com a actualização seja recebido o caça volta à trajectória real como pode ser visto na **Figura 8.3**. O erro máximo nesta trajectória é ligeiramente inferior a 7 metros.

Quando a latência é elevada o erro observado é proporcional à velocidade do caça e à latência da rede.

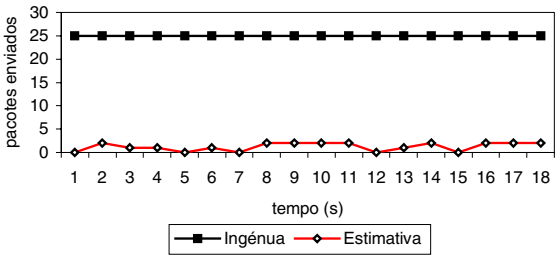

Figura 7 - Número de pacotes enviados utilizando a abordagem da estimativa vs. a abordagem ingénua.

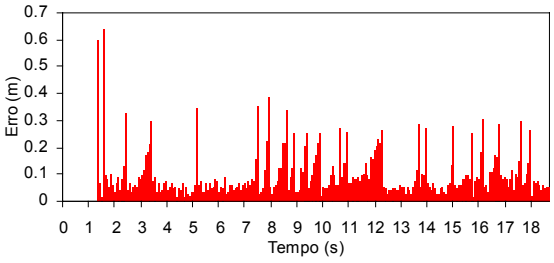

Figura 8.1 - Erro com latência de 0 ms e sem convergência.

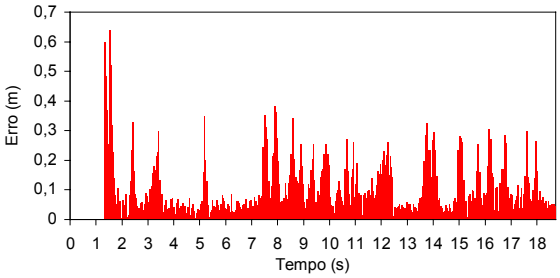

Figura 8.2 - Erro com latência de 0 ms e convergência linear.

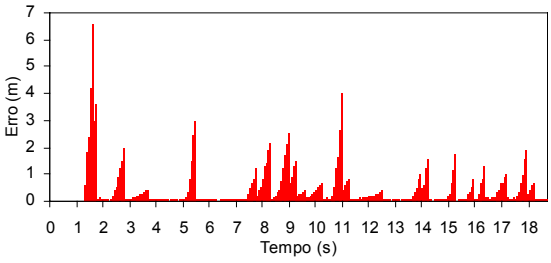

Figura 8.3 - Erro com latência aleatória de 200 a 400 ms e sem convergência.

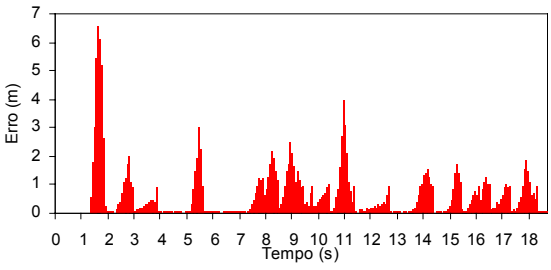

Figura 8.4 - Erro com latência aleatória de 200 a 400 ms e convergência linear.

5. CONCLUSÕES E TRABALHO FUTURO

Os testes demonstram que, utilizando os algoritmos apropriados, o *hardware* actual tem capacidade suficiente para suportar a simulação de ambientes virtuais de grandes dimensões com qualidade gráfica adequada.

A utilização da HLA facilita muito o desenvolvimento deste tipo de aplicações.

A utilização combinada de difusão em grupo e algoritmos de previsão, reduz muito a largura de banda requerida e consegue reduzir a latência percepcionada pelos utilizadores.

No entanto uma dificuldade persiste, a elevada latência das comunicações na Internet [Cheshire96] impõe limitações na fiabilidade que se consegue obter, já que mesmo usando algoritmos de previsão, é impossível esconder atrasos elevados na comunicação, pois as acções de utilizadores remotos são imprevisíveis mesmo estando constrangidas por limites físicos dos objectos simulados.

No futuro, é ainda necessário efectuar testes da aplicação com um número elevado de máquinas distribuídas geográficamente.

6. AGRADECIMENTOS

Agradecemos ao INESC os meios disponibilizados para a realização deste projecto.

7. REFERÊNCIAS

[ARB99] OpenGL Architecture Review Board. OpenGL® 1.2 Programming Guide, 3rd edition. Addison Wesley, 1999.

[Aronson97] Aronson, J. Dead-Reckoning: Latency Hiding for Networked Games. Gamasutra. September 1997.
http://www.gamasutra.com/features/19970919/aronson_01.htm

[Assarsson99] Assarsson, U. and Möller, T. Optimized View Frustum Culling Algorithms. Technical Report 99-3, Department of Computer Engineering, Chalmers University of Technology, March 1999.

[Assarsson00] Assarsson, U. and Möller, T. Optimized View Frustum Culling Algorithms for Bounding Boxes. Journal of Graphics Tools 5(1):9-22, 2000.

[Cheshire96] Cheshire, S. Latency and the Quest for Interactivity. November 1996.
http://www.stuartcheshire.org

[Deering89] Deering, S. E. Host extensions for IP multicasting. RFC1112, Aug-01-1989.

[DMSO98] Defense Modeling and Simulation Office. High Level Architecture Interface Specification, Version 1.3. U.S. Department of Defense, April 1998.
http://hla.dmso.mil

[DMSO00] Defense Modeling and Simulation Office. HLA RTI 1.3-Next Generation Programmer's Guide Version 3.2. U.S. Department of Defense, September 2000.

[Duchaineau97] Duchaineau, M. et. all. ROAMing Terrain: Real-time Optimally Adapting Meshes. IEEE Visualization 1997, 81-88.

[Garland97] Garland, M. and Heckbert, P. Fast Triangular Approximation of Terrains and Height Fields. Carnegie Mellon University, May 2 1997.

[Heckbert97] Heckbert, P. and Garland, M. Survey of Polygonal Surface Simplification Algorithms. Carnegie Mellon University, May 1 1997.

[Hoppe96] Hoppe, H. Progressive meshes. Proceedings of SIGGRAPH 1996, 99-108.

[Hoppe97] Hoppe, H. View-dependent refinement of progressive meshes. Proceedings of SIGGRAPH 1997, 189-198.

[Hoppe98] Hoppe, H. View-dependent level-of-detail control and it's application to terrain rendering. IEEE Visualization 1998, 35-42.

[Lindstrom96] Lindstrom, P. et. all. Real-time, continuous level of detail rendering of height fields. Proceedings of SIGGRAPH 1996, 109-118.

[Macedonia95a] Macedonia, M. A Network Software Architecture for Large Scale Virtual Environments. Doctor's Thesis, Naval Postgraduate School, June 1995.

[Macedonia95b] Macedonia, M., Brutzman D., Zyda, M. et. all. NPSNET: A Multi-Player 3D Virtual Environment over the Internet. Proceedings of the ACM 1995 Symposium on Interactive 3D Graphics, 9-12 April 1995.

[Mills92] Mills, D. Network Time Protocol (Version 3) specification, implementation and analysis. RFC1305. University of Delaware. March 1992.
http://www.eecis.udel.edu/~mills/ntp.htm

[Mills96] Mills, D. Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. RFC2030. University of Delaware. October 1996.

[Pope89] Arthur R. Pope. The SIMNET network and protocols. Technical Report 7102, BBN Systems and Technologies, Cambridge, MA.

[Singhal95] Singhal, S. and Cheriton, D. Exploiting position history for efficient remote rendering in networked virtual reality. PRESENCE: Teleoperators and Virtual Environments 4(2):169-193, Spring 1995.

[Singhal97] Singhal, S. Effective Remote Modeling in Large-Scale Distributed Simulation and Visualization Environments. *Doctor's Thesis*, Stanford University, 1997.

[Zyda99] Singhal, S. and Zyda, M. Networked Virtual Environments: Design and Implementation. ACM Press and Addison Wesley, 1999.

8. APÊNDICE

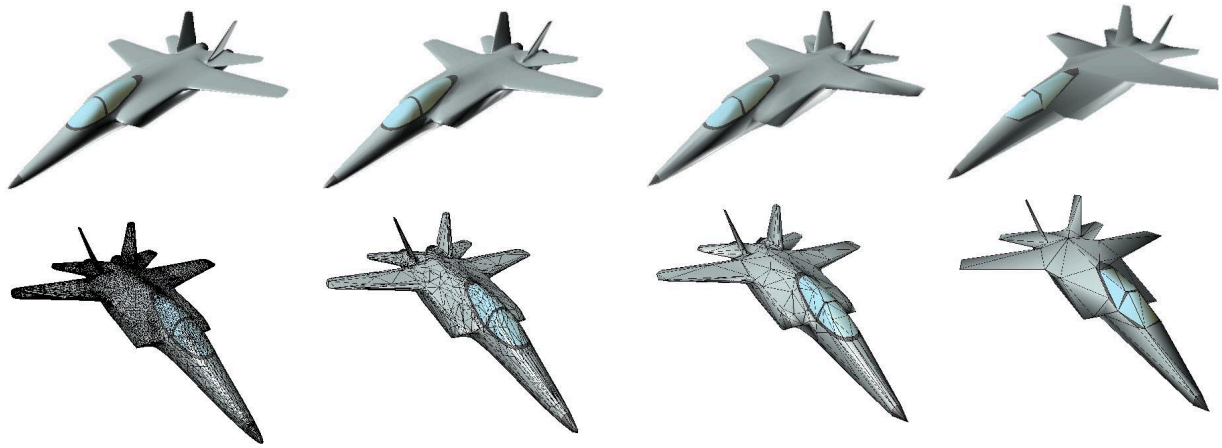

Figura A - Da esquerda para a direita: modelo original (30860 faces); simplificado até 3160 faces; simplificado até 800 faces; simplificado até 200 faces.

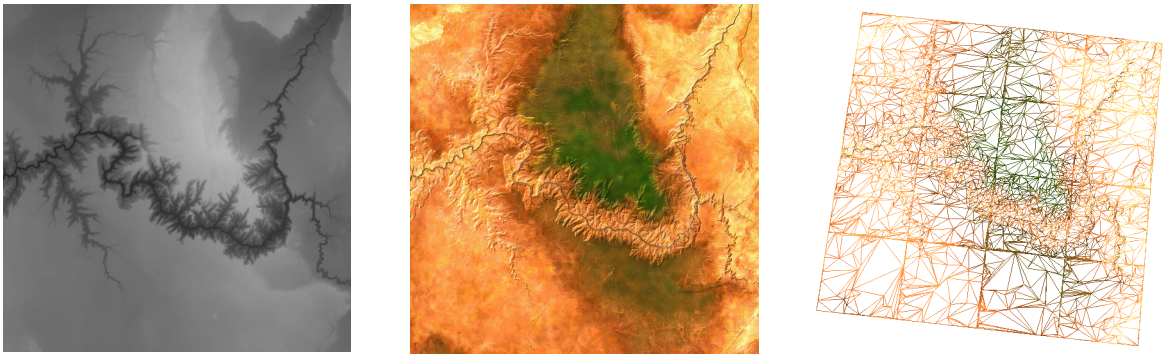

Figura B - Terreno com 2045x2045 pontos. Da esquerda para a direita: dados de elevação; textura; malha com 6000 triângulos.

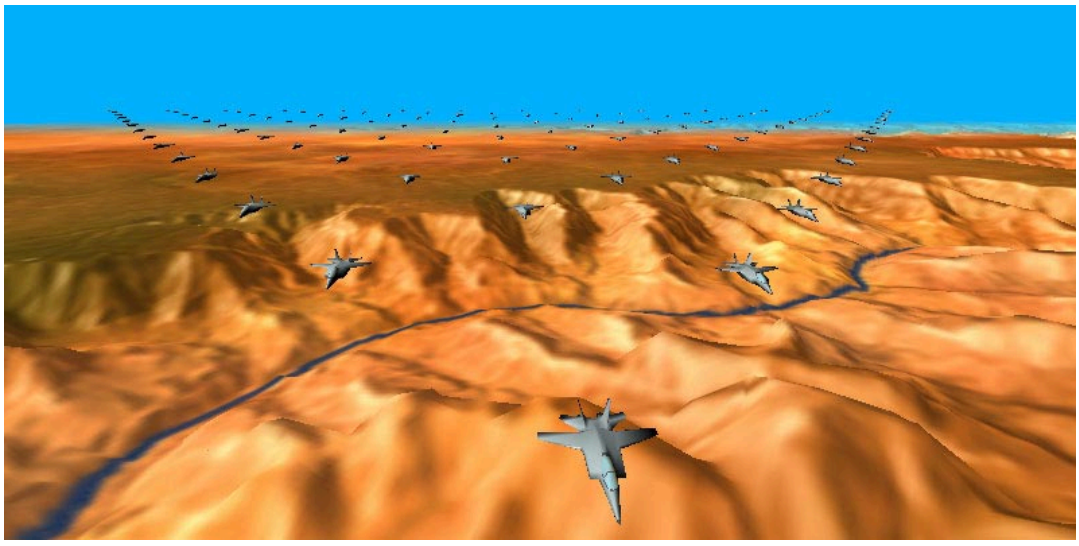

Figura C - Cena com 105 caças.