1 Introdução

I know because I must know. It's my purpose. It's the reason I'm here. (The Matrix)

1.1 Objetivos do trabalho

Os *hardwares* gráficos atualmente podem ser considerados como verdadeiros processadores e, em muitos casos, mais poderosos que os processadores da CPU. Algumas placas comerciais ultrapassaram a barreira dos 120 milhões de transistores (mais do que um chip Pentium IV), podem acessar até 16 GB de memória, processam 4 ou mais *pixels* simultaneamente e as placas com suporte à programação de seu *pipeline* podem possuir vários processadores programáveis (ver capítulo 6). É por esta razão que a tendência hoje é de chamar estes dispositivos de GPU's (*Graphic Processor Units*). Assim, pode-se assumir que um computador munido de uma boa placa gráfica é uma máquina paralela, embora um dos processadores seja de uso dedicado a aplicações gráficas (GPU) e o outro seja de uso genérico (CPU).

Pode-se comprovar, no entanto, que nas aplicações gráficas cada vez mais têm crescido a tendência de que a GPU assuma a maior parte do *pipeline* de visualização, deixando por outro lado que a CPU tenha um tempo ocioso (*idle time*) cada vez maior. Na realidade, o trabalho gráfico da CPU tem se tornado cada vez mais burocrático e limitado a alimentar a placa gráfica com os dados relacionados às geometria e texturas e, quando muito, a resolver alguns cálculos de otimização e *culling*. A figura 1.1 mostra o baixo consumo de tempo do processamento de uma CPU na execução de um jogo 3D, dotado de extensa modelagem geométrica. Diversas aplicações têm-se aproveitado deste tempo livre procurando, com este recurso, resolver operações mais complexas, tais como cálculos de inteligência artificial ou simulação física. Entretanto, poucos trabalhos têm sido feitos no sentido de assumir a CPU|GPU como uma máquina paralela para um mesmo *pipeline* gráfico.

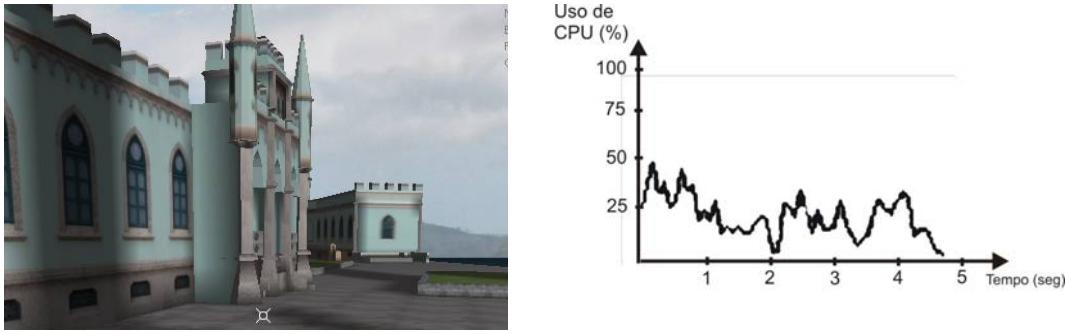

figura 1.1 – Consumo de CPU (em %) em movimentos típicos de um jogo 3D, utilizando um computador munido de GPU. O cenário utilizado possui 14.108 polígonos, sendo que 4.540 estão dentro do campo de visão.

A primeira idéia neste trabalho consiste em aproveitar o tempo ocioso da CPU para operações gráficas em tempo real, realizando operações que são inapropriadas para a GPU sem, no entanto comprometer o desempenho de visualização da aplicação. Estas operações consistem sobretudo em realizar processamento de imagens típicas da área de renderização baseada em imagens, alimentando o *pipeline* da placa gráfica com os resultados obtidos. Assim sendo, tem-se um *pipeline* gráfico duplo: um que é executado pela placa gráfica (*hardware*) e outro que é executado pela CPU (*software*). Este tipo de arquitetura por si só já corresponde, na prática, a ver um sistema com uma CPU e uma GPU como sendo uma máquina paralela. Nesta máquina paralela, a arquitetura proposta toma os cuidados apropriados para que a GPU nunca tenha outra latência, a não ser aquelas geradas pela sua própria limitação de processamento.

Neste sentido, um dos objetivos deste trabalho consiste em apontar que os métodos de renderização baseada em imagens podem ser utilizados para reutilização de resultados obtidos de uma visualização previamente realizada pela CPU ou pela GPU. Isto é feito através de uma extensão do conceito de impostores, primeiramente definido em (Schaufler, 1997) e que consiste num *sprite*, gerado a partir de uma primitiva geométrica, com suporte a profundidade. O nome “impostor com relevo” indica que este tipo de primitiva é um impostor, cuja profundidade serve para se realizar uma operação de *warping* sobre a textura, utilizando o método de mapeamento de texturas com relevo. O método básico para texturas com relevo foi primeiramente apresentado em (Oliveira, 2000a). A figura 1.2 ilustra um objeto modelado como impostor com relevo.

figura 1.2 – Exemplo de um objeto modelado através de impostores com relevo.

Uma vez que as operações de renderização baseada em imagem não são apropriadas para serem processadas pela GPU, o método de *pipeline* híbrido proposto sugere que o processamento da função plenóptica seja na CPU, deixando o *hardware* gráfico dedicado às operações de visualização dos dados geométricos. Esporadicamente o sistema permite que haja uma transferência dos resultados entre a memória da CPU para a memória da GPU. Isto é realizado de forma inteligente para evitar que o gargalo da visualização esteja ou no processamento da aplicação de renderização baseada em imagens ou no processo de tráfego de dados CPU/ GPU.

Um outro objetivo deste trabalho consiste em realizar abordagens paralelas para resolver o mapeamento de texturas com relevo. Para tanto, é explorado o recurso de programação por *threads*, permitindo que haja processos dedicados a certas tarefas, dando-lhes diversos graus de prioridade. São apresentadas diversas abordagens envolvendo paralelismo entre CPU-GPU e paralelismo de recursos de CPU. De forma a comprovar que este paralelismo é um recurso acessível e barato, toda a implementação é realizada em processadores com *hyper-threading*, uma vez que este recurso está se tornando barato e comum no mercado de micro-computadores.

Impostores com relevo, a contribuição inédita desta tese, possibilita que a técnica inventada por Oliveira (2000 a) modele qualquer tipo de objeto geométrico – um resultado de grande impacto em aplicações de tempo real.

1.2 Conceitos Envolvidos

Os conceitos que são tratados nesta dissertação referem-se a:

a) Renderização baseada em imagens - área da computação gráfica que procura criar objetos e cenários utilizando apenas imagens;

b) *Pipeline* de visualização em tempo real, dando-se especial enfoque a aplicações de jogos 3D;

c) Aceleração gráfica por *hardware*, bem como programação para GPU´s;

d) Processamento paralelo e distribuído para o *pipeline* de visualização.

1.3 Estrutura da dissertação

O capítulo 2 introduz primeiramente o conceito da função plenóptica, para em seguida resumir os principais trabalhos e pesquisas na área de renderização baseada em imagens. Para descrevê-los, cria-se uma classificação separada por aplicações que envolvem modelagem de cenários, modelagem de panoramas, modelagem de objetos e métodos de aceleração e otimização para visualização de estruturas geométricas. A seguir, apresenta-se com detalhes o conceito de 3D *image warping*, conceito este que é utilizado na elaboração dos impostores com relevo. O conceito de *view morphing* também é apresentado com detalhes porque, apesar de não ter sido utilizado no desenvolvimento desta pesquisa, aponta-se que este conceito também pode ser explorado para chegar a resultados semelhantes.

A proposta de modelagem deste trabalho se enquadra dentro da categoria de modelagem de objetos por imagens. Desta forma, o capítulo 3 aprofunda-se nas diversas técnicas de renderização baseada em imagens que solucionam esta classe de problemas. O mapeamento de texturas com relevo é detalhadamente exposto e discutido ainda neste capítulo, uma vez que o conceito de impostores com relevo é uma extensão deste trabalho.

O capítulo 4 propõem o conceito de Impostores com relevo, expondo como é adaptado partindo das texturas com relevo.

Os impostores com relevo são texturas com relevo dinamicamente atualizáveis: quando uma delas se torna obsoleta, gera-se, através de um processo paralelo, uma nova textura para substituí-la. O capítulo 5 apresenta um critério capaz de indicar quando esta troca deve ocorrer. Neste capítulo também se apresenta uma possível otimização no processamento do mapeamento de relevo, utilizando a equação de Schaufler.

O capítulo 6, denominado de “A GPU” apresenta o papel do *hardware* gráfico no processo de visualização dos impostores com relevo, que basicamente consiste em tratar a iluminação para cada *pixel*, utilizando o mapa de normais da textura. Nesta parte da dissertação também se discute a conveniência de se utilizar a aceleração por *hardware* para aplicações de renderização baseada em imagens. Finalmente, ainda neste capítulo, apresenta-se uma implementação do mapeamento de texturas com relevo, feita totalmente na GPU. Esta abordagem é posteriormente comparada com o desempenho da proposta deste trabalho.

No capítulo 7, após uma breve introdução aos conceitos de sistemas paralelos e distribuídos, apresentam-se as propostas de paralelismo que são utilizadas para a implementação do *framework*.

O capítulo 8 introduz detalhadamente este *framework*, bem como todas as etapas de implementação realizadas. Ainda neste capítulo podem-se encontrar os resultados obtidos pelas diversas propostas da tese, especialmente em relação aos impostores com relevo.

Finalmente, o capítulo 9, após breves conclusões, discute sobre diversas possibilidades para se estender esta pesquisa.

1.4 Contribuições Alcançadas

A principal contribuição original desta tese é a criação do conceito de Impostores com Relevo, como uma extensão do mapeamento de texturas com relevo de Oliveira (2000). Este novo conceito aumenta a capacidade de representação do mapeamento de texturas com relevo, tornando-o capaz de modelar qualquer tipo de objeto geométrico.

A paralelização da renderização entre GPU e CPU é uma outra contribuição original, porém dividida com Fonseca (2004) que realiza suas pesquisas em parceria com o presente autor. Uma contribuição do presente trabalho está no processamento paralelo para renderização do impostor, enquanto que em Fonseca (2004) o impostor já vem pronto.

Duas outras contribuições originais são as seguintes:

- Processo de otimização de *warping* em texturas com relevo, através do critério de Schaufler (1995);

- Proposta de taxonomia para os métodos de renderização baseada em imagens e suas principais técnicas.

- Uma heurística capaz de apontar quando uma textura com relevo não é mais válida, e portanto convém ser substituída por uma nova.