2º Encontro Português de Computação Gráfica

Porto, Outubro de 1989

Experiências com *Ray Tracing*
Necessidade de Arquitecturas Paralelas

A. Cardoso Costa, INESC.Norte
A. Augusto Sousa, DEEC.FEUP/INESC.Norte
F. Nunes Ferreira, DEEC.FEUP/INESC.Norte

RESUMO

O *Ray Tracing* e a *Radiosity* são as técnicas de síntese de imagens que, na actualidade, produzem imagens com maior grau de realismo. Um sistema CAD dotado com estas técnicas poderá ser de grande utilidade numa empresa, quer no apoio ao *Marketing* quer na antevisão de novos modelos.

Põem-se alguns problemas de ordem prática na aplicação desta ideia. Por um lado, os tempos de processamento ainda são demasiado longos e, por outro lado, o realismo desejável nas aplicações de marketing ainda não é obtido facilmente (problemas de texturas, iluminação, etc...).

Pensamos que os algoritmos de *Ray Tracing* e *Radiosity* poderão ser paralelizados com vista à diminuição dos tempos de processamento. Interessa pois definir as estratégias possíveis na implementação desse paralelismo, tanto ao nível de software como ao nível das arquitecturas paralelas especilizadas.

1. INTRODUÇÃO

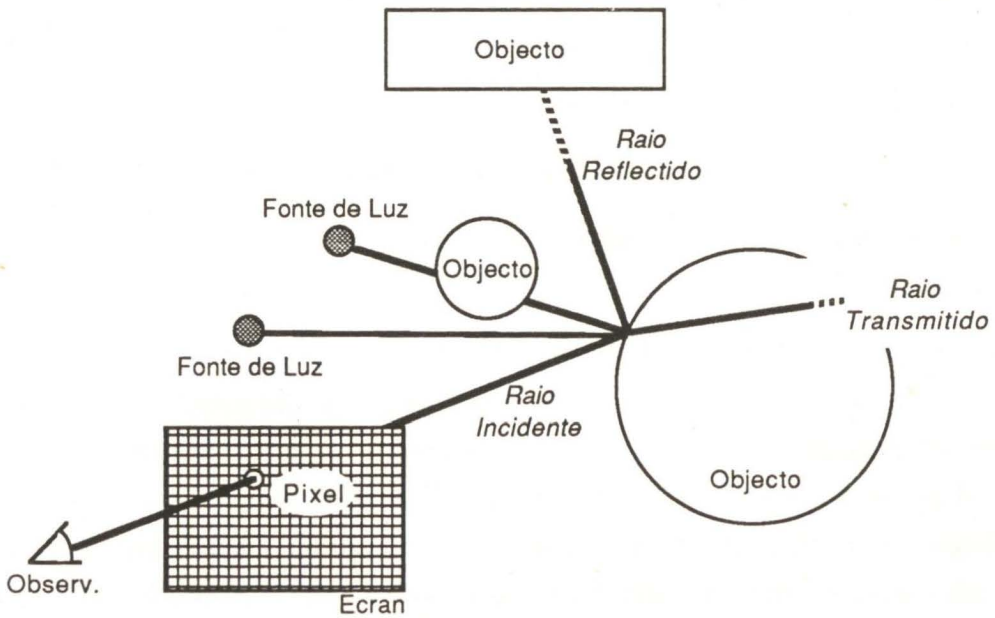
A síntese de imagens foi durante alguns anos baseada em algoritmos de visibilidade que careciam de realismo nas imagens produzidas, mas que computacionalmente eram rápidos (década de 70). A necessidade de aumentar o realismo dessas imagens levou ao aparecimento do *Ray Casting* [APELL68], que ultrapassava algumas das limitações principais dos referidos algoritmos, mas não solucionava totalmente o problema.

Whitted [WHITTED80] alargou esta noção, passando o método a designar-se *Ray Tracing*. Este algoritmo aumentou consideravelmente o realismo das imagens produzidas, embora fossem desde o início conhecidas as suas limitações, quer em termos de modelo, quer em termos de requisitos computacionais (tempos de cálculo muito elevados).

O nosso interesse neste algoritmo reside na sua utilização em sistemas CAD, como apoio ao *marketing*, publicidade, etc, efectuando a síntese de imagens destinadas à produção de catálogos e afins, filmes diversos, etc.

O método *Ray Tracing* utiliza os princípios geométricos da óptica. Para cada pixel no ecran, é traçado um raio luminoso a partir do ponto de visão através do pixel para dentro da cena. O raio é testado contra cada um de todos os objectos em cena e a intersecção com o objecto mais próximo (se existir) é usada para calcular a intensidade do pixel no ecran.

A partir do ponto de intersecção, gera-se um novo raio em direcção a cada uma das fontes de luz (consideradas pontuais). Se o raio encontra um segundo objecto no seu percurso, então aquele ponto do primeiro objecto encontra-se na sombra, senão é calculada a sua intensidade.



Ainda a partir do mesmo ponto podem gerar-se dois novos raios, um por reflexão e outro por transmissão de luz (este último caso, se o objecto não é opaco). Cada um destes novos raios sofre então um tratamento idêntico ao primeiro e, recursivamente, vão-se gerando cada vez mais raios. A cor de cada raio luminoso é determinada a partir das cores dos seus dois "filhos", dos raios enviados

directamente para as fontes luminosas e da iluminação ambiente calculada no ponto de intersecção.

Este processo constrói uma árvore de raios para cada pixel e termina quando se conclui que o seu efeito sobre o pixel no ecran é desprezável ou quando a altura da árvore atinge o seu valor máximo.

Dado que é um assunto actual e que ainda justifica um esforço importante de investigação, implementou-se no INESC.Norte uma versão do algoritmo *Ray Tracing*, afim de se obter experiência neste domínio e de se possuir um banco de ensaios para se testarem algumas ideias novas, tendo em vista a redução dos tempos de cálculo e o aumento do realismo das imagens. Neste trabalho descreve-se sucintamente esse trabalho, relacionando as várias técnicas implementadas com o *state-of-art*.

De futuro, o nosso trabalho de investigação continuará, introduzindo várias extensões/inovações, nomeadamente, modelos ópticos mais sofisticados, tratamento de texturas, processamento paralelo (arquitecturas especiais), etc.

2. *RAY TRACING* - SITUAÇÃO ACTUAL

Da explicação anterior fica claro que o *Ray Tracing* é uma aproximação da realidade e como tal sofre de várias limitações.

* *Modelo* - pelo facto dos raios serem imaginados com um percurso invertido em relação à realidade, só são examinados alguns raios luminosos emitidos pelas fontes de luz. O efeito especular na reflexão ou transmissão de luz não sofre muito com este problema pelo que é razoavelmente aproximado. No entanto, seria impraticável modelar o efeito de difusão lançando um número muito vasto de raios reflectidos (ou transmitidos) por cada raio incidente, em virtude do enorme peso computacional que isso acarretaria. Assim, o efeito de difusão é quase sempre simplificado multiplicando uma reflexão ideal por um factor de difusão. O *Ray Tracing* distribuído [COOK84] contorna o problema lançando um pequeno número de raios nas direcções mais prováveis em função das características de difusão da superfície.

* *Aliasing* - é um dos principais problemas em Computação Gráfica. Existem algumas técnicas sofisticadas para reduzir ou eliminar o *aliasing* em *Ray Tracing*, embora a maior parte seja computacionalmente exigente; por exemplo, o *Ray Tracing* distribuído [COOK84] atribui ao pixel uma iluminação calculada segundo uma média pesada dos raios lançados; o *Jittered Importance Supersampling* [COOK86] vai lançando raios de acordo com as características da superfície até que a variância de côr seja mínima.

* *Tempo de Cálculo* - é conhecida a fama que o *Ray Tracing* possui como gastador de tempo de CPU. Dado que a maior parte do tempo é consumido a determinar possíveis intersecções de raios com objectos e que é necessário utilizar vírgula flutuante em precisão dupla para se produzirem imagens de razoável qualidade, torna-se essencial recorrer a técnicas que permitam reduzir ao máximo esses cálculos intensivos, quer por diminuição do número de testes de intersecção a efectuar, quer por aceleração dos mesmos, recorrendo a testes preliminares mais simples. Entre as técnicas utilizadas para este último caso salientam-se:

-Volumes envolventes (*enclosing boxes*) [KAY86];
-Organização hierárquica dos volumes envolventes [KAY86];
-*Shadow caching* [MARK88];
-Filas de prioridade [KAY86].

3. EXPERIENCIA ADQUIRIDA

Um algoritmo de *Ray Tracing* em C (sequencial) tem vindo a ser desenvolvido e corre em algumas máquinas (VMS e UNIX). É uma versão *Backward Ray Tracing*, com câmara tipo *Pin-Hole* e inclui *anti-aliasing* por superamostragem adaptativa do pixel. Com este algoritmo têm sido testadas algumas ideias no sentido de aumentar a qualidade das imagens: determinação de sombra/luz tem em conta a transparência de objectos no caminho para as fontes de luz; os tempos de cálculo têm sido alvo de melhorias significativas (alguns minutos para uma cena simples, com poucas luzes e objectos opacos), mas ainda estão longe do que seria desejável em certas aplicações.

Algumas soluções têm sido testadas para ultrapassar algumas das limitações mais importantes do *Ray Tracing*. Entre as soluções adoptadas temos:

* *Modelo de Iluminação* - embora o modelo usado seja relativamente simples, procurou-se introduzir alguma sofisticação que o tornasse mais realista:

-Factores de especularidade e de difusão decompostos em três componentes RGB;

-Decomposição em RGB do factor de influência dos raios sobre o pixel; *Shading* adaptativo (a geração de novos raios reflectidos ou refractados prossegue enquanto houver um factor de influência RGB superior a um valor pré-definido, o que conduz a uma árvore de *shading* de altura variável);

-Fontes de luz pontuais orientáveis, com ângulo sólido de irradiação e com controle do gradiente da transição luz/sombra;

-Influência luminosa de uma fonte de luz sobre um ponto de uma superfície dependente da distância entre ambos;

-O efeito das fontes de luz leva em consideração eventuais objectos não totalmente opacos que estejam no percurso entre o ponto a colorir na superfície de um objecto e a fonte luminosa (embora os efeitos de possíveis refracções não sejam considerados, porque tal é impossível numa aproximação determinística).

* *Aliasing* - no presente trabalho foi escolhida a superamostragem adaptativa do pixel que, com modificações posteriores, tem produzido resultados rápidos e bons:

-O pixel é considerado quadrado e é lançado um raio para cada um dos seus cantos; se não existirem "grandes" diferenças entre as cores determinadas para aqueles pontos, a côr do pixel é a média das quatro; de outro modo, o pixel é sub-dividido em quatro sub-pixeis quadrados e são lançados os raios correspondentes aos novos pontos assim criados; o processo repete-se recursivamente.

* *Tempo de Processamento* - estão implementados vários esquemas de aceleração baseados em testes preliminares (estes testes não eliminam sempre a necessidade de testes específicos de intersecção para cada objecto, mas como são simples e rápidos não aumentam praticamente nada o tempo de cálculo):

-Exclusão de objectos cujo volume envolvente se encontre totalmente fora de um octante que é definido em função do raio luminoso em questão;

-Teste de intersecção com os volumes envolventes hierarquizados (dado que o volume envolvente é um paralelipípedo de faces alinhadas com os eixos coordenados, este teste é bastante simples e eficiente);

-Filas de prioridade para as intersecções com os volumes envolventes que permitem reduzir o número de testes específicos de intersecção (efectuam-se primeiro todos os testes de intersecção com volumes envolventes e cria-se uma fila de prioridade ordenada por valores crescentes de distância; de acordo com a ordem dessa fila, testam-se as intersecções com os objectos propriamente ditos até que se obtenha uma distância de intersecção inferior à distância ao próximo volume envolvente ou até que se atinja o fim da fila);

-*Shadow caching* das fontes de luz (para cada fonte de luz e para cada nível de *shading*, armazenam-se os objectos que provocam sombra, de modo que em futuros testes de intersecção com as fontes de luz se consideram primeiro esses objectos);

-Superamostragem adaptativa acelerada (o lançamento de novos raios para dentro dos subpixels só é efectuado em relação aos objectos previamente intersectados, e não para todos os objectos existentes; embora seja uma simplificação que em alguns casos pode falsear a realidade, a coerência entre pixels implica que em praticamente 100% dos casos se comporte correctamente).

Um outro aspecto inovador deste trabalho foi a introdução de Superfícies Bicúbicas [COSTA88], para além dos objectos tradicionais (Esfera, Paralelipípedo, Cone, Cilindro, Polígono, etc). Dado que a formulação analítica destas superfícies é complexa, foi necessário desenvolver métodos especiais para a determinação de intersecções e

para a determinação da normal à superfície. Estes métodos são uma mistura de processos iterativos com heurísticas, afim de se conseguir um compromisso em termos de precisão do resultado versus tempo de cálculo.

4. OUTRAS VIAS DE INVESTIGAÇÃO

O *Ray Tracing* é um método de síntese de imagens com capacidade de representação com um nível de realismo quase fotográfico. Os tempos de processamento são no entanto muito elevados.

4.1. Ray Tracing com Realismo Crescente

Dependendo da aplicação de um sistema CAD, o nível de realismo exigido na síntese de imagens pode variar desde o muito elevado ao médio/baixo. Algumas aplicações exigem mesmo um bom nível de realismo e, simultaneamente, um razoável grau de interactividade.

O *Ray Tracing* é um grande consumidor de tempo de processamento. A imagem é sintetizada tratando cada ponto do ecran exaustivamente, ou seja, até ao seu máximo nível de realismo, ponto por ponto, linha por linha. Deste modo, o utilizador só tem uma imagem completa (todo o ecran) disponível para avaliação muito tempo depois de ter iniciado o processo, o que torna o *Ray Tracing* impeditivo para as aplicações de CAD mais interactivas.

Aceita-se no entanto que, neste caso, uma imagem comece por ser criada com menor realismo e, à custa de mais tempo de processamento, se vá progressivamente tornando mais realista. A qualquer momento, o utilizador pode interromper o processo para efectuar alterações de geometria ou de aspecto dos objectos em cena.

Se o utilizador pretende obter imagens simples, pouco exigentes em termos de informação de aspecto, a primeira fase de tal processo poderia com algum esforço ser implementada com um algoritmo de visibilidade, evoluindo depois para um *Ray Tracing*. No entanto, se o utilizador pretende obter alguma informação de aspecto, esta solução torna-se complexa pois aí o *Ray Tracing* com toda a sua capacidade de simular reflexões, refracções e sombras não é substituível por um algoritmo de visibilidade por mais completo que ele seja.

O que propomos é uma modificação ao algoritmo de *Ray Tracing*, permitindo fazer o tratamento ponto a ponto de um ecran, por níveis de realismo crescentes. Deste modo o utilizador pode, desde o início e ao longo do tempo, avaliar a imagem e interromper o processo se porventura chegar à conclusão que não é a mais conveniente.

O principal inconveniente desta solução é a quantidade de memória que necessita. É necessário manter em memória o estado de cada raio proveniente de cada ponto do ecran, de forma a dar continuidade ao seu processamento no nível de realismo seguinte. O caso agrava-se quando existem objectos transparentes em cena dado que, nestes casos, cada raio luminoso dá origem a dois novos raios, um por reflexão e outro por transmissão.

4.2. Paralelização do *Ray Tracing*

Os tempos de processamento do *Ray Tracing* devem-se essencialmente aos testes de intersecção de raios luminosos com os objectos em cena. Cada novo raio tem que ser testado com todos os objectos para determinar qual destes é o primeiro a cortar o seu percurso.

Várias soluções têm sido apresentadas para resolver o problema, baseadas na sub-divisão espacial da cena e/ou em métodos (alguns já mencionados) que permitem acelerar os testes de intersecção referidos. Apesar disso, os tempos de processamento continuam a ser demasiado elevados para certas aplicações de CAD.

Nos últimos tempos, as tecnologias de arquitecturas paralelas têm sofrido um forte incremento, destacando-se uma em particular: os Transputers.

De momento, encontra-se instalado no INESC.Norte um sistema de desenvolvimento para Transputers sobre o qual se faz algum trabalho, nomeadamente na área do *Ray Tracing*.

Numa primeira fase, o programa de *Ray Tracing* descrito anteriormente, escrito em linguagem C, foi transportado para aquele sistema de desenvolvimento, ficando a correr num só Transputer. Comparando os tempos de execução com os que eram obtidos com o mesmo programa noutras máquinas disponíveis, verifica-se que

somente a SUN 4 (SPARC) consegue melhores resultados e, mesmo assim, com diferenças que não são muito significativas, se atendermos ao facto de o sistema de desenvolvimento para Transputers utilizar o disco do PC/AT em que está instalado, nitidamente mais lento do que o disco daquela Workstation.

Encontra-se em fase de implementação uma versão modificada do programa, ainda em C, que irá correr em vários Transputers. Trata-se de uma versão que divide o trabalho por vários processadores, mantendo em cada um deles uma descrição completa da cena (mais memória com informação redundante).

Outras versões estão a ser pensadas, escritas em OCCAM 2, que aproveitarão os processadores de uma forma mais inteligente e tendo já em vista o *Ray Tracing* de Realismo Crescente (secção 4.1.).

5. CONCLUSÃO

Neste trabalho apresentou-se uma panorâmica sobre a situação actual do *Ray Tracing*, pondo em realce os principais problemas e limitações de que enferma.

A nossa experiência sobre a síntese de imagens realistas baseadas em *Ray Tracing* é sucintamente descrita e apontam-se as ideias por nós utilizadas no sentido de se ultrapassarem as limitações relativas ao modelo de iluminação, tempo de processamento e *aliasing*.

Concluímos com a apresentação de algumas vias de investigação que exploramos, nomeadamente o *Ray Tracing* com realismo crescente e a sua paralelização.

BIBLIOGRAFIA

APELL68
Some Techniques for Shading Machine Renderings of Solids
Proc. AFIPS JSCC, volume 32, páginas 37-45, 1968

COOK84
Distributed Ray Tracing
Robert Cook, Thomas Porter, Loren Carpenter
Computer Graphics, volume 18, nº 3, páginas 137-145, Julho 1984

COOK86
Stochastic Sampling and Distributed Ray Tracing
Robert Cook
ACM Transactions on Graphics, vol. 5, nº 1, pág. 51-72, Janeiro 1986

COSTA88
Geração de Superfícies pelo Método de Revolução Generalizada
António Costa
1º Encontro Luso-Alemão de Computação Gráfica, Outubro 1988

KAY86
Ray Tracing Complex Scenes
Timothy Kay, James Kajiya
Computer Graphics, volume 20, nº 4, páginas 269-278, Agosto 1986

MARK88
The MTV ray tracer
Mark VandeWettering, Eric Haines
Software de domínio público, Setembro 1988

WHIT80
An Improved Illumination Model for Shaded Display
Turner Whitted
CACM, volume 23, nº 6, páginas 343-349, Junho 1980