1.
Introdução

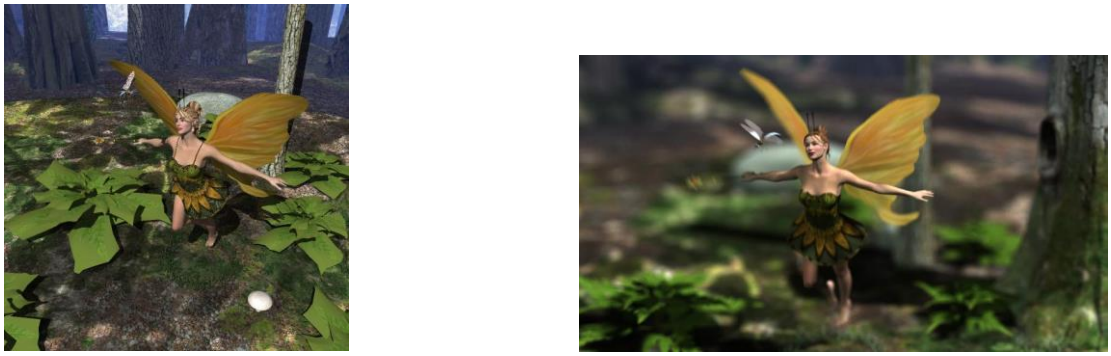
Com o crescente aumento de produções de cinema e TV, que buscam produzir novos meios de contar estórias, reduzir custos e eliminar riscos para as equipes, a área de renderização fotorealista de alto desempenho tem crescido em importância no meio acadêmico e industrial. No entanto, mesmo com as evoluções extremas dos *hardwares* e de alguns algoritmos, o processo de produção das imagens finais em altíssima resolução ainda é um ponto extremamente crítico na linha de produção, sendo reservado até mesmo um tempo considerável no planejamento dos projetos. Os desafios para solucionar esse problema são inúmeros – e clusters, computação em nuvem e técnicas de otimização e de gerenciamento de renderização têm sido explorados.

O *Ray tracing* em tempo real é um tópico ativo de pesquisa na área de games, simuladores e *CAD*. Pesquisas nesta área ainda apresentam resultados muito modestos quando as cenas são dinâmicas e deformáveis. De fato, apesar do uso de paralelismo em *CPU* e *GPU,* os trabalhos mais recentes apresentam desempenho da ordem de apenas 25 *fps* em situações estáticas e 5 fps em situações dinâmicas, levando em conta que as cenas em questão suportam um número de triângulos limitado, entre 180K a 280K (Wald et al., 2007).

Além disso, as cenas estão longe da qualidade imposta pela indústria do Cinema/TV digital, que envolve efeitos visuais complexos (desfocagem, *motion blur*, ...), alta resolução (tipicamente 1920 x 1080 em Full HD e mínimo de 2048 x 1080 no cinema, e atualmente o 4K, 3840x2160, e o 8K, 7680x4320, sendo esta última já bastante usada na produção e, em breve, também na exibição) e estereografia, a qual exige, no mínimo, o dobro de quadros. Ainda, os trabalhos não consideram a

PUC-Rio - Certificação Digital Nº 1021807/CA

geração de imagens em formato *HDR*, imprescindíveis no estado da arte de produção do cinema, que trabalham com canais de 12 a 32 bits, ao invés dos convencionais 8 bits. Até mesmo os novos *encoders* de vídeos já se preparam para suportar imagens com resolução dinâmica de cor, sendo como maior exemplo o h.265 (Sullivan et al., 2012). Exemplos de tempo e qualidade para *ray tracing* estão apresentados na Figura 1.



(a) (b)

Figura 1: A referência “Utah Fairy”(Wald,I. 2010) nas versões: (a) ray tracing em tempo real , 180K triângulos, 1024 x 1024, 1 única fonte de luz e aprox. 4 *fps* em dual core 64 bits (figura extraída de Wald *et al.*, 2007); (b) ray tracing de imagem HDTV, 1280 x 720, wide-screen, com desfocagem e motion blur, aprox. 6M micropolígonos, aprox. 60 seg em GPU (para um único frame) (figura extraída de Hou *et al.*, 2010).

As metas de desempenho para *ray tracing* em produções de Cinema/TV digital são bem mais modestas do que as almejadas em games. A unidade de comparação nem chega a ser fps (*frames per second*). Como exemplo, numa única máquina de oito núcleos (octa core), numa cena complexa de qualidade de cinema, considera-se 50min um bom desempenho (Figura 2). Dessa forma, a meta nesses estudos não é a de *Ray Tracing* em Tempo Real, mas a de *Ray Tracing* Interativo, de maneira que seja possível o artista mudar a câmera, luzes e materiais enquanto produz imagens de qualidade suficiente para analisar o resultado a velocidades interativas.

Recentemente, NVidia produziu dez servidores com um sistema capaz de renderizar cenas próximas das usualmente utilizadas em cenas

PUC-Rio - Certificação Digital Nº 1021807/CA

reais de forma interativa. Tal façanha foi atingida com o uso do sistema NVidia Grid (Herrera, A., 2014), que consiste em um servidor com dois processadores, num total de 64 núcleos, 256 GB de memória *RAM* e oito placas de vídeo da séria K40, que, no momento da escrita desse documento, são as que possuem a maior capacidade de processamento, com cerca de 2880 núcleos por placa. Esta máquina é capaz de gerar uma imagem em resolução de 1280x720 em qualidade final em cerca de 5 minutos; porém, sem todos os passes (informações adicionais da imagem) e sem coerência de frames, podendo criar *flickering* (piscadas) na imagem. Com dez servidores do tipo mencionado acima, conectados em rede Infiniband de 56 Gbps de banda, foi possível utilizar o sistema com qualidade final a 25 fps (teste apresentado no SIGGRAPH 2014 no setor dos expositores). Ou seja, com 640 núcleos de processamento de *CPU* e 28800 núcleos de processamento *GPU*, ou 43 Tflops de capacidade de processamento, mesmo assim, a cena ainda não é representativa de todas as dificuldades de cenas de cinema/TV digital.

Figura 2: Cena complexa com qualidade de cinema, com 678 milhões de triângulos, 106 minutos de rendering, em um Apple G5 com 2 processadores PowerPC de 2 GHz (figura extraída de Christensen et al. (2006)).

O fato apresentado acima indica o nível de complexidade exigido pelo cinema e pela TV; lembrando, porém, que um sistema como o

PUC-Rio - Certificação Digital Nº 1021807/CA

descrito, ainda não é capaz de realizar efeitos com fogo, fumaça, nuvens, que atualmente podem levar de horas a dias para serem renderizados. Atualmente, a solução encontrada pela indústria é a utilização de clusters gigantescos de servidores para computar os frames, sendo chamados de fazendas de render (*Render Farms*). No entanto, coordenar a renderização de milhares de frames é uma tarefa complexa, mesmo tratada de maneira muito simples como fazem a grande maioria dos gerenciadores de *render farms*. Tal importância é tão expressiva que recentemente a Pixar foi contemplada com o *Oscar Scientific & Technical Awards* (Pixar, 2011) por seu sistema de gerenciamento de filas de renderização. Contudo, os gerenciadores ainda são incapazes de avaliar a melhor forma de distribuição de tarefas e como ajustar às necessidades de uma demanda emergencial.

Ainda, mesmo com o uso da computação em nuvem, o uso se restringe a uma expansão fixa do número de servidores. Essa estratégia é, na verdade, apenas um aumento fixo do número de servidores, que muitas vezes se torna inviável por conta da enorme quantidade de dados necessária para a renderização.

Portanto, as lacunas na otimização de renderização e no gerenciamento da renderização são exemplos de necessidades vitais da indústria. A presente tese propõe algumas soluções eficientes para estes problemas.

1.1. Objetivos

O trabalho proposto é composto de um sistema distribuído de renderização fotorealista que tem, como um de seus objetivos principais, a meta de otimizar o processo de renderização localmente e globalmente. Além disso, o trabalho foca num processo de renderização que permite a combinação de técnicas e modelos distintos.

PUC-Rio - Certificação Digital Nº 1021807/CA

Nesse processo, o sistema analisa a sequência temporal dos dados e otimiza as técnicas que devem ser empregadas e as une de forma harmoniosa, ou seja, não física, usando técnicas normalmente utilizadas por especialistas – porém de forma automática e com maior precisão e controle.

Para isso, o trabalho utiliza as otimizações de renderização com *GPUs* e *CPUs*, e organizações de clusters de *CPUs* e *GPUs*, bem como nuvens como Amazon e Azure para acomodação dinâmica de demanda.

Para combinar as diversas técnicas, cria-se o conceito de zonas de suavização, que consistem em um espaço de transição de técnicas e modelos de renderização.

Outro objetivo cerne do sistema é o gerenciamento de renderização (um elemento fundamental, mas pouco explorado na literatura), tanto do ponto de vista local, ou seja, a distribuição de tarefas para as *CPUs* e *GPUs* para um frame ou subframe, quanto para um *cluster*/nuvem. O gerenciamento apresentado é capaz de estimar o tempo de renderização da animação de forma a conseguir se comprometer em entregar o conjunto de frames em um determinado tempo (restrição temporal). Ou seja, usando as estatísticas da renderização, o sistema proposto é capaz de subdividir atividades e alocar mais recursos (no caso da nuvem) para finalizar a animação no tempo determinado. Outra possibilidade é a redução da qualidade para atender a restrição temporal e de recursos. Nesse último caso, o gerenciador é capaz de solicitar ao renderizador que escolha técnicas mais simplificadas para atender as demandas.

Além disso, o renderizador otimiza a forma de subdividir a renderização de um *frame* ou *subframe*, pois há áreas que quase não possuem elementos e outras que possuem muitos elementos e geram muitos raios secundários. Nesse caso, o objetivo é usar técnicas

PUC-Rio - Certificação Digital Nº 1021807/CA

estatísticas para melhor subdividir a imagem (sabendo que muitas vezes não há informação do frame anterior), colocando mais processadores em áreas de maior trabalho. Ainda, essa análise deve ser útil para a animação como um todo – ou seja, ao iniciar um frame, deve-se utilizar os dados do anterior (se houver) para já iniciar a subdivisão.

Por fim, o presente trabalho tem a preocupação de considerar o fato de que, durante a renderização de uma animação em um *cluster*, vários processos são repetidos diversas vezes pelos renderizadores. Dessa forma, um outro objetivo desta tese é a criação de um módulo colaborativo de renderização, onde cada nó (*node*) relata informações relevantes sobre o processo de renderização do *frame*, permitindo que outros nós otimizem a renderização, ou que em outra renderização da sequência isso seja aproveitado. Um exemplo desse conceito é que se uma posição de câmera e animação gera uma zona escura (subexposta) ou clara (superexposta), o renderizador pode reduzir a emissão de raios nessas regiões e, assim, reduzir o tempo de renderização total.

PUC-Rio - Certificação Digital Nº 1021807/CA

Contribuições

As contribuições inéditas da presente tese podem ser elencadas como se seguem:

- Metodologia de Renderização Multitécnica.

  O método proposto realiza a seleção de qual método empregar, de forma automática, usando como base a especificação visual dos materiais dos elementos de cena. A referência mais próxima a esta contribuição é (SABINO et al, 2012). Em relação aos trabalhos relacionados, essa proposta amplia a gama de técnicas empregas e a forma de selecionar, baseando-se em aspectos do material empregado e não na escolha do usuário.

  Várias técnicas podem ser facilmente consideradas; porém apenas as seguintes técnicas são implementadas nesta tese:

  - *Scanline* (Rasterização)
  - *Virtual Ray Lights*

- *Ray Tracing* e *Path Tracing*

- Algoritmo de Suavização de Fronteiras de Técnicas através de interpolação 2D e 3D.

  Sendo um algoritmo criado para a metodologia multitécnica, não foi encontrado algoritmo similar na literatura.

- Algoritmo de Seleção e Distribuição Estatística de seções de renderização de imagens.

  Este algoritmo usa *optical flow* ou canal de velocidade para fazer correspondência de pixels. A referência mais próxima a esta contribuição é (Abraham et al, 2004). Com relação a essa contribuição, a diferença fundamental é a adaptação estatística para renderização não tempo real, ou seja, considera-se que pode não haver informações sobre frames próximos. Nesse ponto, há outra inovação, não encontrada na literatura: a estimativa e a avaliação de dados de um *frame* usando dados de *frames* predecessores e sucessores com uso de *optical flow* para realizar a correspondência.

- Algoritmo de Otimização de *Shaders* Baseado em *Shade Trees*

  A principal mudança em relação à literatura está na manipulação de shader de tempo real e de técnicas como *ray tracing*, o que amplia muito o nível de funcionalidades. Foley e Hanrahan (2011) implementam geradores de shader para linguagens de tempo real (GLSL, HLSL e Cg); além disso, não se discute optimização, apenas geração. Allard & Raffin (2005) criam uma camada para gerenciar as primitivas gráficas, permitindo que sejam feitas otimizações, no entanto, são otimizações de função e funcionalidades novas das placas. Nessa tese, além das otimizações de *hardware*, são feitas otimizações de complexidade na cena, reduzindo a complexidade do *shader*, dependendo também da região e forma que o mesmo aparece na cena.

PUC-Rio - Certificação Digital Nº 1021807/CA

- Algoritmo de gerenciamento de render estatístico para renderizações com restrição temporal e *tiering* de *cluster* e nuvem.

  Quanto ao gerenciamento de render, existem algumas soluções comerciais e acadêmicas, tais como a biblioteca de passagem de menssagens OpenMPI e os gerenciadores Backburner, Qube! e outros. Entretanto, esses sistemas são apenas distribuidores de *frames*, não analisando o tempo que será gasto e nem provendo soluções para atingir metas de tempo e recursos; e muito menos preocupando-se em alocar recursos na Nuvem. Assim, essa tese se diferencia ao criar uma metodologia para gerir uma renderização que seleciona os melhores recursos e determina quantos recursos adicionais serão necessários para cumprir, por exemplo, uma meta de tempo.

- Gerenciamento integrado de renderizações com inteligência (otimização *online*) e com algoritmos de renderização colaborativa.

Na literatura consultada, não foi possível encontrar uma metodologia similar aplicável a renderização que usa a colaboração para otimizar uma renderização. Assim, a presente tese inova ao criar um modelo de colaboração de dados de renderização entre nós, visando otimizar o processo geral.

1.2. Estrutura da Tese

O presente trabalho está organizado da seguinte maneira. O capítulo 2 aborda os trabalhos relacionados. O capítulo 3 concentra-se nos métodos e modelos propostos no renderizador local, ou seja, na etapa de execução da renderização.

O capítulo 4 foca nos métodos e modelos desenvolvidos no processo de gerenciamento da renderização e como eles interferem na renderização local. Além disso, discute-se o modelo empregado para o uso em *cloud computing* públicas.

PUC-Rio - Certificação Digital Nº 1021807/CA

Já o capítulo 5 apresenta a metodologia de compartilhamento de informações entre os nós do *cluster*. No capítulo 6 são apresentados os resultados dos testes de desempenho das metodologias. Finalmente, no capítulo 7, conclui-se a pesquisa e apontam-se os trabalhos futuros. Esta tese também apresenta um glossário, após a seção de bibliografia, com os termos mais importantes referidos no texto.

PUC-Rio - Certificação Digital Nº 1021807/CA