5.
Otimização Integrada

Nos sistemas de gerenciamento da renderização convencionais, todo o processo de controle e passagem de instruções é feito pelo sistema gestor. Nesse ponto, os *nodes* são totalmente isolados dos demais. No entanto, sendo o processo de renderização uma tarefa de longa duração, durante o processamento de um *frame* ou *subframe,* muitas informações relevantes ficam ociosas esperando a finalização do processo para se tornarem públicas.

Em sistemas convencionais de gerenciamento e de renderização, não há troca de informações entre nós, cada nó foca apenas no processamento dos dados. Assim, as informações dos metadados ao respeito de um *frame* são apagadas ao se finalizar a renderização do mesmo.

Dessa forma, uma metodologia diferencial do gerenciador deste trabalho é a possibilidade de que os nós consigam compartilhar informações entre si (renderização colaborativa), permitindo que eles possam se antecipar ou corrigir problemas o mais rápido possível.

Uma hipótese ingênua seria a comunicação N para N, o que garantiria o acesso imediato e assíncrono aos dados de cada servidor, ou o envio de mensagens de *broadcast*. Contudo, comunicar-se dessa forma sobrecarregaria o sistema de *threads* e tornaria o renderizador muito ineficiente, pois em grandes *clusters*, o custo de manter esses dados organizados poderia causar uma grande perda no processo de renderizar, que é a função primária.

O uso de sistema de mensagens de *broadcast*, apesar da simples implementação, necessitaria que o renderizador gerisse as informações de N nós, assim, também é custoso.

De maneira análoga, numa equipe, a troca de informações (conversa) é importante para que todos possam aprimorar e corrigir problemas; no entanto, o excesso delas ou o desvio do foco das informações pode atrapalhar mais o processo do que ajudar. Nesse sentido, deve haver uma forma dos nós se comunicarem e de receberem informações de forma precisa e seleta e, assim como ocorre nas relações de trabalho, esse papel deve ser feito pelo coordenador/gerente do processo.

5.1.1. Compartilhamento de Informações

No sistema proposto, um nó é capaz de gerar as informações a respeito da renderização, através da *Performance Image* (figura 19, capítulo 3), que auxiliam no processo da distribuição da renderização entre as *threads* de processamento, conforme discutido no capítulo 3. Assim, parte dessa informação poderia ser disponibilizada entre os nós a cada bloco processado, ou seja, toda vez que um bloco da *quadtree* finaliza a renderização, a *performance image* desse pode ser compartilhada.

Outra informação relevante são as zonas de alta intensidade, ou seja, zonas em que a quantidade de luz emitida causa um efeito de estouro do espectro de representação de cor. Assim, num processo de *Path tracing*, o tempo de renderização aumenta muito (devido ao número de rebatimentos necessário para que o decaimento de energia seja suficiente para finalizar o procedimento recursivo de amostragem). Portanto, sabido que uma zona da imagem possui tal característica, e que os dados de saída utilizarão uma representação RGB e não CIE LAB, o renderizador pode reduzir o número de percorrimentos recursivos, o qual

pode ser informado pelo nó que executou o processo. Essas são as informações que são compartilhadas entre os nós.

No entanto, a quantidade de informações enviadas para o gerenciador pode ser extremamente grande, criando um grande gargalo no processamento das mesmas. Dessa forma, o gerenciador conta com um sistema de fila de mensagens que são consumidas utilizando uma metodologia *thread-safe* de acesso.

Quando a mensagem é recebida pelo servidor, a *thread* que a recebeu adiciona o dado na fila e inicia o processo de leitura de um novo dado, que fica na fila TCP do sistema operacional para a porta específica. Uma outra *thread* realiza a leitura do dado presente na cabeça da fila e o processa e o adiciona num grafo que chamamos nesta tese de “grafo de *frames*” (figura 49).

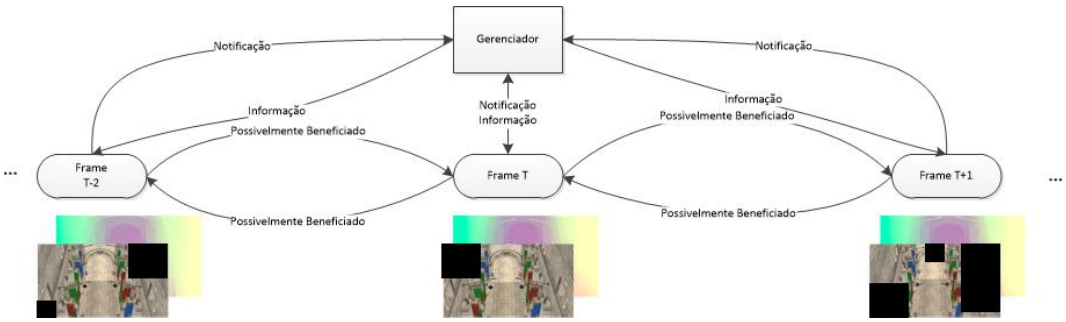

Figura 49: Grafo de frames proposto para a renderização colaborativa (imagem produzida)

Um grafo de *frames* é uma estrutura no formato de grafo direcionado que organiza as informações de renderização, tendo como nó cada *frame* em execução, isso pelo fato de que *frames* finalizados já estão com seus dados armazenados em disco e são usados conforme explicado em 3.1.4. Dessa forma, cada nó possui, excluindo nós de *frames* de fronteira, quatorze conexões, que são, respectivamente, os três nós de *frames* predecessores e os três nós de *frames* sucessores do mesmo (duas por *frame*); assim, dado um *frame*, pode-se obter

informações muito rápidas dos *frames* próximos, se os mesmos estiverem em processamento.

Cada nó possui ainda uma conexão com o nó cena, que basicamente é o acesso direto a cada *frame* e servidor. A implementação é extremamente simples, consistindo de um dicionário indexado pelo número do *frame e* armazenando o servidor que processa o dado.

Dessa forma, ao ser questionado sobre a atualização de estado de um determinado *frame*, o gerenciador rapidamente consegue extrair o dado e até mesmo enviar os dados dos *frames* próximos (no cálculo da *performance image* estimada, pode-se adicionar dados de vários *frames* próximos).

No entanto, existem momentos em que o renderizador tem interesse em verificar se existe algo novo sobre o que está para ser executado. Assim, existem dois momentos especificados para tal execução, o Pré-Calculo e o processo em execução.

5.1.2. Método em Pré-Calculo

Servidores podem solicitar informações de cálculo dos nós em execução quando o renderizador está ainda se preparando para a renderização, ou seja, quando ele apenas possui os dados das *imagens de performance* dos frames já renderizados. Nesse momento, o renderizador solicita ao gerenciador se ele possui atualizações sobre os *frames* próximos a ele. No caso, sendo o *frame* em questão *f*, o renderizador solicitará dados dos *frames* no conjunto [f-3, f+3]; assim, uma vez que o *frame* esteja renderizado, o dado desse é enviado nulo pelo gerenciador.

O renderizador então possui uma *performance image* parcial do *frame* em execução. Assim, na equação 8, os pixels que possuírem dados computados por essa *performance image* parcial utilizam o dado da

mesma, já os demais utilizam apenas estimativas ou *performance images* de *frames* já concluídos.

5.1.3. Método em Execução

Uma outra forma de aproveitar dados em processamento pode ser realizada quando o gerenciador antecipa dados úteis para os nós de renderização. Para isso, o renderizador, após subdividir a *Quadtree* de renderização, envia (notifica) ao gerenciador a lista de blocos a serem executados; sendo que toda vez que um bloco é finalizado, o nó também notifica o servidor sobre a completude do mesmo e envia a PI parcial.

Assim, uma vez recebida uma PI de um bloco, o gerenciador verifica dentro de um raio de n *frames* (no caso dos testes realizados, n = 4 *frames*), quais os nós que se beneficiariam com essa informação, fazendo uma varredura nos blocos remanescentes de cada um deles e calculando a interseção de retângulos. Se houver interseção, o nó responsável pelo *frame* de colisão recebe a PI pela interface administrativa do renderizador, através do *ClusterNode*. O percorrimento desse raio de ação depende da completude de *frames* intermediários, ou seja, dado um *frame* na posição *t*, supondo que o *frame* anterior completo mais próximo de *t* seja *t-2* e o *frame* posterior completo mais próximo seja *t+3*, o intervalo de dados amostrado será [t-1, t+2], conforme figura 50.

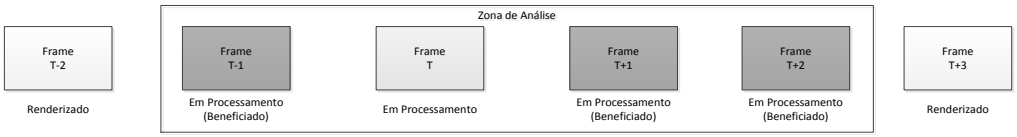

Figura 50: Intervalo de análise em execução (imagem produzida)

Caso, de fato, o renderizador ainda não tenha iniciado a execução dos blocos com interseção, é refeito o cálculo de subdivisão da área remanescente e a nova configuração de blocos é enviada ao gerenciador. Dessa forma, o nó de trabalho pode se antecipar a variações mais

bruscas nas PI Isso pode acontecer em cenas em que a velocidade dos elementos é muito grande, como cenas que envolvem aviões e movimento rápidos de câmeras.

5.1.4. Dump de dados

Uma outra utilidade da metodologia proposta é o uso do recurso de *Dump* de dados. Assim, se algum nó sofre uma falha e é finalizado, o gerenciador, que recebe as notificações de performance continuamente do mesmo, percebe a perda da execução e realiza a gravação da *performance image* parcial no disco. Esse arquivo só difere da *performance image* original por possuir um canal extra de máscara e zonas de alta intensidade (estouro).

Quando a execução do *frame* é retomada por outro nó, o mesmo possui uma forma muito mais precisa de subdivisão para o *frame* em produção, visto que o dump de dados o forneceu uma *performance image* parcial do frame que estava em andamento. Nesse caso, o renderizador utiliza apenas o dado da *performance image parcial* do *frame* (nos *pixels* que efetivamente foram produzidos, os demais pixels seguem a equação 8) para a subdivisão.

Eventualmente o Dump pode gravar os resultados da renderização em sí (imagem final, normais e outros) dos blocos que foram concluídos. Porém, essa funcionalidade não foi implementada por consumir muita memória do gerenciador.

Por fim, para que os dados trafegados não comprometam a banda de comunicação, esses são enviados utilizando a compressão GZip.

Esta inovação no processo de uso de nós de renderização finaliza a parte de exposição dos sistemas desenvolvidos pela presente tese. O próximo capítulo apresenta a etapa de testes e resultados.