CENTRO DE INFORMÁTICA - UFPB

Tópicos especiais em computação

Relatório parte 5

JOÃO PESSOA, 03-MAIO-2019

O objetivo dessa parte 5 da implementação do path tracer é melhorar o desempenho do render, seja por meio de paralelismo, fazendo uso de bibliotecas e API's que implementam multithreding, ou por meio da implementação de uma estrutura de aceleração.

Esse relatório será dividido em tópicos de acordo com a técnica de aceleração e suas peculiaridades, tendo no fim algumas imagens e dados que demonstrem as melhorias encontradas.

1) Multithreding:

Visando explorar ao máximo o desempenho da máquina utilizada para renderizar nossas imagens, foi feito uso da biblioteca OpenMP para que dessa forma pudéssemos usufruir das várias threads do computador e assim tornar o path tracer um algoritmo mais eficiente para casos mais complexos a serem renderizados.

Com relação às melhorias no código, utilizamos o comando do OpenMP "#pragma omp parallel for", fazendo com que fosse feito o uso de várias threads para processar ao mesmo tempo. Além disso, ao invés de cada thread processar uma determinada região da imagem, foi usado o comando "schedule(dynamic, 1)", para as threads ficarem alternando o processamento de cada linha. Isso foi feito pensando no caso da cena estar concentrada em uma região da imagem, e threads que não processam essa região podem acabar mais rápido, causando um uso não otimizado da CPU. Por fim, tendo em mente escolher o número de threads para serem usadas na zona de paralelismo, usufruímos do comando ''omp_set_num_threads(8)", sendo o número entre parênteses o número que você quiser de threads.

Para os testes fizemos uso de uma máquina com processador intel core i7 de oitava geração, quad core, 8M de cache e possuindo 8 threads.

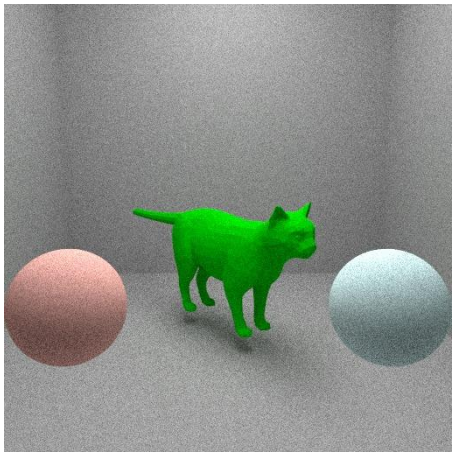
2) Estrutura de aceleração:

Já a construção da estrutura de aceleração não teve o mesmo intuito de explorar ao máximo o desempenho da máquina em si, mas sim de tornar mais eficiente o arremesso de raios em direção à cenas de grande porte, visto que nessas situações o número de raios que são lançados mas que não atingem nenhuma primitiva é muito grande, fazendo com que tempo seja perdido nesses momentos de falta de intersecção.

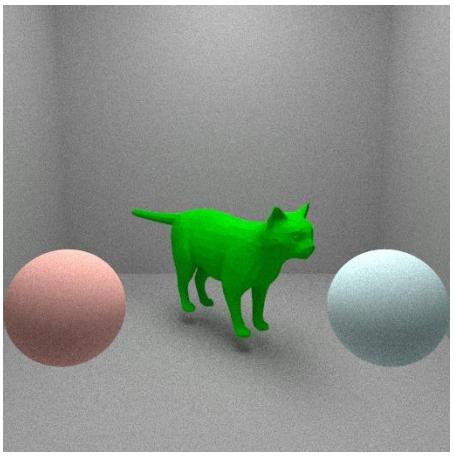
Para a construção da nossa estrutura, fizemos uso de uma árvore do tipo bvh, além de adotar o SAH(surface area heuristic) como critério de subdivisão de cena, tendo como profundidade 19 nós e tempo de construção de 0.024 segundos.

3) Resultados:

A cena montada e renderizada para o relatório cinco é composta por 2099 primitivas, sendo uma caixa difusa de cor cinza, duas esferas difusas (uma de cor azul e outra de cor laranja) e por fim um gato difuso da cor verde.



Cena com 400 raios



Cena com 1000 raios

Para as medições abaixo foi utilizada a cena mostrada anteriormente com 400 raios lançados por pixel. Seguem as medições para a renderização apenas com multithreading, apenas com estrutura de aceleração e por fim com ambas as técnicas.

Tempo de renderização usando apenas o multithreads: 4840.764 seg

Tempo de renderização usando apenas o BVH: 581.063 seg

Tempo de renderização usando ambas as técnicas: 165.63 seg