INF 2063 – Visualização de Modelos Massivos

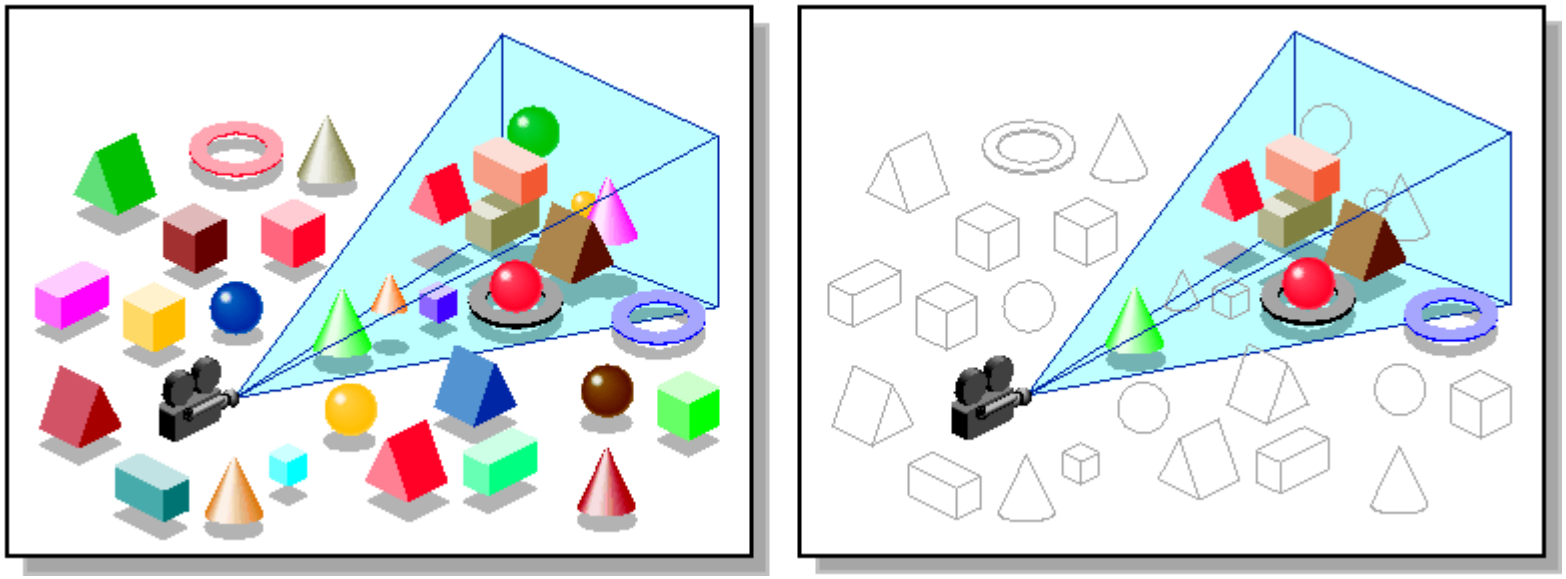
Trabalho de Pesquisa

Descarte por Oclusão
Algoritmo CHC++

Vitor Barata R. B. Barroso
vbarata@tecgraf.puc-rio.br

Descarte por Visibilidade

- Principais tipos de descarte
  - Contra o volume de visão: primitivas fora do campo visual
  - Por oclusão: primitivas escondidas atrás de outras primitivas



Descarte por visibilidade contra volume de visão e por oclusão
(fonte: OpenGL optimizer programmer's guide: an open API for large-model visualization)

Descarte por Oclusão

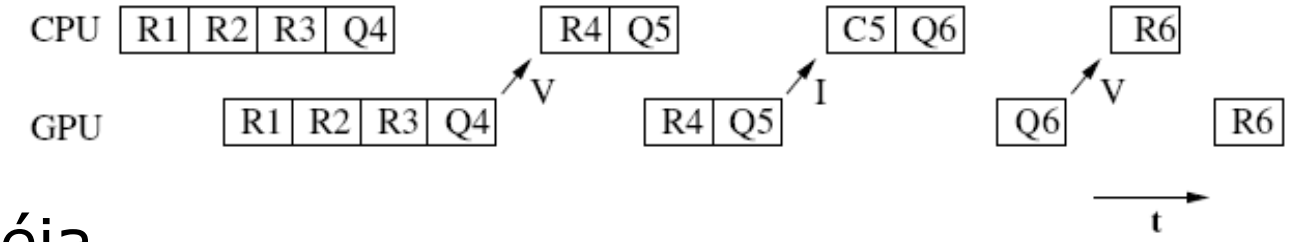
- Processamento
  - *Online* (por ponto): O PVS é determinado dinamicamente, em tempo de execução, para o ponto onde se encontra o observador no momento.
  - *Offline* (por região): Em pré-processamento, determina-se o PVS para vários pontos-de-vista espalhados pela cena. Em tempo de execução, escolhe-se e utiliza-se o PVS do ponto mais próximo.
- Abordagem
  - Exata: Determinam-se exatamente as primitivas visíveis ao observador.
  - Conservativa: Primitivas oclusas podem ser incluídas no PVS para economizar cálculos.
  - Agressiva: Primitivas visíveis podem ser excluídas do PVS.
- Espaço
  - Do objeto: o PVS é determinado a partir de testes envolvendo a geometria dos objetos da cena em seu espaço original. (Fast-V)
  - Da imagem: consideram-se os objetos após a rasterização, ou seja, uma amostragem de suas distâncias até o observador, como a fornecida pelo próprio z-buffer. (CHC++)

Teste de Oclusão em Hardware

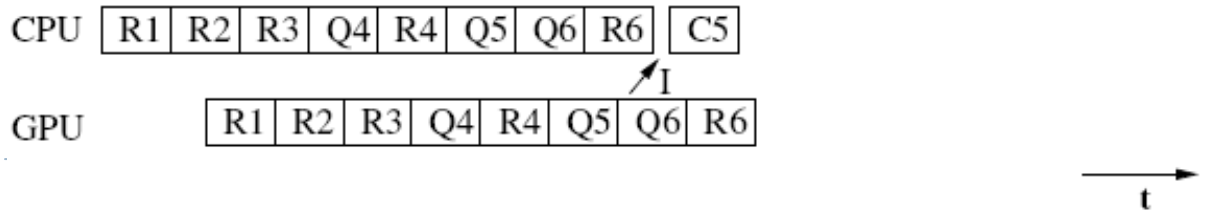
- Permite perguntar à placa gráfica quantos pixels seriam rasterizados numa operação de desenho qualquer
  - Considera teste de z com z-buffer atual
- Operação assíncrona
  - Chamada retorna imediatamente
  - Resultado demora algum tempo para ficar disponível
  - Pode-se consultar rapidamente se o resultado já está pronto
- Problema: ociosidade da CPU e da GPU
  - Se o resultado for requisitado cedo demais, CPU espera até que esteja disponível
  - Falta de novos dados para a GPU deixa-a subutilizada

Descarte Coerente Hierárquico (CHC)

- Ociosidade do CPU e da GPU



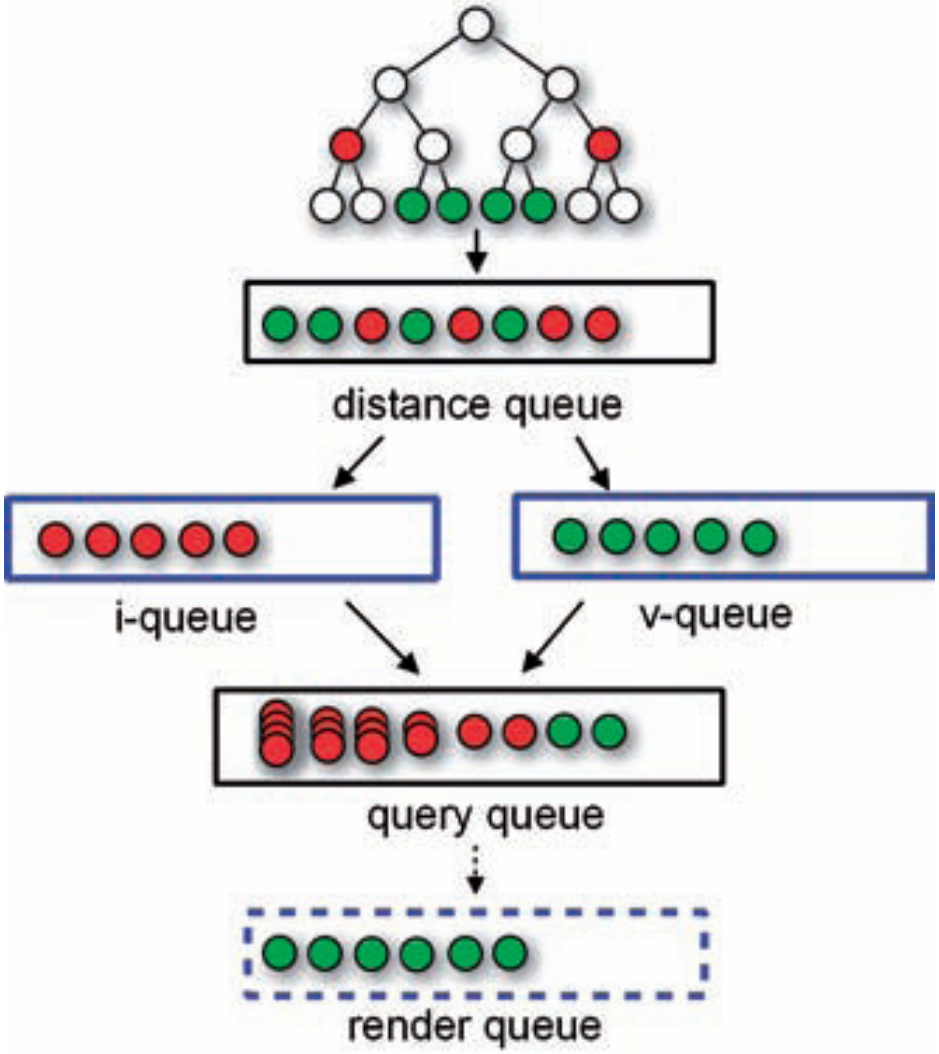
- Idéia
  - Explorar a coerência temporal da classificação de visibilidade
    - Visível tende a continuar visível e vice-versa
  - Reorganizar as chamadas
    - Intercalar testes de oclusão com outras operações
    - Enquanto espera, CPU continua processando e enviando dados à GPU



Algoritmo CHC++

- Testes de visibilidade em lote
  - Minimizar mudanças de estado da renderização
  - Grupos (lotes) de testes são requisitados conjuntamente
- Lotes de nós previamente invisíveis
  - Em vez de requisitar imediatamente, colocar na "fila-i"
  - Requisitar quando o número de nós na fila chegar a um valor *b*
- Lotes de folhas previamente visíveis
  - Em vez de requisitar imediatamente, colocar o teste na fila-v
  - Requisitar quando for preciso esperar por outras operações
- Lotes de nós a renderizar
  - Em vez de desenhar imediatamente, colocar numa fila de render
  - Renderizar toda a fila logo antes de qualquer teste de oclusão

Algoritmo CHC++



Algoritmo CHC++

- Multitestes
  - Agrupar nós previamente invisíveis que têm a mesma probabilidade de permanecerem assim no quadro atual
  - Fazer apenas um teste desenhando todos os nós
    - Se invisível, economiza vários testes
    - Se visível, testa um por um
- CHC: objetos visíveis tendem a continuar visíveis
  - Refazer seus testes de oclusão apenas a cada n_{av} quadros
  - Da primeira vez, escolher n_{av} aleatório entre 0 e um limite
- Volumes envolventes mais justos
  - Uso de AABBs simples
  - Para nós internos, volume é a união dos volumes dos filhos!

Características do Projeto

- Volumes envolventes
  - AABBs
  - Versão final: comparar com OBBs e esferas
- Hierarquia
  - Subdivisões por planos passando pela média dos pontos médios
  - Versão final: comparar com outras técnicas de construção e com subdivisões espaciais
- Percorrimento:
  - Guiado por fila de prioridades de acordo com a distância até o observador
- Integração com grafo de cena:
  - Uso de classes do grupo de visualização
  - Versão final: integração total com o grafo de cena SG

Implementação do Projeto

- Problema não explicado nos artigos
  - Volumes e objetos que atravessam o plano near
- Ciclo do algoritmo:
  - Resultados já disponíveis
    - Pode acrescentar à fila de prioridades e ao lote-R
  - Nós já visitados e com vez na fila de prioridades
    - Pode acrescentar aos lotes I, V ou R
  - Processamento do lote-I (inteiro)
    - Se necessário ou quando atingir tamanho determinado
    - Gera queries
  - Processamento do lote-V (nó a nó)
    - Gastar o tempo necessário para evitar esperar
    - Gera queries
  - Espera por novos resultados

Resultados Preliminares

Cena	VFC	CHC	CHC++
100 objetos médios	71 Fps	71 Fps	113 Fps
1000 objetos médios	7 Fps	23 Fps	28 Fps
1000 objetos pequenos	7 Fps	8 Fps	13 Fps

Referências

[1] P. Santos, **Técnicas de Otimização para Visualização de Modelos Massivos** – Relatório Final de Projeto, PUC-Rio, 2006

[2] J Bittner,M Wimmer, H. Piringer, W. Purgathofer, **Coherent Hierarchical Culling: Hardware Occlusion Made Useful** – Proceedings of Eurographics 2004

[3] O. Mattausch, J. Bittner, M. Wimmer, **CHC++: Coherent Hierarchical Culling Revisited** – Proceedings of Eurographics 2008