LEI / LCC
Departamento de Informática
Universidade do Minho

Visualização II

Visibilidade e Oclusão; Impostores

Resumo

- **BSP**
- Portais
- Oclusões
- Impostores
- Hierarquical Z-Buffer

Partição Espacial - BSP

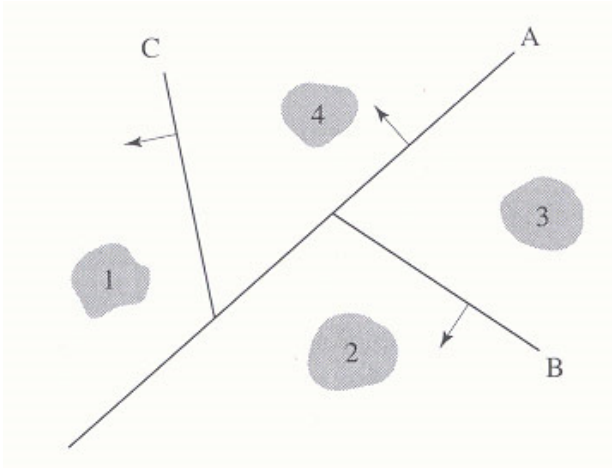
- BSP - Binary Space Partition
  - Utilizam planos para dividir o mundo.
  - A árvore resultante é uma árvore binária
  - Os planos são arbitrários (orientação e posição) e portanto pode ser vista como uma generalização de octrees e quadtrees

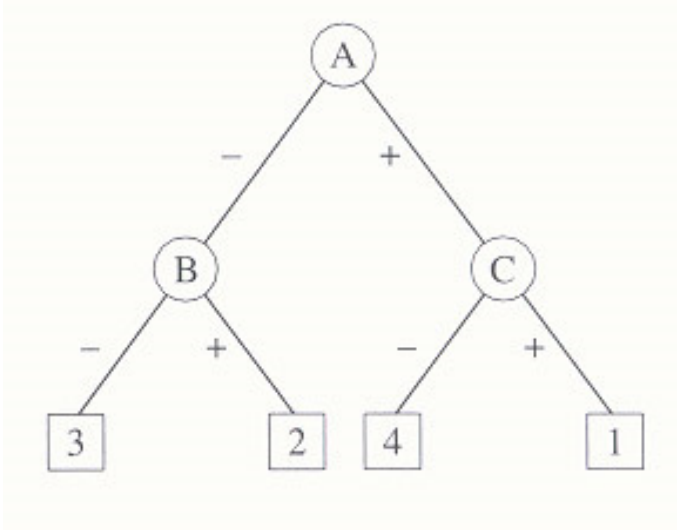
Partição Espacial - BSP

- Estratégias:
  - Utilizar polígonos da cena como "splitters"
  - Utilizar planos que separem objectos
- Questões:
  - Como escolher o plano?

Partição Espacial - BSP

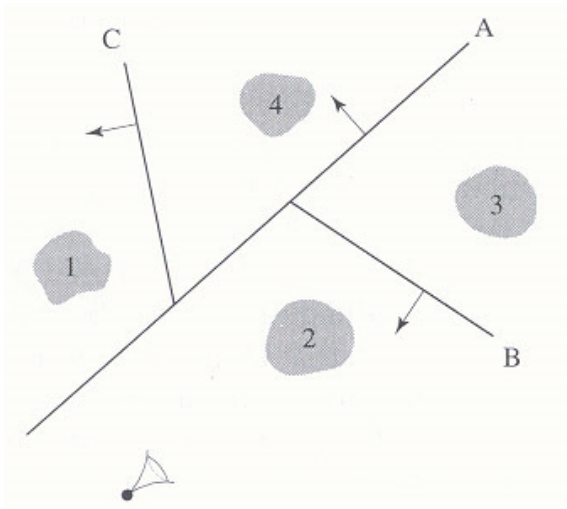
- Um exemplo de partição espacial com BSP

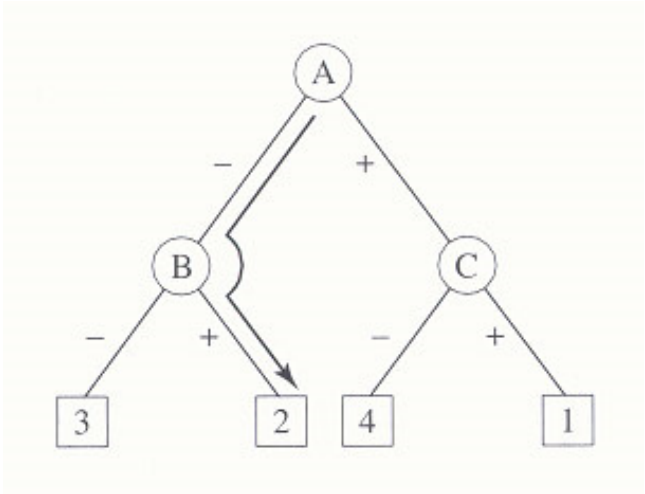




Partição Espacial - BSP

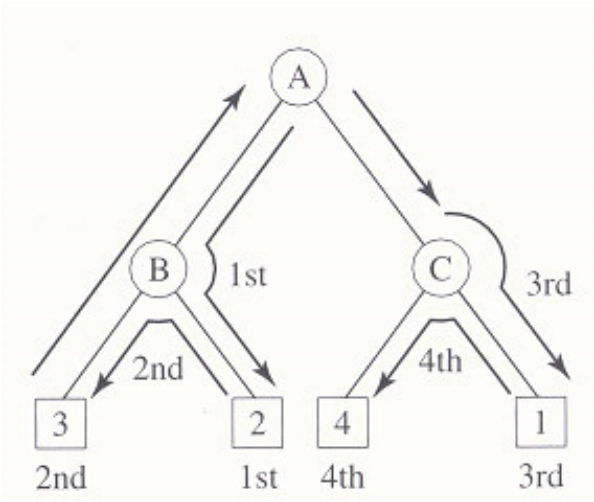
- Garantem uma ordenação dos objectos/polígonos





O objecto 2 é o mais "próximo"

Travessia ordenada da arvore

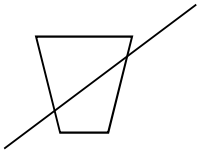


Partição Espacial - BSP

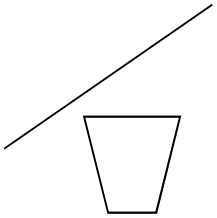
- Óptimas para ordenar
  - transparências
  - minimizar overwrite
- Dividem o espaço em zonas convexas

Partição Espacial – BSP

- View Frustum Culling
  - Se o VF "atravessa" o plano então ambos os lados do plano são potencialmente visíveis...



  - Caso contrário só é visível o lado onde se encontra o VF



Partição Espacial - BSP

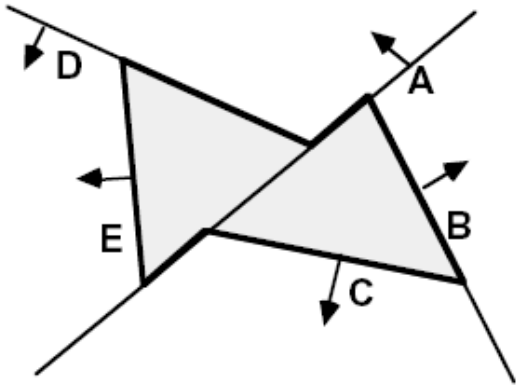
- View Frustum Culling
  - Uma outra estratégia, assumindo um mundo finito, é calcular uma hierarquia de bounding volumes para as célula definidas pelos planos.
  - O teste de culling é neste caso realizado com os bounding volumes e não com os planos.

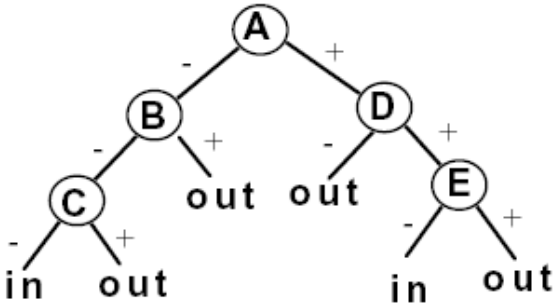
Partição Espacial - BSP

- Até agora nada foi assumido sobre os polígonos.
- Consideremos agora cenas de interiores de edifícios
  - polígonos = paredes, portas, janelas, ...
- As "paredes" servem portanto para definir os planos que recursivamente dividem o mundo

Partição Espacial - BSP

- Folhas representam regiões convexas





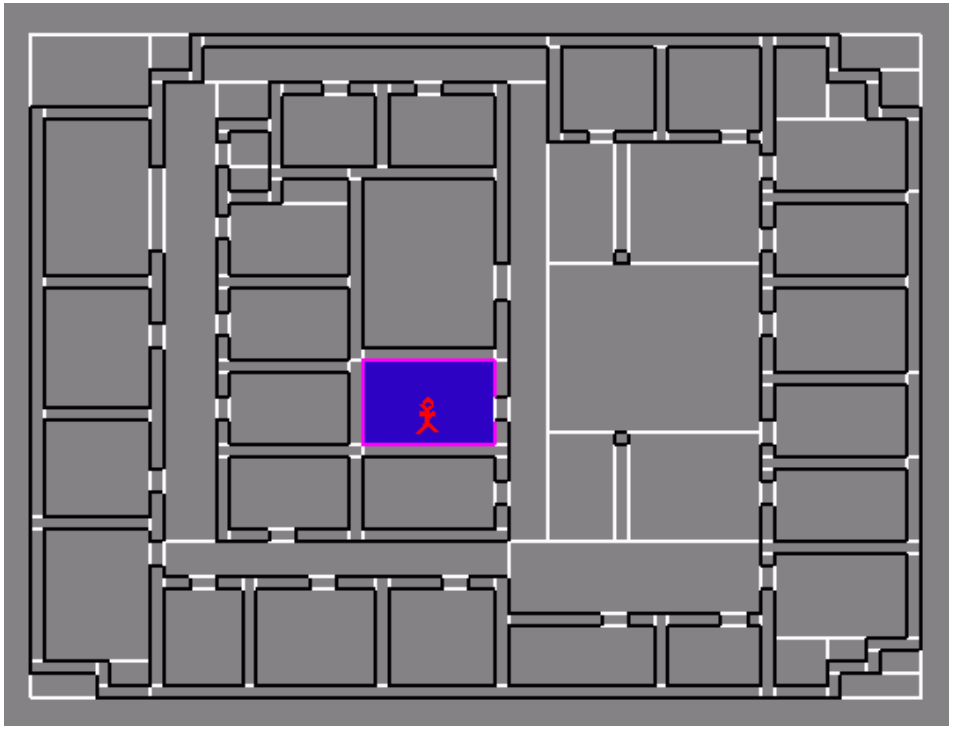
Resumo

- BSP
- **Portais**
- Oclusões
- Impostores
- Hierarquical Z-Buffer

Partição Espacial - Portais

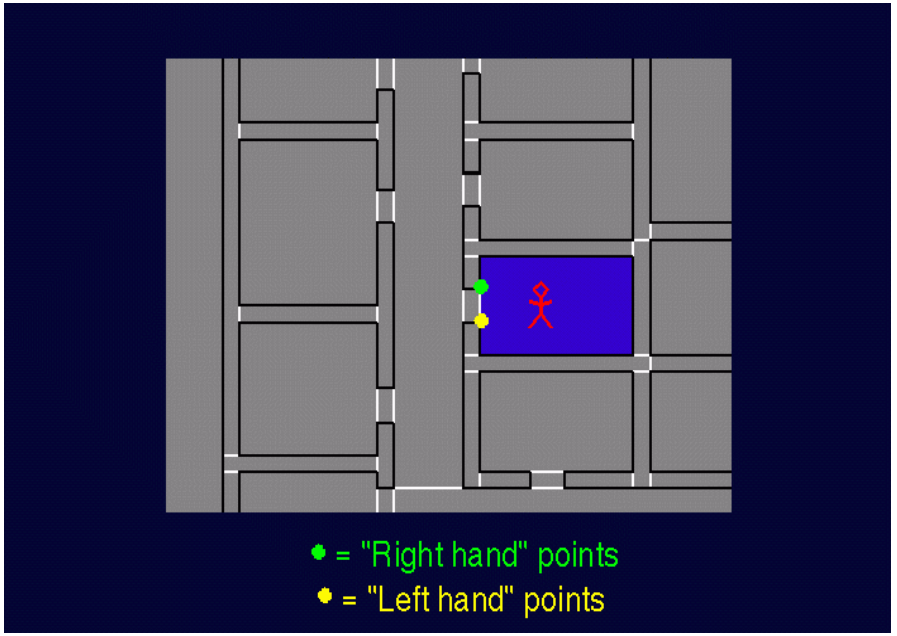
- O objectivo da divisão é identificar regiões convexas disjuntas, potencialmente ligadas por um "portal".
- Portais são, por exemplo, portas ou janelas.
- São também criados portais para garantir que as regiões são convexas.

(Imagens de Seth Teller - MIT)



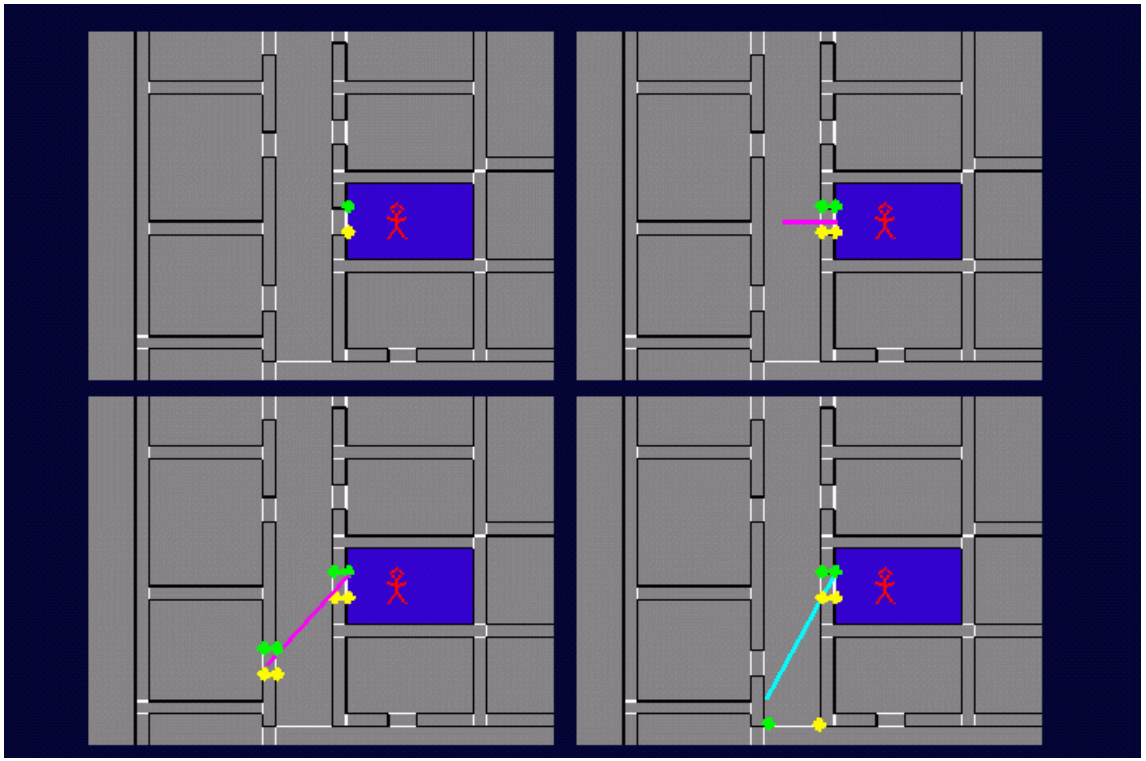
Partição Espacial - Portais

- Como determinar a visibilidade entre células (Teller and Sequin 1991)?



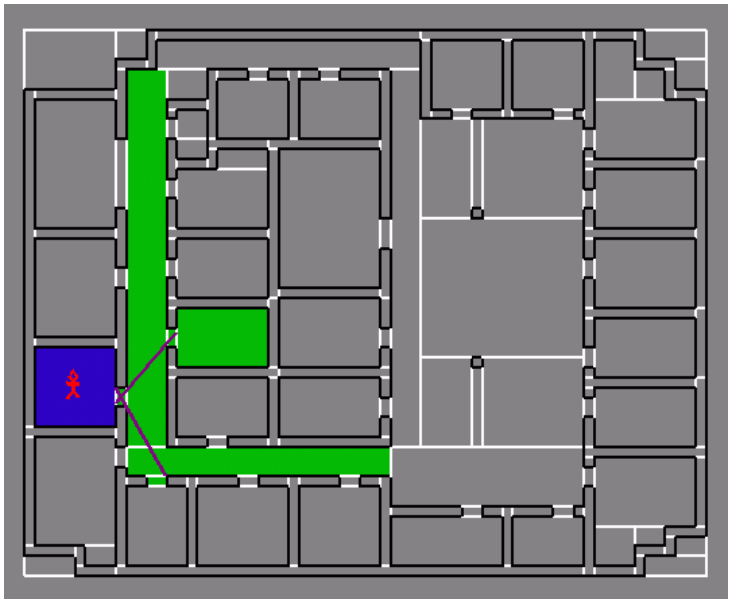
Partição Espacial - Portais

Descobrir linhas que atravessem portais sem no entanto intersectar geometria pelo caminho



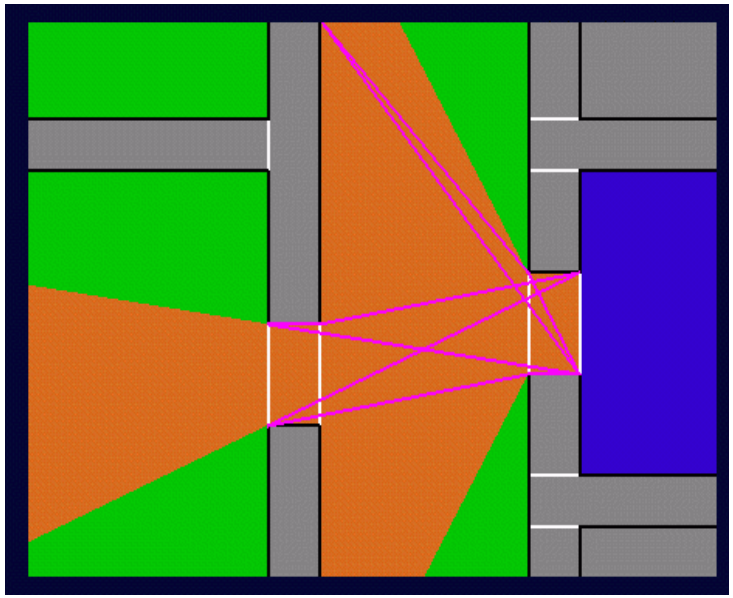
Partição Espacial - Portais

- Visibilidade entre células.
- Cada célula tem portanto associada a si um conjunto de células potencialmente visíveis (PVS).



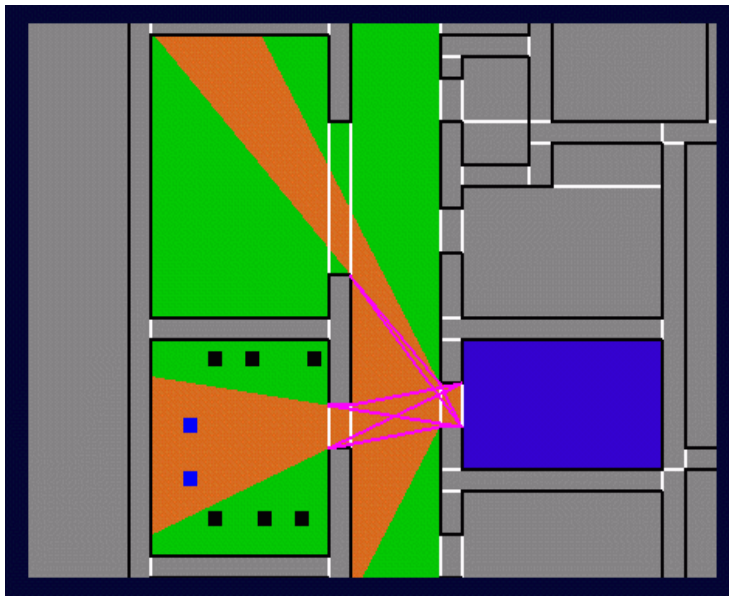
Partição Espacial - Portais

- Um nível de refinamento é possível caso se considere que em vez de células, se considera somente as regiões potencialmente visíveis dentro das mesmas.



Partição Espacial - Portais

- Finalmente, também podemos incluir os objectos presentes na cena.



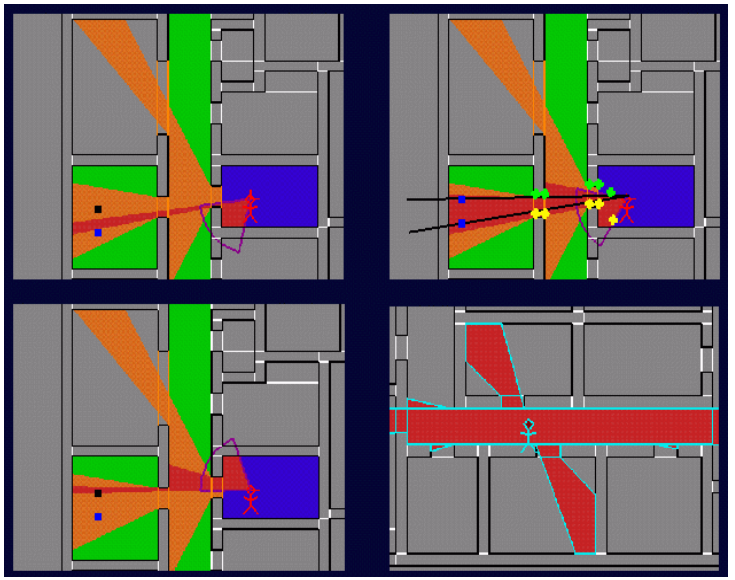
Partição Espacial - Portais

- Todo este processo pode ser realizado offline!
- Não depende da posição da câmara.
- Que informação adicional traz a posição da câmara?

Partição Espacial - Portais

- O View Frustum permite um nível adicional de culling passível de ser realizado em tempo real!(Luebke and Georges 1995)

- Cada vez que o VF atravessa um portal é reduzido pela dimensão do portal.



Partição Espacial - Portais

- Bom para cenas de interiores
  - Resolve de uma só vez visibilidade e oclusão.
- Menos apropriado para exteriores

Resumo

- BSP
- Portais
- **Oclusões**
- Impostores
- Hierarquical Z-Buffer

Oclusões

- Os portais resolvem de uma maneira simples o problema da oclusão para cenas de interiores.

- Como implementar a detecção de oclusões numa octree?

Oclusões

- Prioritized-Layered Projection (PLP)
  - Algoritmo que trabalha com base num "orçamento fixo", por exemplo número de polígonos
  - Determinar os nodos vizinhos do nodo onde se encontra a câmara
  - Ordenar os nodos vizinhos
  - Para o nodo no topo da lista
    - render
    - adicionar os vizinhos à lista
    - reordenar a lista
    - remover nodo

Oclusões

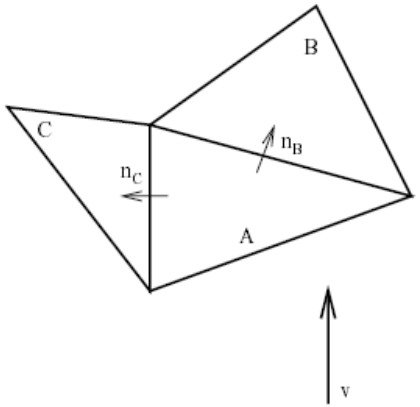
- PLP
  - Pré-processamento:
    - Criar a octree
  - Ordenar (versão simples):
    - Os nodos são ordenados pela distância à câmara
    - Dada a estrutura espacial da octree a ordenação é um processo simples

Oclusões

- PLP:
  - Ordenar nodos por "opacidade"
    - A cada nodo é inicialmente atribuída uma "opacidade" baseada no número de triângulos que contem.
    - Após projectar o nodo A, a opacidade dos seus vizinhos é acumulada tendo em conta o ângulo que a face partilhada forma com a direcção da câmara.

Oclusões

- PLP:
  - Após projectar A as "opacidades" de B e C são actualizadas.
  - B vai acumular mais "opacidade" que C.



Oclusões

- PLP:
  - O algoritmo pára quando é atingido o "orçamento" fixado inicialmente
  - Produz erros na imagem, mas com um "orçamento" razoável podem ser aceitáveis!

Oclusões

- cPLP: conservative PLP
  - Garante a qualidade da imagem final, mas é computacionalmente mais exigente.
  - Contêm teste de visibilidade e só pára quando a imagem está "cheia".
  - Ponto de partida pode ser o resultado do algoritmo PLP.

Oclusões

- cPLP: Teste de visibilidade
  - projecção das caixas dos vizinhos
    - A cada vizinho é atribuida uma cor distinta
  - leitura do color buffer correspondente à zona projectada
  - verificar os nodos visíveis, i.e. as cores que aparecem na imagem
  - Para cada nodo visível desenhá-lo e acrescentar os vizinhos à lista

Oclusões

- cPLP: Aceleração por Hardware
  - OpenGL => OCCLUSION_QUERY
  - Esta funcionalidade permite determinar se um dado volume é visível, indicando inclusivamente o número de pixels visíveis.

Oclusões

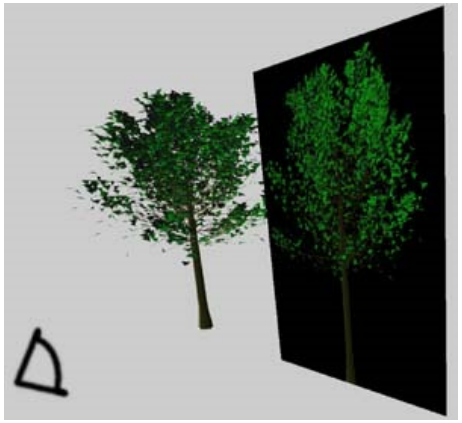
- cPLP:
  - O algoritmo pára quando a lista está vazia, i.e. quando não há mais nodos visíveis.
  - Pode funcionar como "tapa buracos" da versão não conservativa: PLP

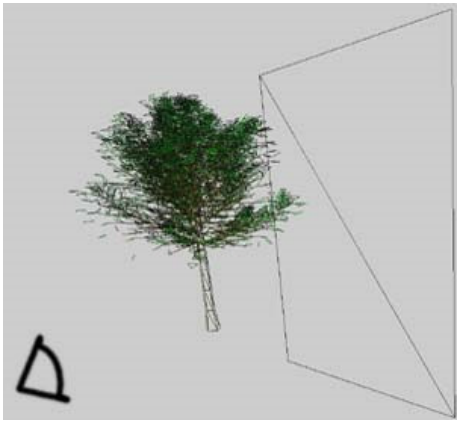
Resumo

- BSP
- Portais
- Oclusões
- **Impostores**
- Hierarquical Z-Buffer

Impostores

- Substituição de Geometria complexa e distante por uma imagem
  - Utilizar a versão geométrica ao perto e a imagem ao longe



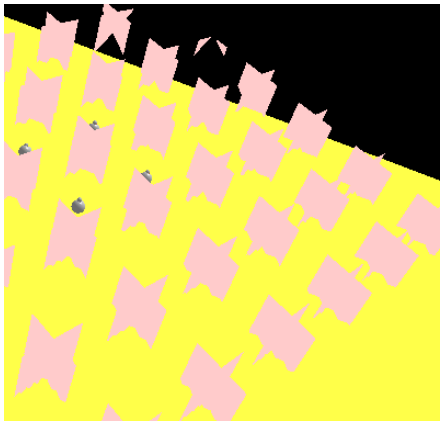


Impostores

- Problema:
  - O que acontece quando a posição do utilizador varia?
  - Por exemplo ao rodar a árvore?

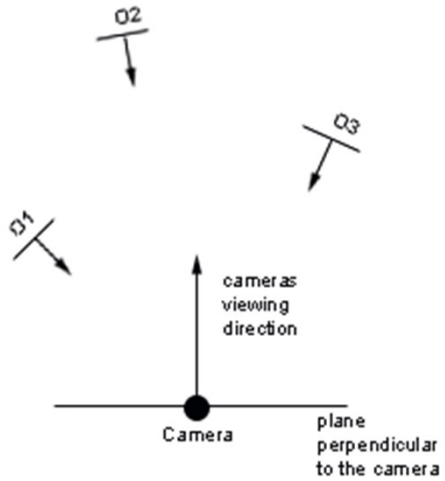
Impostores

- Solução simples:
  - No caso de árvores é comum utilizar dois impostores perpendiculares.



Impostores

- Solução com Billboards:
  - As árvores rodam de forma a estarem sempre viradas para a câmara

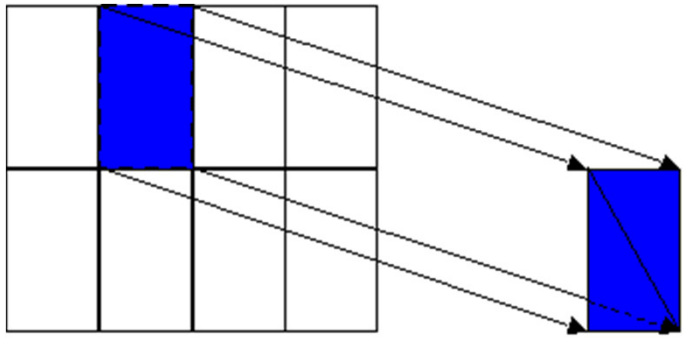


Impostores

- Solução :
  - Obter um conjunto de vistas para a árvore: N, NE, E, SE, S, SW, W, NW.
  - Cada vista é portanto uma orientação do polígono + uma textura
  - Em run-time decidir qual a vista a utilizar.

Impostores

- Questões de Desempenho
  - Para evitar o swap de texturas utilizar uma única textura com as oito vistas.

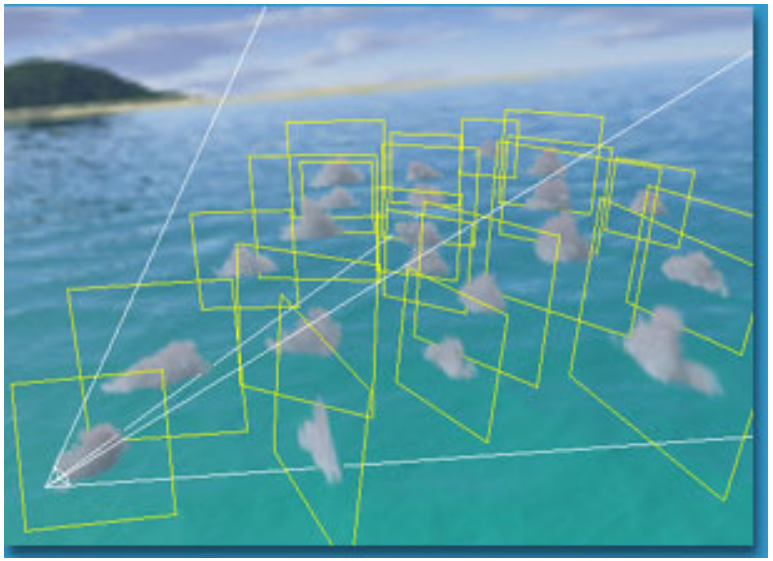


Impostores

- Questões de Desempenho
  - Colocar as oito variantes, posições dos vértices + coordenadas de textura num único Vertex Buffer.
  - Em run-time seleccionar o índice apropriado no VB e desenhar um oitavo (uma variante)

Impostores

- Exemplo de billboards com Nuvens:



Impostores

Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery

Modelo completo

Modelo Local

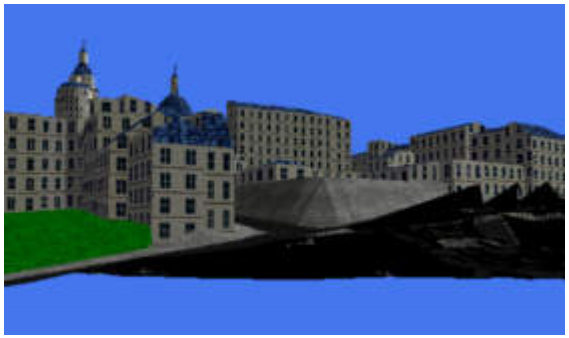
Modelo Distante

Impostor

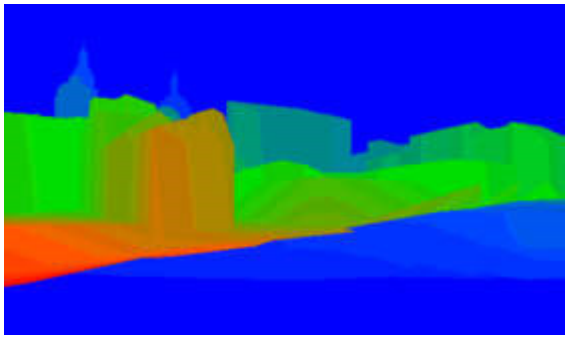
Modelo Local + Impostor

Impostores

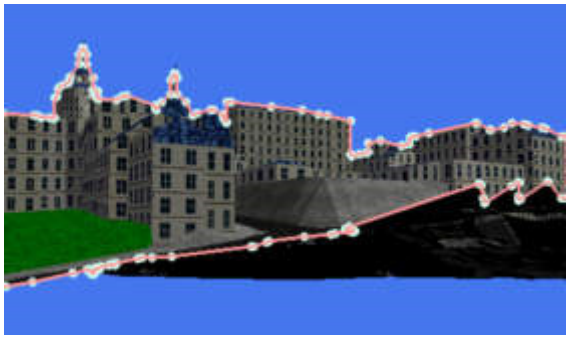
Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery



Impostor

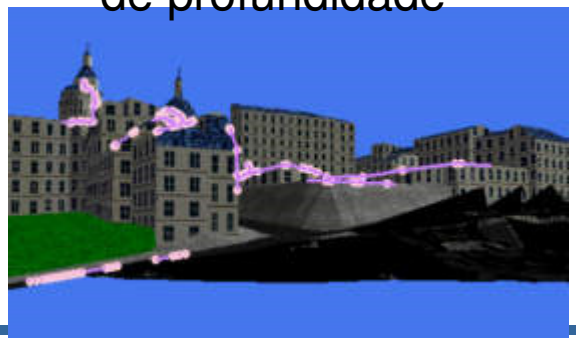


Mapa de Profundidade

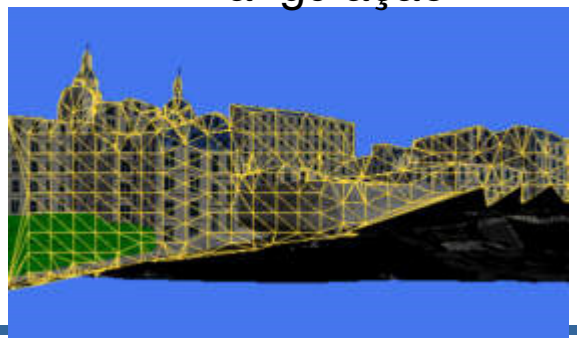


Contorno Externo

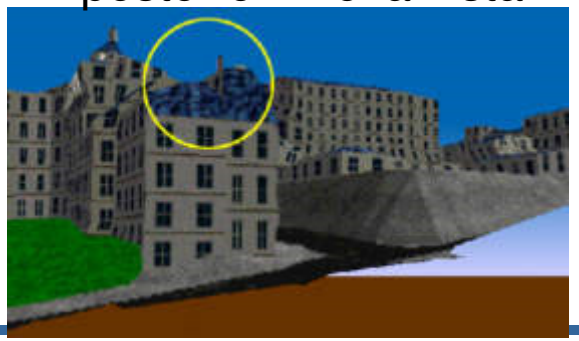
Descontinuidades de profundidade



Triangulação



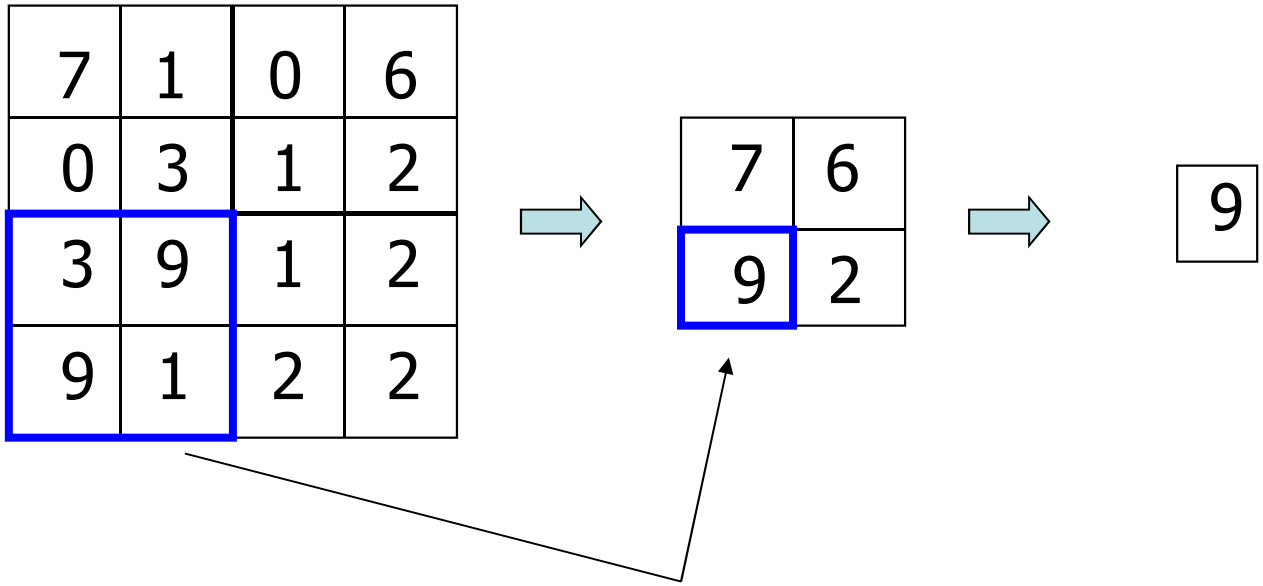
Impostor em nova vista



Resumo

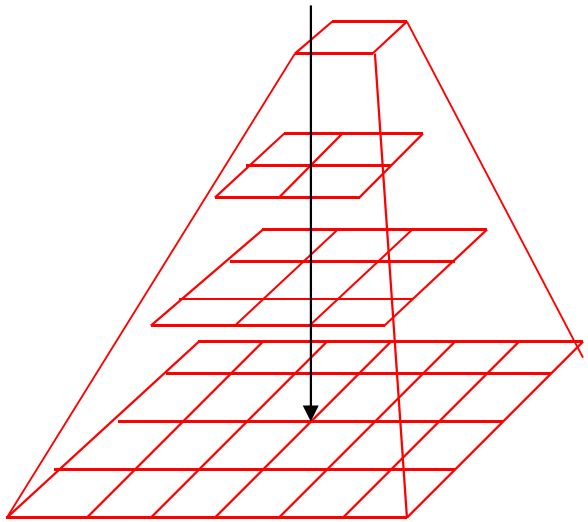
- BSP
- Portais
- Oclusões
- Impostores
- **Hierarquical Z-Buffer**

Hierarchical Z Buffer

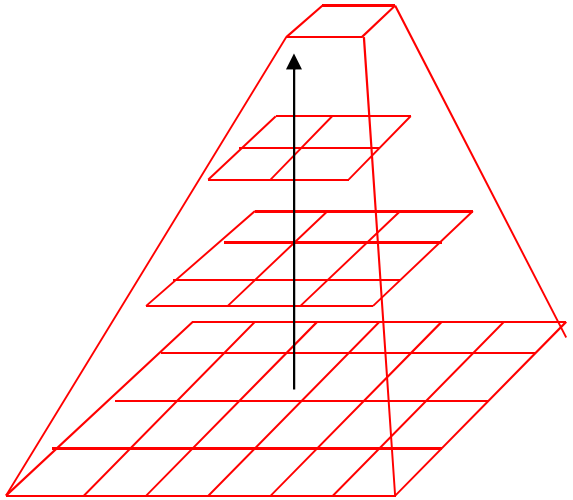


Hierarchical Z Buffer

Test



Update



Referências

- **Visibility Preprocessing For Interactive Walkthroughs (1991),** Seth J. Teller, Carlo H. Sequin

- **Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets,** David P. Luebke and Chris Georges

- ***Real-Time Cloud Rendering,*** Mark J. Harris and Anselmo Lastra, Computer Graphics Forum (Eurographics 2001 Proceedings), 20(3):76-84, September 2001.

- **Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery**
  François X. Sillion, George Drettakis and Benoit Bodelet
  Proceedings of Eurographics'9

- **Efficient Conservative Visibility Culling Using The Prioritized-Layered Projection Algorithm (2000)** James T. Klosowski, Claudio T. Silva, IEEE Transactions on Visualization and Computer Graphics