UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Lucas Rodrigo da Silva Suppes

Algorítmos e Estruturas de dados para visualização de polígonos

Florianópolis
2020

Lucas Rodrigo da Silva Suppes

Algorítmos e Estruturas de dados para visualização de polígonos

Tese submetida ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Ciencias da Computação.
Orientador: Prof. Álvaro Junio Pereira Franco , Dr.

Florianópolis
2020

Ficha de identificação da obra

A ficha de identificação é elaborada pelo próprio autor.

Orientações em:

http://portalbu.ufsc.br/ficha

Lucas Rodrigo da Silva Suppes

Algorítmos e Estruturas de dados para visualização de polígonos

O presente trabalho em nível de Bacharelado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof.(a) xxxx, Dr(a).
Instituição xxxx

Prof.(a) xxxx, Dr(a).
Instituição xxxx

Prof.(a) xxxx, Dr(a).
Instituição xxxx

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Bacharel em Ciencias da Computação.

Coordenação do Programa de
Pós-Graduação

Prof. Álvaro Junio Pereira Franco , Dr.
Orientador

Florianópolis, 2020.

Este trabalho é dedicado aos meus colegas de classe e
aos meus queridos pais.

AGRADECIMENTOS

Inserir os agradecimentos aos colaboradores à execução do trabalho.
Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

“Texto da Epígrafe.
Citação relativa ao tema do trabalho.
É opcional. A epígrafe pode também aparecer
na abertura de cada seção ou capítulo.
Deve ser elaborada de acordo com a NBR 10520.”
(SOBRENOME do autor da epígrafe, ano)

RESUMO

Um estudo das estruturas e técnicas comuns para otimização para renderização em janelas. Tendo em mente o constante crescimento de polígonos dos objetos a serem renderizados, há uma necessidade de que os objetos sejam acessados e "encontrados"de forma rápida e eficiente.

Palavras-chave: Estruturas de Dados. Arvores. Geometria Computacional.

ABSTRACT

A study of the most common tecniques for optimizing rendering on windows. Beign awere of the constant growth of the polygon count and the size of graphical applications, and the necessity for easy and quick access to the objects.

Keywords: Data Structure. Tree. Computational Geometry.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RAM	Memória de Acesso Aleatório
V-RAM	Video RAM
VSD	Visible-Surface Determination

SUMÁRIO

1 INTRODUÇÃO

A contagem de polígonos em artefatos gráficos em filmes, jogos, pesquisas medicas e cientificas seguem em crescimento vertiginosa e apesar do Hardware acompanhar este crescimento, existem claras restrições com relação a Video RAM (V-RAM) e a Memória de Acesso Aleatório (RAM). Com essas restrições em mente, são necessárias estruturas de dados que permitam o acesso rápido dessas figuras, consultas e armazenamento. A seleção e determinação de faces visíveis Visible-Surface Determination (VSD)

1.1 OBJETIVOS

O objetivo deste trabalho visa a busca rápida de estruturas geométricas. Visando aplicações gráficas em tempo real. Quando a câmera da aplicação precisa saber quais figuras geométricas precisam ser desenhadas tendo apenas a informação da posição da câmera e das coordenadas do mundo, este artigo visa o estudo de algoritmos para a solução deste tipo de problema.

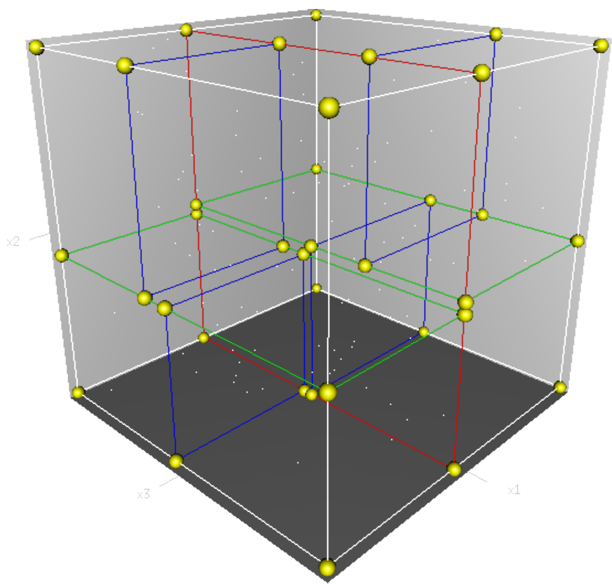
2 FUNDAMENTAÇÃO TEÓRICA

Neste capitulo apresenta-se uma visão de como funcionam os algoritmos e estruturas de dados base para o desenvolvimento de algoritmos para renderização em janela acesso rápido para estruturas geométricas. Sao os algoritmos e estruturas de dados base para a construção de um controle de janela para acesso rapido de poligonos em janela.

2.1 ARVORES KD

Uma Arvore KD é uma arvore binaria onde cada folha é um ponto *k-dimensional*. E cada nodo não-folha é um corte do espaço, representando implicitamente um hiperplano. Pontos a esquerda desse hiperplano estão na subárvore da esquerda, e respectivamente para o lado direito. Cada nodo é associado com uma das *k dimensões*. Então, a citar um exemplo, se dado nodo divide o eixo x, a subárvore a esquerda contem os pontos com o eixo x menor que o ponto de corte.

Figura 1 – Arvore k*dimensional* - 3D



Fonte – GPL

2.1.1 Construção da Árvore

A termos didáticos segue a construção da arvore KD de 2 dimensões. Seja *P* o conjunto de *n* pontos em um plano.

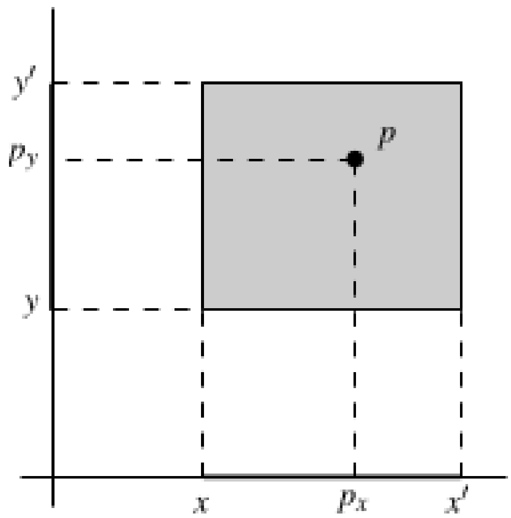
Uma busca de alcance 2-dimensional em P é uma busca de quais pontos da busca estão entre o retângulo de busca $[x,x']X[y,y']$. Um ponto p:= (p_x,p_y) está dentro do retângulo de busca se e somente se:

$$p_x \in [x,x'] \text{ e } p_y \in [y,y']$$

.

Podemos dizer que uma busca 2-dimensional é composta de duas sub-buscas 1-dimensional, uma no eixo x-coordenada de um dos pontos e um na y-coordenada.

Figura 2 – Busca em alcance *dimensional* - 2D

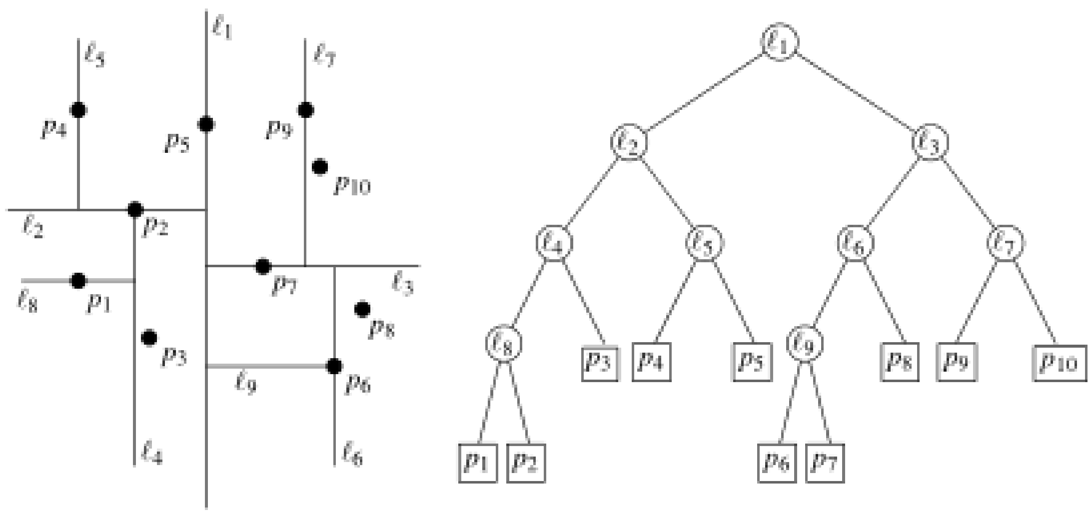


Fonte – Computational Geometry

Na construção de uma arvore para 2 dimensões, cada ponto tem uma x-coordenada e uma y-coordenada. Seguimos então escolhendo um eixo inicial e salvando o valor de corte deste eixo que divide os pontos deste eixo em dois conjuntos, e no próximo nível da arvore, alterna-se o eixo e repete-se o processo recursivamente.

Na raiz ordena-se todos os pontos e divide o conjunto de pontos P com uma linha vertical l que divide os pontos pelo eixo x. Guarda-se o valor de x_v no nodo e alterna-se o eixo. Agora, recursivamente repete o processo para os dois subconjuntos de pontos: Ordena-se os pontos pelo eixo y e divide o conjunto e encontra-se a reta horizontal que subdivide os pontos e guarda no nodo do valor de corte y. A condição de parada é até quando o conjunto de pontos restantes contiver apenas um ponto. Este sendo então o nodo folha.

Figura 3 – Arvore 2D



Algorithm 1 ConstroiArvoreKD(*Pontos*, *profundidade*)

---

if P contem apenas um ponto **then**
  return *P*
else

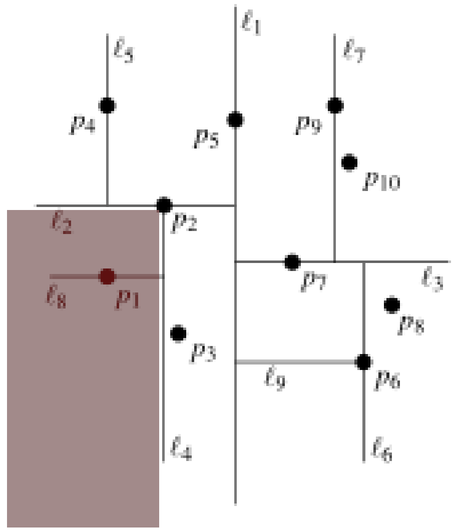
  if *profundidade* é par **then**
    Divide P em dois subconjuntos com um linha vertical *l* pela mediana da $x-$ *coordenada* dos pontos em P. Seja P_1 o conjunto dos pontos à esquerda de *l* e seja P_2 o conjunto de pontos à direita de *l*.
  else
    Divide P em dois subconjuntos com um linha horizontal *l* pela mediana da $y-$ *coordenada* dos pontos em P. Seja P_1 o conjunto dos pontos acima de *l* e seja P_2 o conjunto de pontos à abaixo de *l*.
  end if
end if
$v_{esquerda} \leftarrow$ ConstroiArvoreKD(P_1, *profundidade* + 1)
$v_{direita} \leftarrow$ ConstroiArvoreKD(P_2, *profundidade* + 1) Cria um nodo v guardando l e fazendo v_e*squerda* o nodo da esquerda de v e fazendo v_d*ireita* nodo da direita de v.
return v

2.1.2 Busca com alcance

Agora retomamos para o algoritmo de busca. Podemos imaginar que os pontos na subárvore à esquerda da raiz, estão limitados à direita pela reta com o eixo x com valor de $x \leq l_1$. Enquanto os pontos na subárvore à direita do nodo l_1 estão limitados com o eixo $x > l_1$.

A exemplo: o nodo l_4, a região correspondente de l_4 é limitada à esquerda de l_1 e abaixo de y do nodo l_2.

Figura 4 – Área respectiva de um nodo



Denotaremos esta área de um nodo v como *regiao*(v). A região da raiz é simplesmente (no caso de uma arvore 2D) o plano inteiro. Portanto o algoritmo buscará a subárvore de v somente se o retângulo de busca intersectar a *regiao*(v). O algoritmo de busca funciona descendo a arvore mas visitando somente os nodos que a *regiao*(v) intersecta o retângulo da busca. Quando uma *regiao*(v) esta contido no retângulo de busca retornamos todos os pontos na subárvore. Quando chegarmos nos nodos folhas temos de checar se o nodo esta dentro da busca, se tiver, retorna-o.

Segue o algoritmo que recebe como parâmetros a raiz da arvore-KD e o retângulo de busca R. Usa-se uma chamada *RetornaSubarvore*(v) que atravessa a arvore de nodo v e retorna todos os pontos nas suas folhas. Segue como notação *fe*(v) sendo o filho da esquerda e *fd*(v) o filho da direita do nodo v.

A principal comparação realizada é checar se a área de *Busca* intersecta a região de um nodo v. Para isso precisamos computar *regiao*(v) para todos os nodos v durante a fase de construção da arvore. Uma alternativa é manter a região salva nas chamadas recursivas usando as linhas guardadas nos nodos internos. Por exemplo a região correspondente ao filho esquerda de um nodo v em uma profundidade par (no exemplo 2D, analisamos no eixo x) pode ser calculado com:

$$regiao(fe(v)) = regiao(v) \cap l(v)^{esquerda}$$

, onde $l(v)$ é a linha que divide o eixo salvo em v, e $l(v)^{esquerda}$ é a metade esquerda do plano.

Algorithm 2 BuscaEmArvoreKD(v, *Busca*)

```
if v é folha then
  Retorna o ponto de v se estiver dentro da Busca
else

  if regiao(fe(v)) está contido na Busca then
    RetornaSubarvore(v)
  else

    if regiao(fe(v)) intersecta Busca then
      BuscaEmArvoreKD(fe(v), Busca)
    end if
  end if
  if regiao(fd(v)) está contido na Busca then
    RetornaSubarvore(v)
  else

    if regiao(fd(v)) intersecta Busca then
      BuscaEmArvoreKD(fd(v), Busca)
    end if
  end if
end if
```

2.2 ARVORE DE INTERVALOS

2.3 ARVORE DE SEGMENTOS

3 SEÇÃO

Este *template* contém algumas seções criadas na tentativa de facilitar seu uso. No entanto, não há um limite máximo ou mínimo de seção a ser utilizado no trabalho. Cabe a cada autor definir a quantidade que melhor atenda à sua necessidade.

4 CONCLUSÃO

As conclusões devem responder às questões da pesquisa, em relação aos objetivos e às hipóteses. Devem ser breves, podendo apresentar recomendações e sugestões para trabalhos futuros.

APÊNDICE A – DESCRIÇÃO 1

Textos elaborados pelo autor, a fim de completar a sua argumentação. Deve ser precedido da palavra APÊNDICE, identificada por letras maiúsculas consecutivas, travessão e pelo respectivo título. Utilizam-se letras maiúsculas dobradas quando esgotadas as letras do alfabeto.

ANEXO A – DESCRIÇÃO 2

São documentos não elaborados pelo autor que servem como fundamentação (mapas, leis, estatutos). Deve ser precedido da palavra ANEXO, identificada por letras maiúsculas consecutivas, travessão e pelo respectivo título. Utilizam-se letras maiúsculas dobradas quando esgotadas as letras do alfabeto.