Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Engenharia de Automação e Controle

MATA40 - Estrutura de Dados — Período: 2015.2 — Data: 02/05/2016.
Prof. Antonio L. Apolinário Junior

Segundo Trabalho

Motivação:

Estruturas de dados que permitem identificar, de forma adaptativa, por regiões do espaço 3D onde um determinado objeto se localiza são muito utilizadas em aplicações gráficas. Jogos eletrônicos, por exemplo, podem se valer desse tipo de estrutura para detecção de colisão entre elementos do jogo [5].

A partir de uma caixa envolvente (*Bounding Box*), que determina uma região do espaço onde o objeto se encontra, uma subdivisão do espaço pode ser feita. Esses elementos, denominados ***voxels*** (*volume elements*) são classificados em relação a superfície do objeto: interior, exterior ou fronteira. Elementos interiores ou exteriores não tem influência na colisão, apenas os elementos de fronteira, onde efetivamente a colisão se dá. O processo de subdivisão dos ***voxels*** continua até que se alcance um certo tamanho de ***voxel*** ou nível máximo de refinamento. Na Figura 1 podemos visualizar a relação entre o processo de subdivisão espacial (a esquerda) e a estrutura que representa essa subdivisão.

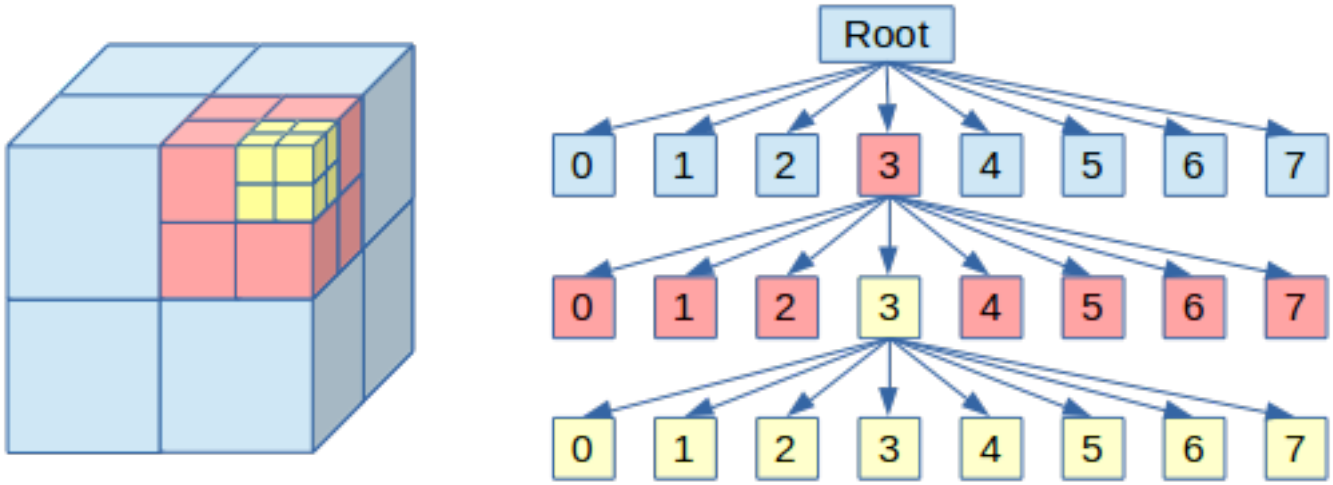


Figura 1 - Estrutura de uma Octree.

Ao final do processo de refinamento, os ***voxels*** classificados como de fronteira representam uma aproximação da superfície do objeto, como mostra a Figura 2. Dessa forma, para determinar se um objeto colide com o outro, um algoritmo “ingênuo" classificaria cada *voxel* de uma subdivisão espacial contra a subdivisão do outro objeto, verificando se há colisão entre seus ***voxels*** [5].

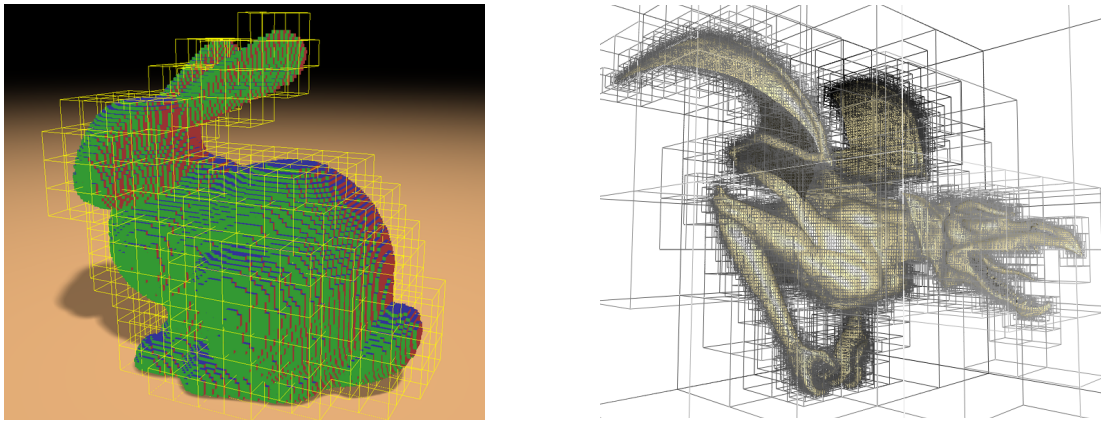
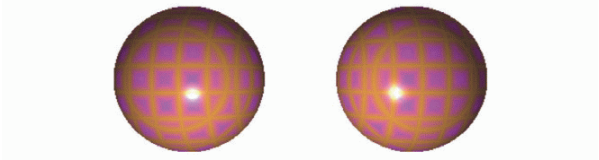


Figura 2 - Aproximação da superfície de um objeto por uma Octree.

Para que o esquema acima descrito funcione é necessário ter uma forma simples de classificar voxels quanto a sua posição relativa em relação a superfície de um objeto. Um maneira de descrever a superfície de um objeto é através de uma equação matemática implícita do tipo $F(x,y,z) = 0$. Essa equação permite, dado um ponto $P(x_p, y_p, z_p)$, substituir suas coordenadas e determinar se: $F(x_p, y_p, z_p) = 0$, então o ponto localiza-se na superfície do objeto; $F(x_p, y_p, z_p) < 0$, então o ponto P localiza-se no interior do objeto; e finalmente se $F(x_p, y_p, z_p) > 0$, então o ponto localiza-se no exterior do objeto. Na Figura 3 à esquerda podemos ver um exemplo de duas equações implícitas de 2 esferas deslocadas no eixo Y de +- 1.5.

Portanto, dada a equação implícita de um objeto, a construção de uma subdivisão espacial pode ser guiada pela avaliação dos vértices da cada ***voxel***. Caso todos sejam positivos ou negativos o ***voxel*** pode ser classificado como, respectivamente, externo ou interno. Caso haja classificações alternadas, positivas/negativas, a superfície do objeto passa pelo interior do ***voxel***. Esse processo é ilustrado na Figura 3, à direita, onde os elementos da subdivisão espacial refinados se concentram na superfície do objeto.

$$f_1(\mathbf{x}) = 1 - (x^2 + (y + 1.5)^2 + z^2)$$
$$f_2(\mathbf{x}) = 1 - (x^2 + (y - 1.5)^2 + z^2)$$



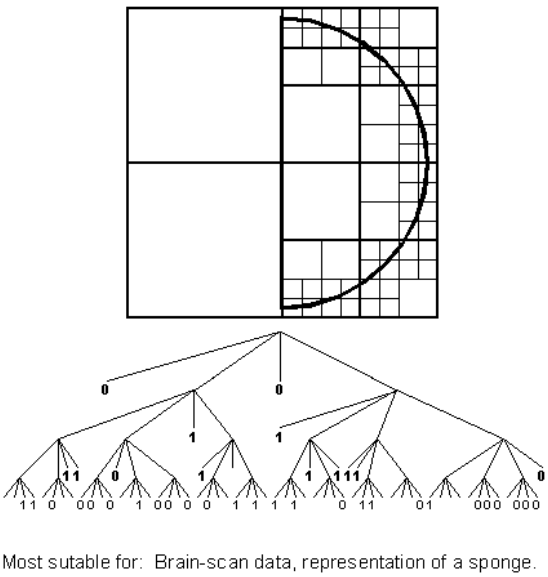


Figura 3 - À esquerda exemplos de equações implícitas que descrevem as duas esferas de raio 1. À direita uma representação em 2D de uma estrutura em árvore que descreve a superfície de uma semicircunferência.

Objetivos do trabalho:

O objetivo principal do trabalho é aplicar os conceitos de Tipos Abstratos de Dados (TAD) [1] [2][3] apresentados em sala de aula, no contexto de um problema real e prático.

Os alunos deverão analisar o problema proposto, e definir como estruturar as informações necessárias para a resolução do problema no TAD proposto.

Definido o TADs, sua implementação deverá ser integrada ao problema e uma solução computacional codificada em linguagem C.

O Problema:

O problema é desenvolver um programa que constrói a subdivisão do espaço em ***voxel*** a partir da definição implícita de um objeto. O objeto poderá ser visualizado pela aproximação gerada pelo conjunto de ***voxels*** que passam por sua fronteira.

Seu programa, portanto, deve:

1. Gerar uma subdivisão espacial de um objeto definido na forma implícita;
2. Apresentar a visualização desse objeto, como um conjunto de ***voxels*** que representam uma aproximação de sua superfície;
3. Oferecer ao usuário a possibilidade de modificar em tempo real o nível de refinamento em que deseja visualiza a aproximação do objeto (respeitando o limite máximo definido na criação da subdivisão).
4. Permitir ao usuário que indique, via linha de comando, qual o objeto implícito deseja visualizar e qual o nível máximo[1] de subdivisão será gerado. Forneça pelo menos 3 opções para os objetos[2]. Caso o usuário não forneça essas definições de parâmetros, valores *default* devem ser utilizados;

A Implementação:

A implementação deve ser desenvolvida em linguagem C ANSI (independente de qualquer IDE ou SO) e tomar por base o código fonte base fornecido pelo professor. Esse código fonte está divido em alguns módulos, a saber:

- **EstruturasDeDados.h**
  Descrição das estruturas de dados a serem utilizadas nesse trabalho;
- **winGL.***
  Rotinas responsáveis pelo controle das janelas e dos desenhos;
- **trabalho.***
  Programa principal e rotinas de tratamento de eventos de teclado e desenho.

Os TADs que voce implementar devem ser definidos em um módulo próprio. Sua codificação não deve estar “misturada” ao módulo do programa principal (**trabalho**).

---

[1] Esse valor pode ser ilimitado? Caso não seja estabeleça um limite superior que o usuário não pode ultrapassar.

[2] Sugestões de funções matemáticas que geram objetos implícitos interessantes podem ser encontradas em http://paulbourke.net/geometry/. Cuidado pois nem todas as formas são descritas na forma F(x,y,z) = 0. Na duvida consulte seu professor.

Para que você possa utilizar a biblioteca gráfica **glut**, você deve baixar os arquivos *.h e ***libs*** e instala-los/copia-los para os diretórios do seu compilador. Esse processo e o arquivo dessa ***lib*** para download podem ser encontrados em [4].

Você deverá definir seu TAD baseado na estrutura de dados inicial, definida no módulo **EstruturasDeDados.h**. **Não é permitido modificar essa estrutura. Caso ache necessário, consulte o professor para saber se sua modificação é válida.**

Os trabalhos deverão ser desenvolvidos individualmente. O código fonte gerado deve ser comentado e legível. Acompanhando o código fonte um breve relatório técnico deve ser entregue, descrevendo as estruturas de dados utilizadas, uma justificativa para seu uso e quais testes foram realizados para validar o funcionamento do programa (descrição, objetivo e resultado).

A Entrega:

O trabalho deverá ser submetido via *Moodle*, respeitando a data e hora limite para entrega. Em caso de atraso, será aplicado um fator de penalização de 1,0 ponto por dia de atraso. Qualquer problema de arquivos corrompidos ou similar o trabalho será considerado não entregue. Portanto, verifique bem o que for entregar.

Os arquivos devem ser enviados seguindo o seguinte padrão: arquivo compactado (zip, rar, tgz ou gzip apenas) contendo um diretório com o nome do(s) aluno(s) e seu arquivos. Arquivos fora desse padrão sofrerão penalização de 0,5 ponto na nota final. Códigos com erros de compilação ou qualquer outra pendência que impeça a compilação não serão avaliados.

Uma aula será destinada a apresentação dos trabalhos, onde cada aluno mostrará ao professor o que foi feito.

A cooperação entre alunos e grupos é considerada salutar. No entanto, trabalhos com alto grau de similaridade serão tratados como “plágio”, o que resultará em avaliação ***zero*** **para todos os envolvidos**.

Qualquer dúvida adicional, evite problemas: não presuma nada, procure o professor para esclarecimentos.

Referencias Bibliográficas:

[1] Ziviani, Nivio. **Projeto de Algoritmos: com Implementações em Pascal e C**. Vol. 2. Thomson, 2004.

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. **Introdução a algoritmos**. 2001.

[3] Sedgewick, Robert. **Algorithms in C++**. Vol. 2. Pearson Education India, 2003.

[4] Martin Payne, ***Using freeglut or GLUT with MinGW***, seção "***Setting Up GLUT for Win32 With MinGW”***, disponível em: http://www.transmissionzero.co.uk/computing/using-glut-with-mingw/.

[5] Michael Kelleghan, **Octree Partitioning Techniques**. Disponível em: http://www.gamasutra.com/view/feature/131625/octree_partitioning_techniques.php