Universidade de São Paulo – ICMC
Bacharelado em Ciência da Computação
SCC0201 – Introdução à Ciência da Computação II
Professores: Prof. Moacir Ponti – moacir@icmc.usp.br ,
Prof. Rodrigo Fernandes de Mello – mello@icmc.usp.br
Monitores: Lucas Pagliosa– carvalho.pagliosa@gmail.com,
Ricardo Fuzeto – ricardofuzeto@gmail.com,
Yule Vaz – yule.vaz@usp.br

Trabalho 1 - Octree

1 Descrição

Na computação gráfica, os objetos tridimensionais são definidos como um conjunto de triângulos que formam a superfície de uma malha, tal como ilustrado na Figura 1, que apresenta uma forma toroidal a partir desses componentes. Quanto maior a quantidade de triângulos, melhor é a resolução de um determinado objeto e, por conseguinte, a qualidade do gráfico. Nesse sentido, imagine o problema de determinar se há a colisão entre dois objetos. Obviamente, a colisão é detectada quando seus triângulos se interseccionam de alguma forma. Contudo, se for verificada essa intersecção analisando todos esses elementos, o custo computacional de aferir uma colisão aumenta quão maior a resolução dos elementos gráficos. Neste sentido, emprega-se, tipicamente, uma Octree: um tipo de árvore que reduz o custo computacional para o tratamento de colisões.

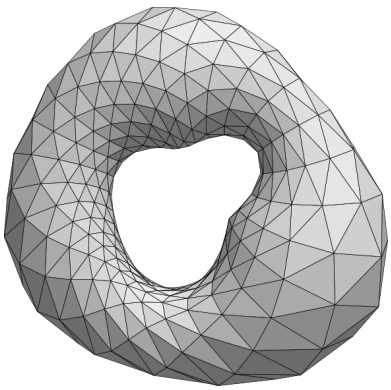

Figura 1: Malha de um objeto toroidal construída a partir de triângulos (Extraído de https://en.wikipedia.org/wiki/Triangulation_(topology)).

1.1 Árvores

Árvores são estrutura de dados sobre as quais pode ser definida uma "hierarquia" entre seus elementos. Nesse sentido, pode-se dizer que há uma ordem de maneira que alguns elementos se encontram acima ou abaixo de outros tal como ilustrado na Figura 2. Mais precisamente, uma árvore é composta de nodos que armazenam algum tipo de dado. Além disso, como ilustrado na Figura 2, pode-se dividir a árvore em níveis. O nodo de nível zero em sua hierarquia é denominado de raíz e os do último nível de folhas. Além disso, dado um nodo X_p e um X_f, se X_p está logo acima de X_f ele é dito como sendo pai de X_f e X_f como sendo filho de X_p. Como indicado na Figura 2, o nodo de índice 3 é filho do nodo 1 e pai dos nodos 6 e 7. Note que um nodo filho pode ser vinculado apenas a um nodo pai.

Essa estrutura permite dividir um determinado problema em subproblemas menores de maneira a reduzir o custo computacional ao procurar a sua solução. Por exemplo, suponha um conjunto de palavras $S = \{\text{abacate}, \text{abóbora}, \text{acerola}, \text{banana}, \text{beringela}\}$ e um problema cujo objetivo é buscar uma

determinada cadeia de caracteres (*string*) P nesse conjunto. Uma primeira solução para esse problema é criar uma lista a partir do conjunto S e, elemento por elemento, verificar se há compatibilidade com P. Contudo, imagine que o conjunto S é composto por um bilhão de palavras. Nesse sentido, caso a palavra requerida esteja na última posição, o algoritmo de busca deverá efetuar um bilhão de verificações e despenderá de uma grande quantidade de tempo para sua execução.

Assim, pode-se construir uma árvore de maneira que o nodo do nível 0 represente uma string vazia, os nodos do nível 1 representem strings de 1 caracter, os do nível 2 representem uma string de 2 caracteres e assim sucessivamente. Além disso, os elementos de S serão os nodos folha dessa árvore tal como ilustrado na Figura 3. Então, para buscar alguma palavra, verifica-se qual seu primeiro caracter. Então, sabe-se que ela está "abaixo" do nodo que representa esse caracter. Verifica-se, posteriormente, qual o seu segundo caracter e, assim, sabe-se também que ela está abaixo não só do nodo que representa o primeiro caracter como também do que representa o segundo. Isso é feito até se encontrar a palavra ou averiguar que ela não pertence ao conjunto S.

Assim, suponha que é desejado buscar a palavra "banana". Obviamente, pode-se dizer que existe uma string vazia antes desta palavra, assim como em todas as outras existentes. Verifica-se então, qual seu primeiro caracter (no caso "b"). Portanto, o nodo "banana" está abaixo do nodo "b" – note que, com isso, elimina-se a busca sobre as strings "abacate", "abóbora" e "acerola" e, portanto, reduziu-se o custo computacional do problema. Posteriormente, é verificado o segundo caracter da palavra em questão, que é "a". Portanto, sabe-se que "banana" está abaixo do nodo "ba". Assim, como o nodo "banana" é o único filho do nodo "ba" a busca é concluída. Suponha agora que deseja-se encontrar a palavra "bala". Como o exemplo anterior, é verificado que o nodo referente a esse termo está abaixo do "b" e do "ba". Contudo, o nodo filho de "ba" é "banana" e portanto "bala" não pertence a S. Essa estrutura em particular é conhecida como *tries* (Sedgewick, 1990).

Cabe ressaltar que, no contexto de implementação na linguagem de programação C, uma árvore pode ser definida por uma estrutura `tree` a qual contém uma variável que aponta para o nodo raiz.

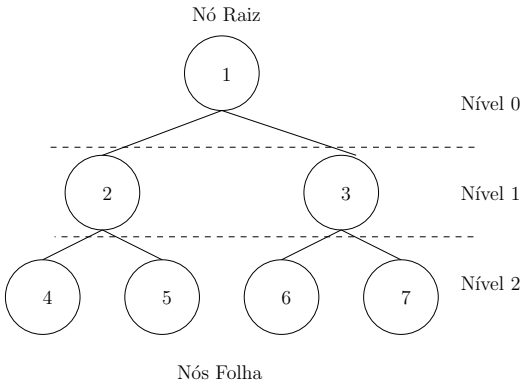

Figura 2: Exemplo de uma estrutura de dados do tipo árvore contendo nodos de índice $1, 2, 3, 4, 5, 6$ e 7.

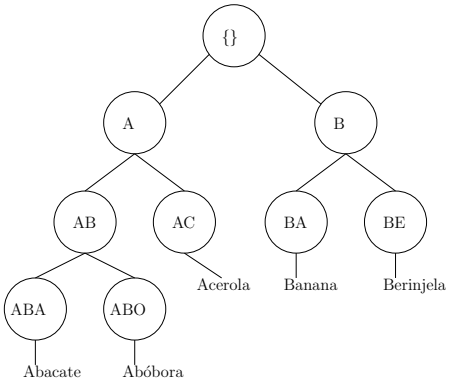

Figura 3: Exemplo de uma estrutura de dados do tipo árvore utilizada para a busca de palavras.

Além disso, ela pode armazenar outras informações tais como a quantidade de nodos e o seu nível máximo. Além disso, seus nodos podem ser implementados a partir de uma estrutura `node` que contém o seu respectivo dado e ponteiros para seus respectivos filhos. Note que se a quantidade de filhos variar de um nodo para o outro então a estrutura `node` deve conter uma lista de ponteiros. No contexto deste trabalho, um vetor de ponteiros com tamanho fixo será suficiente. Uma implementação dessas estruturas segue abaixo:

```
typedef struct{

        int size;
        int levels;
        node *root;

} tree;

typedef struct {

        int data;
        node **children;

} node;
```

1.2 Octree e Caixa Delimitadora

Neste trabalho, os elementos considerados serão tridimensionais e deve-se detectar a colisão entre um ponto p (objeto colisor) e um objeto O qualquer (objeto alvo) composto a partir de um conjunto de vértices. Primeiramente, para a construção da octree, tal como ilustrado na Figura 5, o objeto alvo O deve ser delimitado por um paralelepípedo, i.e., uma caixa delimitadora. Uma caixa delimitadora é definida como o paralelepípedo de menor volume que contenha o objeto em questão. Neste trabalho, ela pode ou não ser dada como parâmetro de entrada.

Para definir a caixa delimitadora deve-se estipular os limites do objeto alvo, i.e., dado um conjunto de vértices $V = (x_i, y_i, z_i)_{i=1,\dots,n}$ que formam as malhas do elemento gráfico, encontram-se os mínimos e máximos ($p_{min} = (x_{min}, y_{min}, z_{min})$ e $p_{max} = (x_{max}, y_{max}, z_{max})$) de V. A partir de p_{min} e p_{max}, a caixa delimitadora será formada por todas as composições possíveis dos elementos de p_{min} e p_{max}, i.e., por $(x_{min}, y_{min}, z_{min})$, $(x_{min}, y_{min}, z_{max})$, $(x_{min}, y_{max}, z_{min})$, $(x_{min}, y_{max}, z_{max})$, $(x_{max}, y_{min}, z_{min})$, $(x_{max}, y_{min}, z_{max})$, $(x_{max}, y_{max}, z_{min})$ e, finalmente, $(x_{max}, y_{max}, z_{max})$. Por exemplo, suponha o triângulo composto pelos vértices $(0, 0, 0)$, $(0, 1, 1)$ e $(1, 1, 0)$. Tem-se que x_{min}, y_{min} e z_{min} são iguais a zero e x_{max}, y_{max} e z_{max} são iguais a 1. Assim, a caixa delimitadora deste triângulo será formada por $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 0)$, $(0, 1, 1)$, $(1, 0, 0)$, $(1, 0, 1)$, $(1, 1, 0)$ e $(1, 1, 1)$ tal como ilustra a Figura 4.

Assim, de uma maneira procedimental, é verificado se p está contido dentro da caixa delimitadora de O. Caso não esteja, não é detectada a colisão e, caso esteja, o paralelepípedo é dividido em octantes, dentre os quais é selecionado aquele ao qual p pertence. Dado o octante selecionado, se este não apresenta intersecção com O, então não há colisão. Caso contrário, esse octante é subdivido em outros e é novamente selecionado aquele que contém o objeto colisor p assim como também é verificado se o objeto alvo O intersecciona a região em questão. Isso é feito sucessivamente até um nível pré-determinado o qual será dado como parâmetro de entrada para o programa. Se ao atingir esse nível ambos os objetos interseccionam o octante, a colisão é detectada. Isso é ilustrado na Figura 6.

Nesse sentido cada nodo da Octree representa um octante cujos filhos representam as regiões formadas pela subdivisão do octante "pai" e, portanto, cada nodo tem oito filhos, exceto pelas folhas. Assim, essas estruturas deverão armazenar os vértices de seu respectivo octante e se algum dos triângulos que

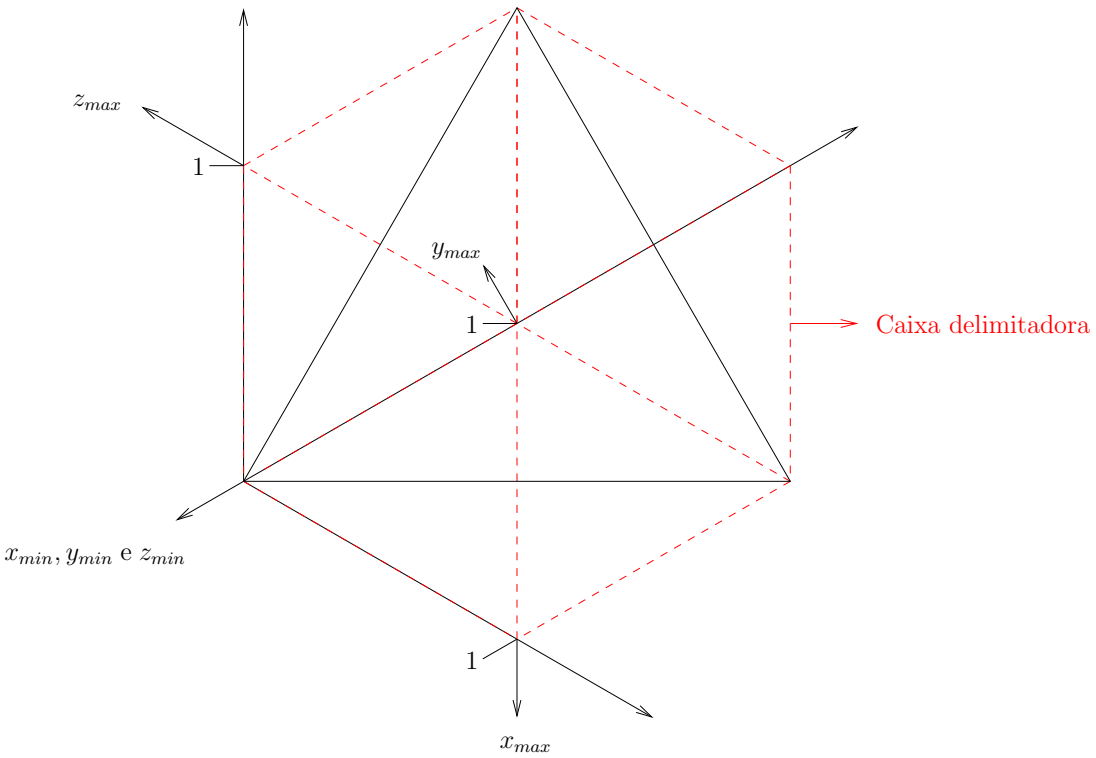

Figura 4: Caixa delimitadora de um triângulo de vértices $(0, 0, 0)$, $(0, 1, 1)$ e $(1, 1, 0)$.

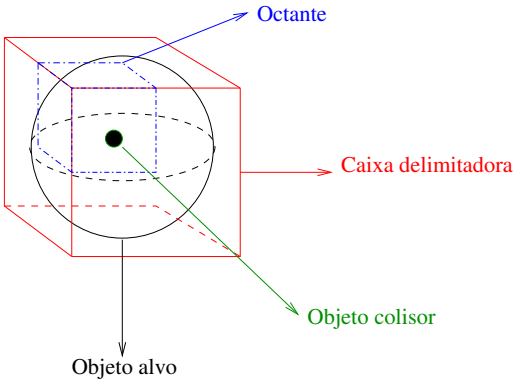

Figura 5: Representação de uma esfera como objeto alvo e sua respectiva caixa delimitadora bem como o primeiro octante a ser escolhido a partir da posição do objeto colisor.

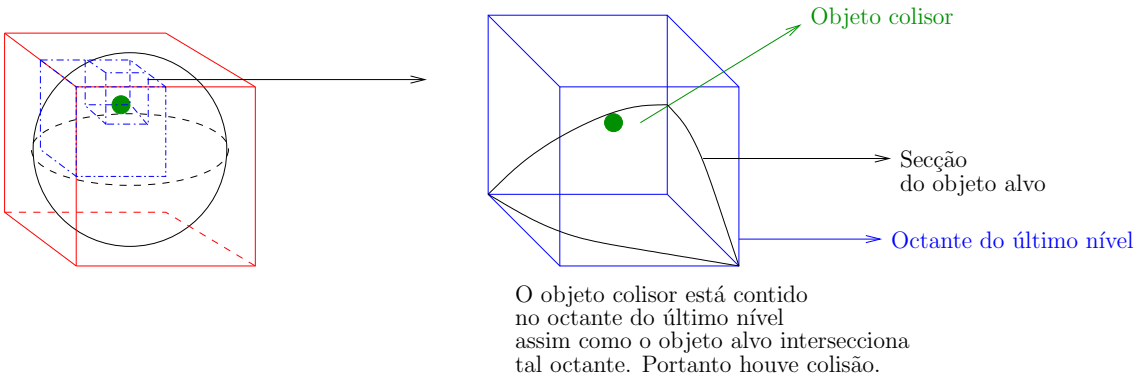

Figura 6: Detecção de colisão em uma octree de nível 2.

compoem o objeto alvo intersecciona a região em questão. Para aferir se o ponto p colidiu ou não com o alvo, deve-se verificar, primeiramente, se a sua posição está contida dentro da região armazenada no primeiro nodo. Se não está contida então não houve colisão, caso contrário é selecionado o nodo filho N_s o qual armazena o octante que contém a posição de p e é verificado se ele intersecciona o objeto alvo. Se não há intersecção então não houve colisão. Caso contrário, repete-se o procedimento para o nodo filho N_s. Isso é feito até que o último nível da árvore seja atingido e detecta-se a colisão caso, nesse último nível, há intersecção com o objeto alvo.

Implemente um programa que, a partir de uma octree, verifique se houve ou não a colisão entre dois objetos.

2 Informações Complementares

- Seu programa deverá receber como entrada, primeiramente, o valor 1 caso a caixa delimitadora deva ser implementada ou o valor 0 caso contrário.
- Caso o valor seja 0, seu programa receberá: i) o nível n da octree, ii) a quantidade de triângulos que descrevem o objeto alvo, iii) o ponto (descrito em três dimensões) que representará o objeto colisor, iv) a lista dos triângulos que constituem o objeto alvo e, v) os vértices da caixa delimitadora.
- Caso o valor seja 1, seu programa receberá: i) o nível n da octree, ii) a quantidade de triângulos que descrevem o objeto alvo, iii) o ponto (descrito em três dimensões) que representará o objeto colisor e iv) a lista de triângulos que constituem o objeto alvo.
- Seu programa deverá fornecer como saída o número 0 se não houver colisão e o 1 caso haja;
- Utilize recursão para a busca dentro da octree;
- A memória deve ser alocada dinamicamente.

Referências

Sedgewick, R. (1990). *Algorithms in C*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Entrada	Saída

0
4
2
0.9 0 0
//Malha 1
1 0 0
0 0 1
0 1 0
//Malha 2
0 0 0
1 0 0
1 1 0
//Caixa delimitadora
0 0 0
1 0 0
0 1 0
0 0 1
1 1 0
1 0 1
0 1 1
1 1 1
	1

Tabela 1: Tabela dos dados de entrada e saída do programa considerando que a caixa delimitadora será dada. Desconsidere os comentários como sendo entrada do programa, ele é meramente explicativo.

Entrada	Saída

0
4
3
0.9 0 0
//Malha 1
1 0 0
0 0 1
0 1 0
//Malha 2
0 0 0
1 0 0
1 1 0
	1

Tabela 2: Tabela dos dados de entrada e saída do programa considerando que a caixa delimitadora não será dada. Desconsidere os comentários como sendo entrada do programa, ele é meramente explicativo.