Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Sistemas de Informação

MATD04 - Estrutura de Dados Período: 2018.1 Data: 28/06/2018.
Prof. Antonio L. Apolinário Junior

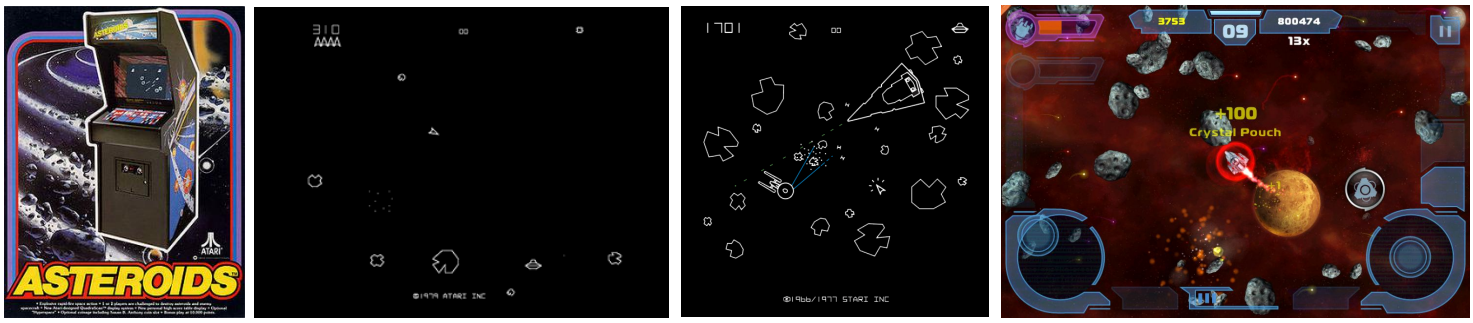
Trabalho Final - Asteroides em Colisão!

Motivação:

Jogos eletrônicos costumam demandar por estruturas de dados eficientes e robustas em várias de suas etapas [1]. Mesmo jogos simples podem requerer estruturas de dados apropriadas para que o roteiro do jogo possa ser implementado de forma eficiente e com desempenho fluido e constante.

Um jogo de bastante sucesso nos primeiros consoles de *Arcade* foi o "*Asteroids*"[1]. Como vemos na Figura 1(b), esse jogo, em sua versão original (de 1979), consiste de uma nave no espaço, representada por um triangulo, rodeada de asteróides, na forma de poligonos. Para evitar ser destruida a nave deve navegar entre os asteróides (que se movem em direções aleatórias), desviando deles e atirando para destrui-los. Os controles são simples e se resumem em impulsionar a nave para frente e faze-la girar nos dois sentidos, horário e anti-horário. Versões mais "sofisticadas" foram criadas, com objetos mais "complexos", como na Figura 1(c), até versões 3D, como a da Figura 1(d).

Como podemos perceber, o maior desafio computacional do jogo é a detecção de colisão entre os objetos. Para destruir um asteróide, os tiros da nave devem atingir os asteróides; para que a nave seja destruída ela deve colidir com um asteróide; asteróides ao colidirem entre si mudam sua trajetória.



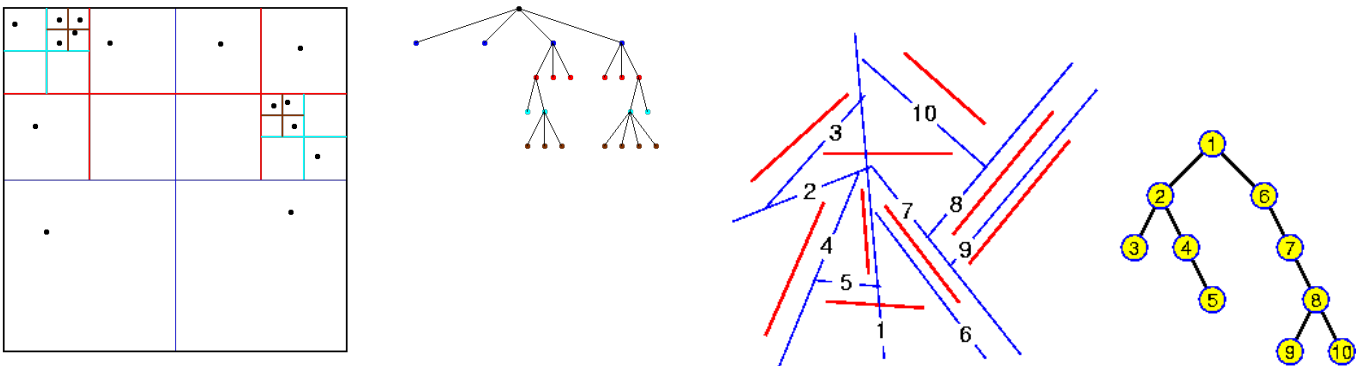
(a) *(b)* *(c)* *(d)*

Figura 2 - Jogo Asteroides: (a) console original; (b) versão básica; (c) naves mais sofisticadas; (d) versão 3D.

[1] https://en.wikipedia.org/wiki/Asteroids_(video_game)

Uma das operações básicas em qualquer jogo eletronico é a sua capacidade de detectar se 2 objetos colidem. Seja em um jogo 2D ou 3D, várias ações como pegar, atirar, arremessar, entre outras, dependem da correta identificação do contato entre os objetos.

Para resolver esse tipo de problema, diversas estruturas de dados que particionam o espaço onde os objetos se encontram já foram propostas na literatura. Duas que merecem destaque são as **QuadTrees** [5] e as **Partições Binárias do Espaço** (***Binary Space Partitioning*** - ***BSP***) [4], As duas estruturas podem ser visualizadas na Figura 1. Como podemos perceber, ambas utilizam como base uma estrutura hierárquica para construir uma subdivisão do espaço, na qual os objetos podem ser localizados. Essa estrutura hierárquica é uma **árvore**.



(a) *(b)*

Figura 1 - Estruturas de dados para partição do espaço: (a) ***QuadTree****; (b)* ***BSP****.*

Objetivo do Problema:

O objetivo principal do problema é construir e analisar um **Tipo Abstrato de Dado** (TAD) [1][2] [3] baseado nos conceitos de Á**rvores**, aplicados a um problema real.

Os alunos deverão analisar os 2 tipos de árvores sugeridos para resolver o problema de detecção de colisão, e escolher um deles para implementar e analisar o comportamento da estrutura e seus algoritmos.

O Problema:

Inicialmente o protótipo do jogo deve criar, de forma aleatória, um conjunto de **N** asteróides (considere um valor *default*, mas tambem a possibilidade do usuário fornecer o numero inicial pelo teclado na chamada do programa). Cada Asteróide tem uma posição inicial distinta (ou seja, inicialmente não há colisão entre eles). Cada asteróide será representado simplificadamente por um circulo, definido por sua posição, seu raio e cor[2]. Tambem de forma aleatória deve ser definido um vetor de velocidade, que fará o asteróide se mover ao longo do tempo.

Após a inicialização os asteróides se movimentam livremente segundo a direção definida na sua criação. O jogo deve detectar sempre que os asteróides colidirem. Nesse caso, 3 ações podem ser tomadas:

1. Os asteroides se juntam e formam um único, se deslocando na direção da soma das direções iniciais;
2. Os asteroides se fragmentam gerando outros menores, seguindo direções opostas;
3. Os asteróides se vaporizam e somem da tela.

A situação 1 acontece quando os asteroides se chocam quase frontalmente (angulo entre as direções proximo de 90^o). A situação 2 acontece quando os asteroides se chocam “lateralmente" (angulo entre as direções abaixo de 60^o). E finalmente a situação 3, quando o choque se dá frontalmente (angulo entre as direções abaixo de 180^o).

Para facilitar o acompanhamento do processo, o protótipo deve permitir que a estrutura de dados seja desenhada na tela ao longo da execução do programa.

A Avaliação:

Os itens a serem avaliados nesse problema são:

Item	Valor
Uso de modularização para o desenvolvimento da aplicação	0.5
Uso correto dos recursos da linguagem C (variáveis locais, passagem de parametros, ponteiros,….)	0.5
Implementação correta da estrutura escolhoda	4.0
Visualização da estrutura de dados durante a execução do protótipo	3.0
Tratamento correto da colisão	2.0
Total	**10.0**

A Implementação:

A implementação deve ser desenvolvida em linguagem C ANSI (independente de qualquer IDE ou SO). A parte gráfica do trabalho (utilizando GLUT[6] [7] e OpenGL) deve ser feita pelas rotinas fornecidas pelo professor.

---

[2] Outras propriedades podem ser incorporadas.

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 como executar o programa.

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 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] Sherrod, Allen. Data structures and algorithms for game developers. Cengage Learning, 2007.

[2] 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] Naylor, Bruce F. "**A tutorial on binary space partitioning trees.**" Computer Games Developer Conference Proceedings. 1998.
Disponível em: https://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/2001/pub/www/notes/bsp_tutorial.pdf.

[5] Samet, Hanan. "**The quadtree and related hierarchical data structures.**" ACM Computing Surveys (CSUR) 16.2 (1984): 187-260.
Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.299.334&rep=rep1&type=pdf

[6]