Detecção de colisão para simulação entre partículas representadas como superelipses

Alice Herrera de Figueiredo e Waldemar Celes
Instituto Tecgraf/PUC-Rio, Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brasil
{herrera,celes}@tecgraf.puc-rio.br

***Abstract*—This work explores the use of superellipses for representing various geometrical shapes of grains. A methodology is proposed for collision detection based on quadtrees and interval arithmetic. We also propose a projection method for handling collisions using relaxation for simulating particles of different shapes.**

***Keywords*-superellipses; quadtree; physics simulation**

***Resumo*—Este trabalho explora a utilização de superelipses para a representação de variadas formas geométricas de grãos. É proposta uma metodologia para detecção de colisão baseada em quadtree e aritmética intervalar. Também é proposto um método de projeção para o tratamento das colisões usando relaxação para simulação de partículas com formas variadas.**

***Palavras-chave*-superelipses; quadtree; simulação física**

I. Introdução

Este traballho foi motivado pelo artigo "An investigation and optimization of the 'OLDS' elevator using Discrete Element Modeling" [1] que usa o método DEM (Discrete Element Method) para simulação de grãos, modelando partículas como superquádricas [2].

Superquádricas são uma família de formas geométricas definidas por fórmulas que se assemelham às de elipsóides e de outras quádricas:

$$\left|\frac{x}{A}\right|^r + \left|\frac{y}{B}\right|^s + \left|\frac{z}{C}\right|^t \leq 1$$

A Fig. 1 mostra exemplos de superquádricas, em que $r = s = \varepsilon_1$ e $t = \varepsilon_2$.

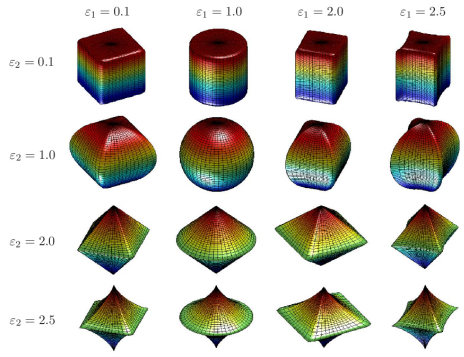

Fig. 1. Exemplos de superquádricas [3]

A fim de encontrar uma forma eficiente e correta do cálculo da colisão entre superquádricas, investigamos neste trabalho o problema em duas dimensões, adotando técnicas que são também aplicáveis a superfícies 3D. Nesse caso, os objetos que vamos simular tem a forma de uma superelipse sólida [4]. Na sua forma mais simples (centrada na origem e com os eixos paralelos aos eixos coordenados) uma superelipse é dada pelo conjunto de todos os pontos (x, y) tais que $h(x, y) \leq 1$, onde:t

$$h(x,y) = \left|\frac{x}{s_x}\right|^{e_x} + \left|\frac{y}{s_y}\right|^{e_y}$$

Nessa expressão, s_x é o semi-eixo horizontal, s_y é o semi-eixo vertical e os expoentes e_x e e_y controlam o quão arredondada é a superelipse. Por exemplo, quando $e_x = e_y = 2$, a equação $h(x, y) = 1$ define uma elipse, que é um círculo quando $s_x = s_y$.

Vamos considerar a forma mais geral que inclui uma translação da origem para o seu centro (x_0, y_0) e uma rotação de ângulo θ. Portanto, as superelipses que vamos considerar são dadas por $f(x, y) \leq 1$, onde

$$f(x,y) = h(R_{-\theta}(x - x_0, y - y_0)) \text{ e } R_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

é a matriz de rotação de ângulo α.

A proposta deste trabalho é utilizar as superelipses para representar diversas formas geométricas de grãos. A partir dessa representação, simulamos partículas com formas variadas. Para a simulação, é necessário detectar a colisão entre essas partículas. Para isso, utilizamos decomposições em quadtree e aritmética intervalar (Seção II) para garantir a detecção robusta da colisão. Para o tratamento de colisão, usamos um método de projeção (Seção III).

II. Quadtree

Para detectar a colisão entre duas superelipses, utilizamos quadtree com a aritmética intervalar [5], [6]. Esse método permite detectar a colisão entre quaisquer superfícies implícitas.

Quadtree [7] é um método para particionar um espaço bidimensional pela divisão recursiva e adaptativa desse espaço em quatro quadrantes. Neste trabalho, usamos quadtree para detectar a colisão entre superelipses.

Para utilizarmos esse método, primeiro classificamos cada quadrante em relação a cada objeto (dentro, fora e na fronteira). Para realizar essa classificação, calculamos (veja a

Seção II-A) um intervalo $[a, b]$ dos valores que a função $f(x, y)$, que determina o objeto, assume no quadrante:

1) Se $a > 0$, o intervalo é todo positivo e o quadrante está fora do objeto.
2) Se $b < 0$, o intervalo é todo negativo e o quadrante está dentro do objeto.
3) Caso contrário, o quadrante está na fronteira do objeto.

Em seguida, utilizamos essa classificação para classificar o quadrante em relação à colisão de dois objetos:

1) Se um quadrante estiver fora de um dos objetos, então a colisão não acontece dentro desse quadrante e ele pode ser descartado.
2) Se um quadrante estiver dentro de ambos os objetos, então esse quadrante faz parte da colisão e não precisa ser refinado.
3) Se um quadrante estiver na fronteira de algum objeto e ainda não foi descartado, refinamos o quadrante.

O processo de refinamento recursivo termina ao atingir a precisão escolhida pelo usuário.

A vantagem do método da quadtree é que ele detecta a colisão de quaisquer duas superelipses de forma robusta, concentrando o esforço na região de penetração. Note nas Fig. 2 e Fig. 3 que o refinamento se dá em direção aos trechos das fronteiras na área de colisão. Além disso o método da quadtree pode ser facilmente estendido para calcular colisão entre outras formas definidas implicitamente. Por essas razões o método da quadtree foi o método escolhido neste trabalho.

Exemplo 1:

$$|x|^2 + |y|^2 \leq 1 \text{ e } \left|\frac{x - x_0}{2}\right|^2 + \left|\frac{y - y_0}{2}\right|^2 \leq 1$$

O centro da segunda equação varia na Fig. 2.

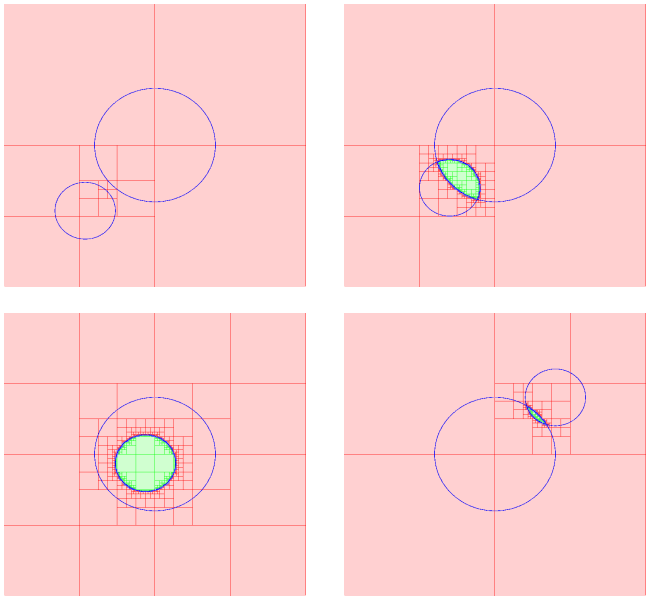

Fig. 2. Testando colisão entre círculos usando a quadtree

Exemplo 2:

$$\left|\frac{x}{2}\right|^3 + \left|\frac{y}{2}\right|^5 \leq 1 \text{ e } \left|\frac{x - x_0}{4}\right|^{0.5} + \left|\frac{y - y_0}{4}\right|^{0.5} \leq 1$$

O centro da segunda equação varia na Fig. 3.

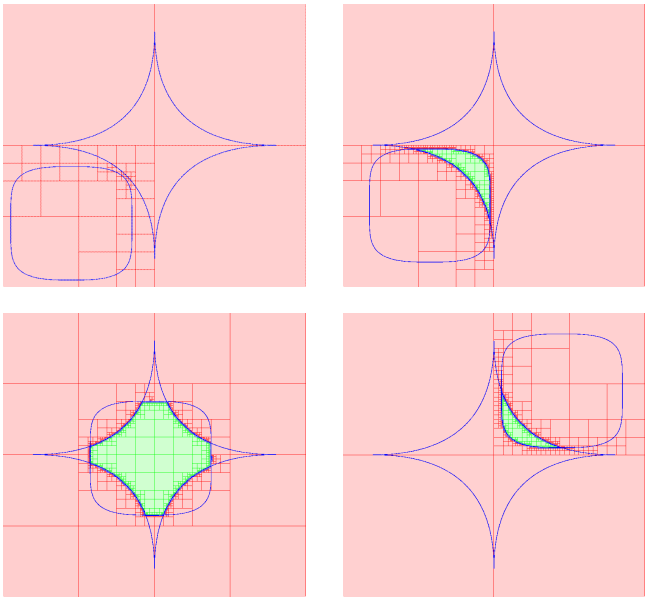

Fig. 3. Testando colisão entre outras superelipses usando a quadtree

A. *Avaliação Intervalar*

Para a classificação dos quadrantes, usamos a aritmética intervalar [8] para calcular um intervalo $[a, b]$ dos valores que a função $f(x, y)$ assume no quadrante $[x_1, x_2] \times [y_1, y_2]$.

Quando não há rotação, a equação que define a superelipse é suficientemente simples para que possamos calcular o intervalo $[a, b]$ exatamente.

De fato, queremos calcular o intervalo $[a, b]$ que contém todos os valores de

$$\left|\frac{x - x_0}{s_x}\right|^{e_x} + \left|\frac{y - y_0}{s_y}\right|^{e_y} - 1$$

para $(x, y) \in [x_1, x_2] \times [y_1, y_2]$.

Inicialmente, queremos calcular o intervalo $[a_x, b_x]$ que contém todos os valores de $\left|\frac{x-x_0}{s_x}\right|^{e_x}$ quando $x \in [x_1, x_2]$. Isso é feito em três passos.

1) Temos
$$\frac{x - x_0}{s_x} \in \left[\frac{x_1 - x_0}{s_x}, \frac{x_2 - x_0}{s_x}\right]$$
2) Tomando
$$t = \frac{x - x_0}{s_x}, \quad t_1 = \frac{x_1 - x_0}{s_x}, \quad t_2 = \frac{x_2 - x_0}{s_x}$$
temos:
$$|t| \in \begin{cases} [t_1, t_2], & \text{se } t_1 > 0 \\ [-t_2, -t_1], & \text{se } t_2 < 0 \\ [0, \max(-t_1, t_2)], & \text{se } t_1 \leq 0 \leq t_2 \end{cases}$$

3) Finalmente,

$$|t|^e \in [a_x, b_x] = \begin{cases} [t_1^e, t_2^e], & \text{se } t_1 > 0 \\ [(-t_2)^e, (-t_1)^e], & \text{se } t_2 < 0 \\ [0, \max(-t_1, t_2)^e], & \text{se } t_1 \leq 0 \leq t_2 \end{cases}$$

Do mesmo modo, obtemos o intervalo $[a_y, b_y]$ que contém todos os valores de $\left|\frac{y-y_0}{s_y}\right|^{e_y}$ quando $y \in [y_1, y_2]$.

Finalmente, o intervalo que procuramos é dado por

$$[a, b] = [a_x + a_y - 1, b_x + b_y - 1]$$

Quando há rotação, calculamos a caixa envolvente do quadrante $[x_1, x_2] \times [y_1, y_2]$ após a rotação de ângulo $-\theta$ e usamos essa caixa envolvente no cálculo anterior.

B. Rendering

O esquema de classificação de quadrantes em relação a um objeto é usado também para desenhar o objeto. Os quadrantes classificados como totalmente fora do objeto não são pintados. Os quadrantes classificados como totalmente dentro do objeto são pintados com a cor do objeto. Os quadrantes classificados como borda no fim da decomposição são pintados com uma mistura da cor do fundo e da cor do objeto. Essa mistura é uma interpolação linear dessas duas cores cujo parâmetro é calculado dividindo o quadrante em quatro partes e classificando essas partes como dentro, fora e na borda do objeto. Esse esquema gera bordas suaves (*anti-aliasing*).

III. Simulação física

Utilizando o método de detecção de colisão descrito acima, fizemos uma simulação física de partículas caindo em uma caixa. A cena é composta por uma caixa aberta e vários objetos em forma de superelipses dentro dela. Assim que a simulação começa, os objetos vão surgindo no topo e ao longo do tempo vão caindo até colidirem com o fundo da caixa, com as paredes ou com outros objetos.

Para simular a física utilizamos a integração de Verlet [9]:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + (1 - \delta)(\mathbf{x}_i - \mathbf{x}_{i-1}) + \frac{h^2}{m}\mathbf{f}$$

onde

- $\mathbf{x}$ é a posição do objeto;
- δ é o coeficiente de viscosidade;
- h é o passo de integração do método;
- m é a massa do objeto;
- $\mathbf{f}$ é a resultante das forças que atuam no objeto.

Nessa simulação, a posição do objeto é um ponto no plano, tal que $\mathbf{x} = (x, y)$. Assim, na integração de Verlet temos, em princípio, que recalcular tanto a coordenada x quanto a coordenada y.

O coeficiente de viscosidade δ serve para amortecer a simulação, evitando oscilações. Ele assume valores pequenos. Utilizamos $\delta = 0.1$.

Quanto menor o passo de integração h, mais preciso é o método; entretanto, a simulação fica mais lenta. Assim, é necessário encontrar um valor para h que forneça um equilíbrio entre a precisão e a eficiência. Utilizamos $h = 0.01$.

Nessa simulação, não estamos considerando outras forças, apenas o peso. Pela Segunda Lei de Newton, temos que $\mathbf{f} = m \cdot (0, -g)$, onde $g = 9.8$ é a aceleração da gravidade. Note também que $\mathbf{f}$ só tem componente na vertical e portanto basta atualizar y na integração de Verlet.

Percebemos que m divide $\mathbf{f}$ na integração de Verlet. Assim, como só estamos considerando a força de gravidade, o valor da massa do objeto é irrelevante, pois será cancelado com a massa da equação da força. Utilizamos $m = 1$.

A cada passo da simulação, para cada objeto, primeiro atualizamos a posição do objeto pela integração de Verlet e em seguida verificamos se o objeto colidiu com outro ou com as paredes da caixa. Se ocorrer colisão, a posição do objeto é atualizada. Para tratar a colisão utilizamos o método de projeção com relaxamento, como proposto por Jakobsen [9]. Isso é feito até que não ocorram mais colisões ou até que o número máximo de iterações pré-definido seja alcançado. O algoritmo abaixo descreve esse processo:

```
while simulating do
  for all objects do
    object.position ← verlet()
    while inCollision or maxIter do
      handleCollision()
    end while
  end for
end while
```

A função $handleCollision()$ verifica se ocorreu a colisão entre dois objetos ou entre um objeto e uma parede e atualiza a posição do objeto.

A. Cálculo da colisão entre dois objetos

O cálculo da colisão entre dois objetos é feito avaliando a colisão do objeto i com os próximos objetos iniciando de $j = i + 1$.

Como realizar o cálculo da colisão é muito custoso, calculamos a caixa envolvente de cada objeto e verificamos se elas se interceptam, como ilustrado na Fig. 4 e na Fig. 5.

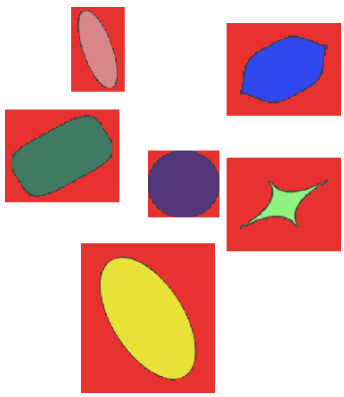

Fig. 4. Exemplos de superelipses e suas respectivas caixas envolventes

Como o objeto é representado por uma superelipse, o limite da sua caixa é definido inicialmente pelos seus eixos vertical e horizontal. Esse limite é atualizado após o cálculo de rotação em relação ao centro da superelipse.

Assim, se as caixas envolventes não tiverem interseção, então os objetos correspondentes não colidem. Se as caixas envolventes tiverem interseção, o cálculo da colisão entre os objetos (i.e., a construção da quadtree) é realizado apenas dentro da interseção das caixas envolventes. Isso permite um desempenho melhor do que se a colisão fosse calculada em uma caixa maior que essa. A Fig. 5 ilustra a caixa em que o cálculo da colisão é feito.

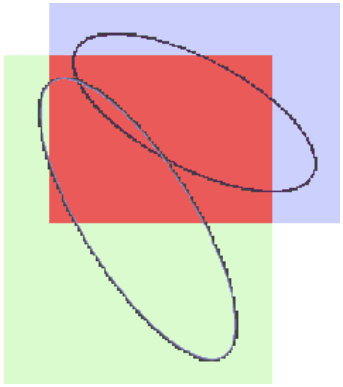

Fig. 5. Exemplo de colisão entre dois objetos

Embora esse método de colisão considere um número quadrático de pares de objetos, ele só executa um número *linear* de cálculos custosos. Como trabalho futuro, pode-se empregar estruturas de aceleração para detectar eficientemente potenciais pares de objetos em colisão (*broad-phase*).

Utilizamos o método descrito na Seção II para detectar a colisão e, ao chegarmos na precisão escolhida pelo usuário, determinamos o ponto de cada objeto que é mais interior ao outro objeto. Para isso, encontramos o ponto médio que, substituído na equação do outro objeto, resulta no menor valor. (O interior de um objeto é formado pelos pontos nos quais o valor da função implícita associada ao objeto é negativo.)

Se conseguirmos encontrar esses dois pontos mais interiores em cada objeto, podemos estimar a penetração. Seja A_1 o ponto do objeto 1 que é mais interior ao objeto 2. Calculamos $\nabla f_2(A_1)$, onde f_2 é a função que define o objeto 2, e procuramos o ponto B_2 onde a semi-reta de A_1 na direção de $\nabla f_2(A_1)$ cruza a fronteira do objeto 2. O vetor $\overrightarrow{A_1B_2}$ estima a penetração. O cálculo inverso, utilizando o ponto A_2 e a função f_1 para encontrar o vetor $\overrightarrow{A_2B_1}$, é feito da mesma forma.

Para estimarmos o ponto B_2, utilizamos o método da bisseção. Em poucos passos é possível encontrar uma boa aproximação para o ponto.

Na Fig. 6 podemos visualizar a estimativa dessa penetração e a direção de penetração máxima. Note que quando um objeto passa do centro do outro, a direção de penetração está

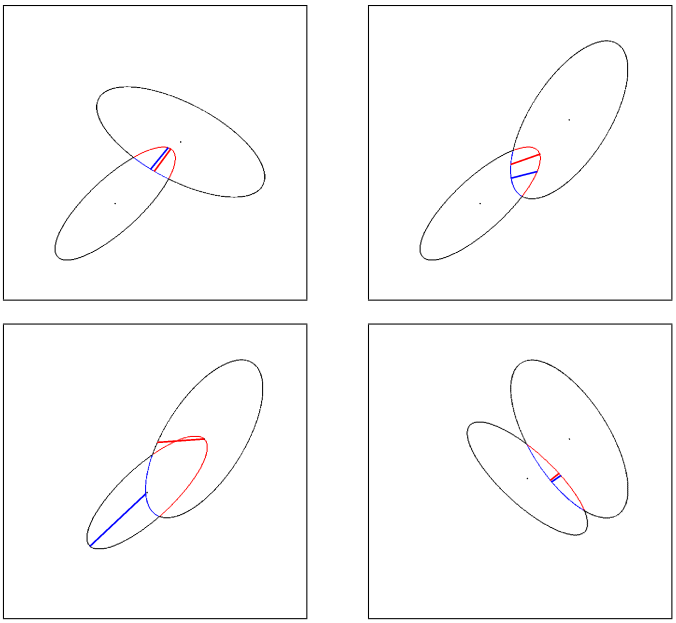

Fig. 6. Exemplo da estimativa de penetração

errada. Como na nossa simulação isso não acontecerá pois a penetração será muito pequena, esse cálculo funciona e é o primeiro passo para calcular a direção de deslocamento.

Utilizaremos, como exemplo, o vetor $\overrightarrow{A_1B_2}$ para determinar a nova posição do objeto 1 após a colisão, como ilustra a Fig. 7. Para encontrar a nova posição do objeto 2 após a colisão, utilizaríamos o vetor $\overrightarrow{A_2B_1}$. O objetivo é projetar os objetos de forma que a interpenetração seja nula.

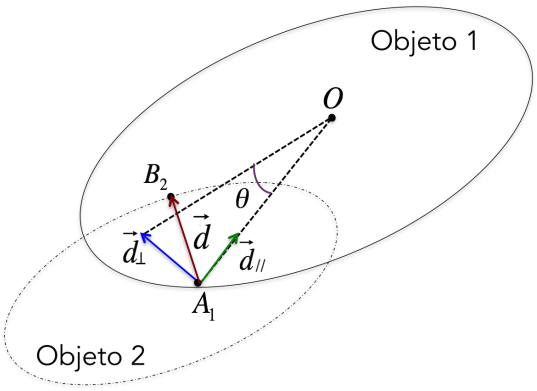

Fig. 7. Cálculo da direção de deslocamento

Se a posição atual do centro do objeto é (x_0, y_0), então a nova posição do centro é dada por $(x_0, y_0) + 0.5 \cdot d_{\parallel}$, onde:

$$d_{\parallel} = \frac{v \cdot u}{u \cdot u} u, \qquad u = \overrightarrow{A_1O}, \qquad v = \overrightarrow{A_1B_2}$$

Se θ é o ângulo de rotação atual do objeto, então o novo ângulo de rotação é dado por $\theta \pm 0.5 \cdot \tan^{-1}\left(\frac{\|d_{\perp}\|}{\|u\|}\right)$, onde:

$$d_{\perp} = d - d_{\parallel}$$

e o sinal $\pm$ é o sinal do produto vetorial $d_{\perp} \times d_{\parallel}$ no plano.

B. Cálculo da colisão entre um objeto e as paredes da caixa

O cálculo da colisão entre um objeto e as paredes da caixa é muito parecido com o cálculo entre dois objetos. Nesse caso, cada borda tem sua própria equação implícita:

- Parede esquerda:

$$g(x, y) = x - x_E$$

  onde x_E é a posição da parede esquerda na Fig. 8;
- Parede direita:

$$g(x, y) = x_R - x$$

  onde x_R é a posição da parede direita na Fig. 8;
- Chão:

$$g(x, y) = y - y_C$$

  onde y_C é a posição do chão na Fig. 8.

Nos três casos, o interior da parede é dado implicitamente por $g(x, y) < 0$. Essa formulação permite tratar paredes e objetos uniformemente. Isso significa que o cálculo da colisão é o mesmo como descrito na Seção III-A, exceto que nesse caso não é necessário utilizar o método da bisseção, pois as paredes possuem equações simples.

A caixa envolvente do cálculo de colisão entre um objeto e uma parede também é mais simples. Ela é a interseção da caixa do objeto com a parede. Portanto, para a parede esquerda, a caixa de colisão tem $x_{max} = x_E$; para a parede direita, a caixa de colisão tem $x_{min} = x_R$; para o chão, a caixa de colisão tem $y_{max} = y_C$; como ilustra a Fig. 8.

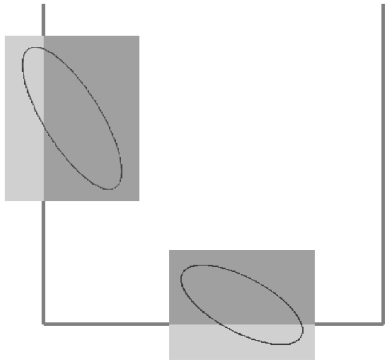

Fig. 8. Exemplo de colisão entre objetos e as paredes da caixa

IV. Resultados

Para implementarmos a simulação, utilizamos a linguagem C++ e a biblioteca gráfica OpenGL.

As próximas figuras contem exemplos de diferentes simulações conforme ocorrem mudanças nos parâmetros da equação da superelipse:

$$\left|\frac{x - x_0}{s_y}\right|^{e_x} + \left|\frac{y - y_0}{s_x}\right|^{e_y} \leq 1$$

Além desses parâmetros, cada superelipse tem uma rotação inicial aleatória de ângulo θ em torno do centro (x_0, y_0). O refinamento da quadtree para quando o nível 7 é atingido.

Em todas as simulações, o valor inicial da coordenada x_0 é um valor aleatório entre os limites das duas paredes da caixa e o valor inicial da coordenada y_0 é o topo da janela da simulação.

A Fig. 10 mostra a simulação de 35 círculos ($e_x = e_y = 2.0$) com raios escolhidos aleatoriamente no intervalo $[0.3, 0.7]$.

A Fig. 11 mostra a simulação de 35 elipses ($e_x = e_y = 2.0$), orientadas aleatoriamente e com eixos escolhidos aleatoriamente nos intervalos $[0.1, 0.8]$ e $[0.1, 0.9]$.

A Fig. 12 mostra a simulação de 40 superelipses ($e_x = e_y = 4.0$), orientadas aleatoriamente e com eixos escolhidos aleatoriamente nos intervalos $[0.3, 0.4]$ e $[0.3, 0.7]$.

A Fig. 13 mostra a simulação de 40 superelipses com e_x e e_y escolhidos aleatoriamente no intervalo $[0.5, 4.0]$, orientadas aleatoriamente e com eixos escolhidos aleatoriamente nos intervalos $[0.3, 0.4]$ e $[0.3, 0.7]$.

A. Discussão

Como as regiões de potencial colisão tendem a ser retângulos finos, a decomposição usada no método de quadtree é feita somente nas dimensões maiores do que a tolerância ε escolhida pelo usuário, como ilustrado na Fig. 9.

Embora a simulação reduza os testes de colisões a um número linear no número de objetos, esses testes ainda são aonde a simulação gasta mais tempo. Assim, quanto maior o número de objetos na cena, maior o tempo gasto a cada iteração.

Supreendentemente, o uso da função intrínseca *pow(x,e)*, que é crucial na definição da superelipse, é responsável pela maior parte do tempo gasto na simulação (conforme reportado pela ferramenta de *profiling Instruments* no Mac OS X). Obtivemos visível melhora no desempenho substituindo a função intrínseca pela nossa própria implementação que trata explicitamente os casos $e = 1$ e $e = 2$.

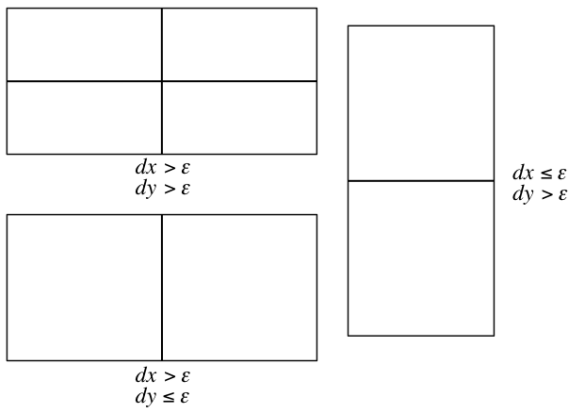

Fig. 9. Subdivisão retangular

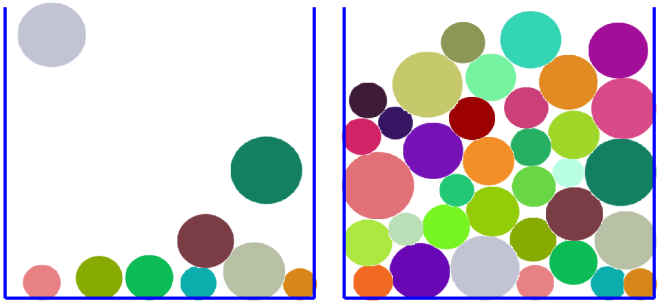

Fig. 10. Simulação de colisão entre círculos

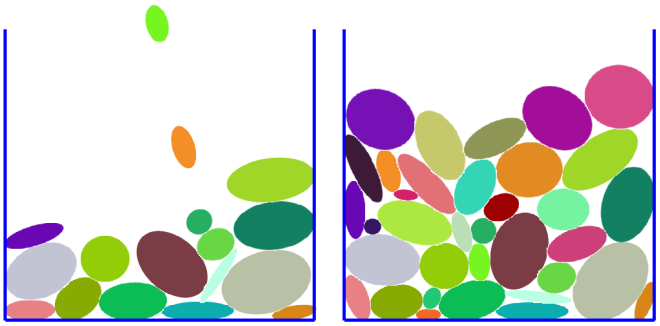

Fig. 11. Simulação de colisão entre elipses

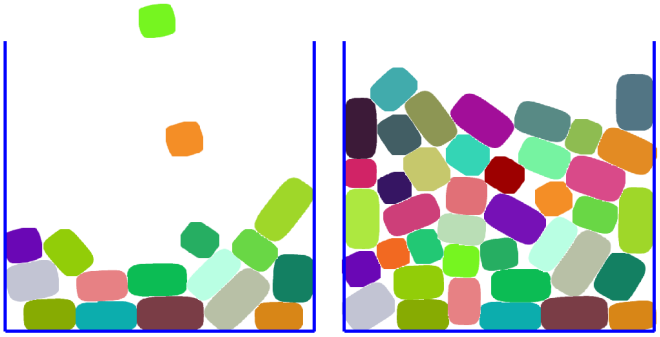

Fig. 12. Subdivisão retangular

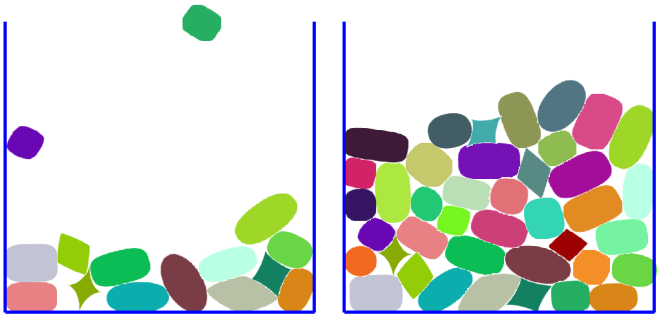

Fig. 13. Subdivisão retangular

Por outro lado, testamos uma versão do programa escrita explicitamente para o caso particular de colisão de círculos usando o teste geométrico da distância entre os centros ser menor que a soma dos raios em vez de aritmética intervalar. Esse programa obteve um desempenho mais que 10 vezes melhor. Isso nos motiva a trabalhar para melhorar o desempenho da versão intervalar.

V. Considerações finais

Este trabalho atingiu seu objetivo principal que era detectar e calcular de forma eficaz a colisão de superelipses. A escolha da quadtree e o método para eliminar a interpenetração resultaram em simulações coerentes.

A continuação natural deste trabalho é estendê-lo para 3D, adaptando todos os cálculos para superquádricas. Para isso é necessário desenvolver um modelo de colisão que trate a atualização dos dois ângulos de orientação do objeto. Acreditamos que embora seja possivelmente trabalhoso, a detecção e a simulação de superquádricas não sejam muito mais complicadas do que a versão em 2D tratada aqui, visto que é simples mudar de quadtree para octree e mudar a avaliação intervalar para três variáveis.

Referências

[1] W. McBride and P. Cleary, "An investigation and optimization of the 'OLDS' elevator using Discrete Element Modeling," *Powder Technology*, vol. 193, no. 3, pp. 216–234, 2009.

[2] K. Duncan, S. Sarkar, R. Alqasemi, and R. Dubey, "Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 4238–4243.

[3] GSoC, disponível em: http://www.pointclouds.org/assets/uploads/gsoc14-superquadrics.jpg, [Último acesso em 1 de julho de 2015].

[4] M. Gardner, "Piet Hein's superellipse," *Mathematical Carnival. A New Round-Up of Tantalizers and Puzzles from Scientific American*, pp. 240–254, 1977.

[5] H. Lopes, J. B. Oliveira, and L. H. de Figueiredo, "Robust adaptive polygonal approximation of implicit curves," *Computers & Graphics*, vol. 26, no. 6, pp. 841–852, 2002.

[6] K. G. Suffern, "Quadtree algorithms for contouring functions of two variables," *Comput. J.*, vol. 33, no. 5, pp. 402–407, Oct. 1990.

[7] H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surv.*, vol. 16, no. 2, pp. 187–260, Jun. 1984.

[8] R. Moore, R. Kearfott, and M. Cloud, *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, 2009.

[9] T. Jakobsen, "Advanced character physics," in *In Proceedings of the Game Developers Conference 2001*, 2001, p. 19.