SIMULACAO COM ESTRUTURA MISTA IMPLICITA PARAMETRICA

Aluno: Rafael Martinez
Orientador: Thomas Lewiner

Introdução

A simulação de fluídos vem sendo cada vez mais utilizada na indústria e na pesquisa. A evolução dos meios computacionais permitiu alcançar simulação de fenômenos de grande escala. Em particular em casos industriais como a aplicação de mecânica dos fluídos à indústria do petróleo, é possível simular escoamentos de tamanho de alguns minutos em menos de um mês de cálculo computacional. Porém, esse tempo pode ser superado usando técnicas adequadas para representar a geometria do problema, em particular na representação da superfície livre do fluído.

Objetivos

O objetivo deste trabalho é representar dinamicamente a fronteira de um fluído durante a simulação, misturando estruturas implícitas às estruturas explícitas. Para isso, implementamos em Java uma estrutura de partículas e uma estrutura para a superfície livre que constituam o fluído onde podem se aplicar as leis da física específicas à superfície.

A Hierarquia de Estruturas de Dados - Construção de um universo em duas dimensões

As primeiras estruturas de dados do projeto foram responsáveis por criar as abstrações geométricas necessárias para manipular formas num universo em duas dimensões.

A estrutura de dados mais básica no projeto corresponde à abstração de um ponto no plano. Ela simplesmente armazena como informação as suas duas componentes no plano e um Vetor Normal que é utilizado nos métodos de manipulação da curva. Cada instancia desta classe possui métodos de troca e alteração das informações nela encapsuladas.

A segunda estrutura de dados mais básica corresponde à abstração de um vetor no plano. Ela possui como informação as suas duas coordenadas. A classe e suas instancias possuem métodos de somar vetores, subtrair vetores, calcular o produto interno, o produto vetorial, o ângulo com o eixo x, a norma e métodos para normalizar ou rodar um vetor.

A próxima classe em nível de abstração é a classe das figuras geométricas. Ela é uma classe abstrata que é uma superclasse de duas classes de figuras: o Circulo e o Retângulo. Como superclasse abstrata, ela dispõe como herança para suas subclasses somente métodos gerais que podem ser usados por qualquer figura geométrica. O primeiro método a ser herdado pelas suas subclasses verifica se a figura apresenta intersecção com algum retângulo especifico. O segundo método verifica se existe alguma partícula no interior da figura.

A classe dos círculos, como subclasse das figuras geométricas, implementa os métodos de sua superclasse e seus métodos próprios de troca de informação. A classe dos retângulos além de implementar os métodos de sua superclasse, disponibiliza métodos para obtenção de seus quadrantes, que são utilizados depois para a construção da quadtree.

A Hierarquia de Estruturas de Dados – Construindo o fluido

Uma vez criada a abstração geométrica necessária ao projeto, pode-se seguir as abstrações mais especificas do fluido. As classes do fluido têm um nível mais alto de

abstração uma vez que utilizam as classes geométricas ora como atributo, ora em seus métodos, quando não nos dois.

As partículas formam a classe mais primaria especificamente do fluido. Em nossa abstração uma partícula possui como informação a sua massa, o raio, a sua posição e as suas aceleração e velocidade atuais. O principio básico da simulação é, a cada passagem de tempo, calcular os três últimos destes atributos de cada partícula do fluido e alterar depois a curva em função do conjunto de partículas.

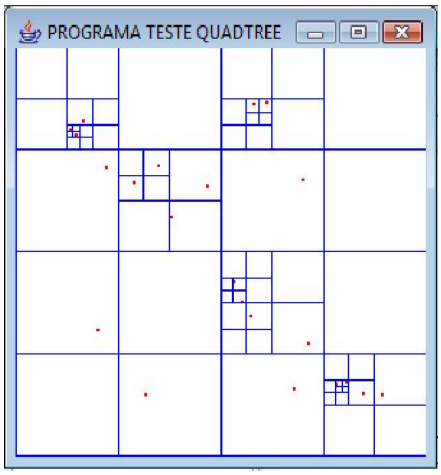
Subindo na hierarquia, encontramos agora uma divisão. De um lado, a estrutura de dados correspondente à parte implícita do fluido, a Quadtree que funciona em conjunto com um vetor de partículas como veremos mais tarde. E do outro lado, a Curva que corresponde à parte explicita do fluido. Estas duas estruturas de dados são unificadas mais tarde pela ultima estrutura da hierarquia: O Fluido.

Estrutura implícita

Na simulação lidamos com muitas partículas simultaneamente, portanto se avaliássemos a força resultante em uma partícula usando as interações de todas as outras partículas, teríamos um algoritmo de ordem $O(n^2)$. Isto inviabilizaria a simulação, visto o numero de partículas a serem avaliadas a cada passo. Para resolver este problema usamos a estrutura de busca espacial quadtree. Ela funciona dividindo o espaço recursivamente em quadrantes até que em cada quadrante exista no máximo uma partícula. Desta forma, para cada partícula podemos descobrir quais partículas estão mais próximas. Assim, no momento de calcular a força resultante, podemos avaliar as interações com as partículas mais próximas e dispensar do calculo as mais distantes. Uma vez que o valor destas interações é desprezível.

A classe das quadtrees é particularmente interessante. Ela poderia ser implementada de diversas maneiras, mas a maneira escolhida viabiliza um menor consumo de memória, uma vez que no lugar de cada nó armazenar uma instancia de partícula ele armazena apenas um numero inteiro que representa o índice do vetor onde esta a partícula. Este vetor é um atributo da classe dos fluidos que veremos mais adiante.

A quadtree é em essência uma estrutura recursiva, embora seus métodos sejam adaptados para não o serem, pois se eles fossem também recursivos, gastariam muito a pilha de execução. Cada instancia da quadtree é chamada de nó por se tratar de uma arvore quaternária de busca. Cada nó, por sua vez, possui como informação um nó pai, um nó irmão, um nó para o primeiro filho, um retângulo (que representa a região do plano que pertence ao nó) com o numero de partículas no seu interior, e, por ultimo, o índice da partícula no vetor (caso aja somente uma partícula neste nó).



A estrutura é construída utilizando o método de subdivisão. Ele funciona da seguinte maneira: O primeiro nó (raiz) da quadtree engloba todas as partículas do fluido e possui como informação o primeiro dos seus quatro filhos, que representa o quadrante noroeste do retângulo. Este filho vai possuir como nó irmão o quadrante nordeste do retângulo do nó pai. Por sua vez, o quadrante nordeste vai possuir o quadrante sudeste o qual vai possuir o quadrante sudoeste. E assim o retângulo do nó pai fica dividido em quatro quadrantes, cada um pertencendo a um nó filho. Embora o nó pai só tenha como informação o primeiro filho, todos os filhos têm o nó pai como informação. Esta operação se sucede recursivamente ate que os últimos nos (folhas) contenham uma ou zero partícula.

Os métodos da quadtree como dito anteriormente não são recursivos poupando tempo e memória. Eles se dividem em dois tipos: os métodos de mudança de estado e o método de busca de partículas. Os métodos de mudança de estado são a subdivisão, usado para construir ou reconstruir a quadtree, e a atualização, usado para deslocar uma partícula e atualizar a quadtree sem precisar reconstruí-la. O método de busca de partículas recebe uma figura geométrica e procura todas as partículas no interior desta figura. Este método é utilizado para minimizar o tempo do calculo das forças resultantes nas partículas como veremos mais adiante.

Estrutura explícita

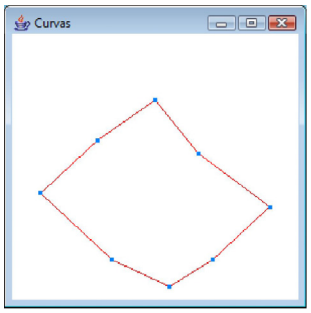
A parte explícita da estrutura do Fluído corresponde à Curva. Ela representa a fronteira do fluído. Ela dispõe de um método de criação que, através de um algoritmo desenvolvido no projeto, calcula os pontos da curva de maneira a englobar todas as partículas do fluido em seu interior.

Antes de passar para a classe das curvas, cabe explicar uma particularidade desta classe que é o uso de uma Estrutura Genérica em sua implementação. Esta estrutura genérica corresponde por sua vez à classe das listas genéricas.

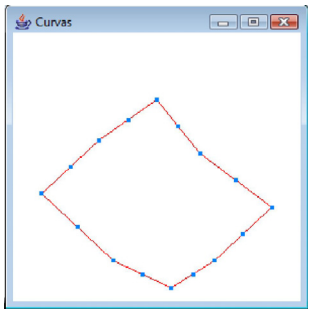
As listas genéricas são listas duplamente encadeadas que utilizam um recurso em JAVA chamado GENERICS. Este recurso disponibiliza a criação das chamadas *classes genéricas*. Estas classes funcionam da seguinte maneira: ao implementá-las podemos não especificar o tipo de um atributo (ou de um parâmetro ou retorno de um método), e depois, no momento da criação de instancias desta classe, devemos especificar o tipo genérico. Este recurso foi de grande utilidade uma vez que simplificou o projeto transformando classes semelhantes em uma só classe genérica.

Voltando a classe das curvas, ela é responsável pela parte explicita do fluido. Cada instancia desta classe possui uma lista de pontos como atributo e métodos de manipulação da curva. Os métodos são: refinamento colinear ou ponderado (métodos que aumentam a precisão da curva fornecendo mais pontos à mesma), suavização, cálculo das normais e suavização das mesmas.

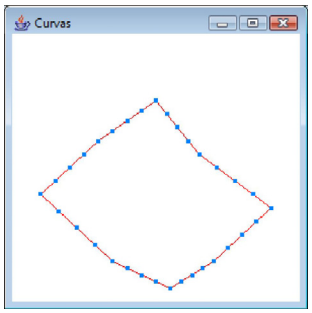
O refinamento colinear segue um algoritmo que insere um novo ponto na curva entre cada dois pontos consecutivos, de forma que o novo ponto seja o ponto médio dos antigos. Podemos ver aqui a evolução de uma curva simples sobre este método:



Curva exemplo: apenas 8 pontos.



2º Passo: curva com 16 pontos.



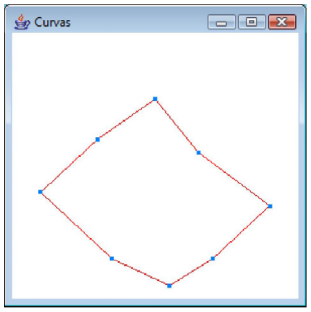
3º Passo: curva com 32 pontos.

Como se pode observar, o objetivo deste método na simulação é aumentar a precisão da curva sem alterar a sua forma.

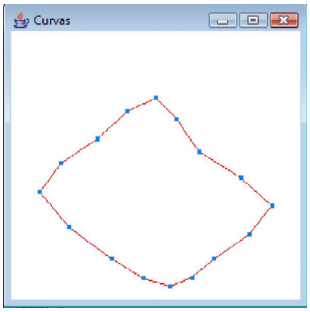
O refinamento ponderado por sua vez segue um algoritmo que além de colocar um novo ponto na curva entre cada dois pontos consecutivos, o posiciona a partir da posição dos 4 vizinhos de maneira a suavizar a curva, seguindo um esquema cúbico diferente dos splines usuais. Para compreender o funcionamento deste algoritmo imaginemos uma seqüência de 4 pontos vizinhos da curva. Chamemos de p1, p2, p3 e p4. Para posicionarmos um novo ponto entre p2 e p3, por exemplo, para cada componente do novo ponto faríamos uma média ponderada da forma:

$$\text{NovoPonto} = -\frac{1}{16}\cdot p1 + \frac{9}{16}\cdot p2 + \frac{9}{16}\cdot p3 - \frac{1}{16}\cdot p4 \;;$$

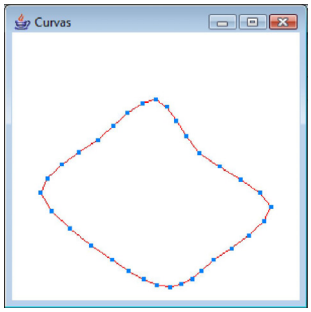
Assim, a curva além de mais precisa se torna mais suave. Como podemos ver na evolução da mesma curva do exemplo anterior agora sobre este método:



Curva exemplo: apenas 8 pontos.

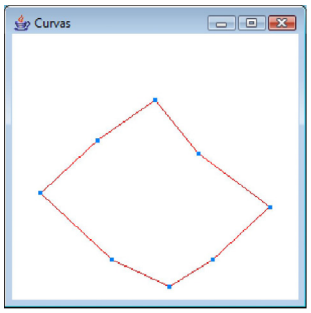


2º Passo:
curva mais suave com 16 pontos.

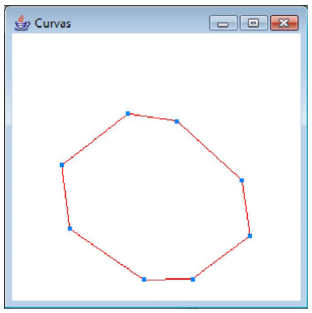


3º Passo:
curva mais suave com 32 pontos.

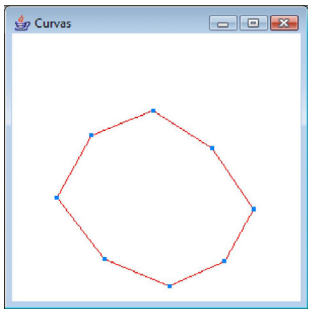
Existe também o método de Suavização, para uniformizar a curva, diminuindo as saliências da mesma, geralmente devidas aos erros numéricos. Este método não altera a precisão (numero de pontos) da curva, mas desloca cada ponto em função dos seus 4 vizinhos como podemos ver:



Curva exemplo: apenas 8 pontos.

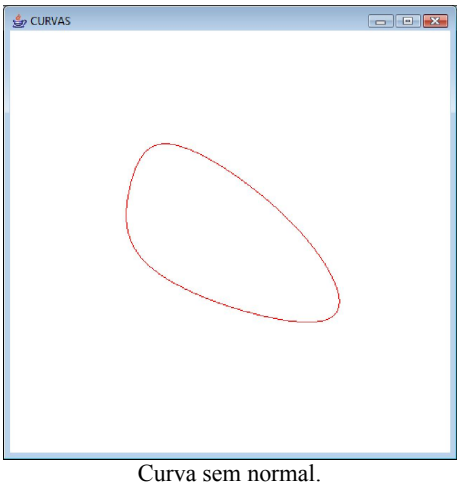


2º Passo: curva mais suave.

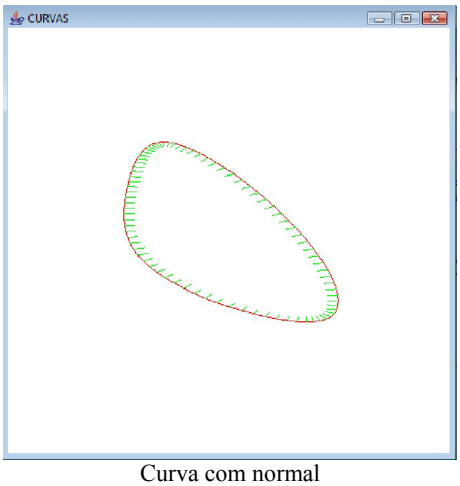


3º Passo: curva mais suave

Além destes, a Curva tem métodos para calcular a normal a seus pontos. O calculo da normal é feito da seguinte forma: imaginemos um ponto p1 da curva, do qual queremos calcular a normal. Sejam p0 e p2 os seus pontos anterior e próximo respectivamente. Como os pontos da curva não são deriváveis, o calculo da normal de p1 é feito rodando o vetor que vai de p0 a p2 e depois o normalizando.

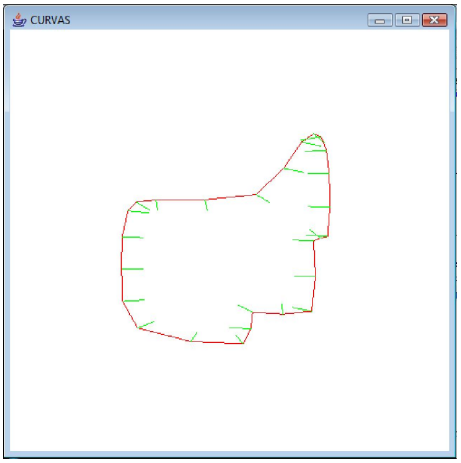


Curva sem normal.

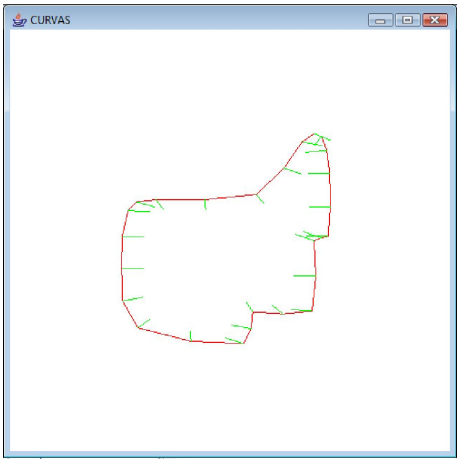


Curva com normal

Existe também um método para suavizar cada normal. Seu algoritmo é semelhante ao utilizado para suavização da curva. Para cada ponto, ele considera a normal dos 4 vizinhos e calcula a media ponderada.



Curva com normais em evidencia.



Curva após o suavização das normais

Um conjunto de testes permite aprovar a coerência da estrutura após cada operação.

O Fluido

Finalmente, chegando ao topo da hierarquia temos a classe dos fluidos. A classe dos fluidos é responsável pela interface entre as estruturas de dados implícitas e explicitas. Desta forma, ela possui como atributos uma curva, uma quadtree, um vetor de partículas e o numero de partículas do fluido.

A classe dos fluidos é responsável também pelas constantes utilizadas na simulação. São elas: o raio do circulo e o valor da constante sigma utilizado pelo método de projeção desta classe, o raio do circulo utilizado no método de busca da quadtree, o intervalo de tempo e a constante G utilizados no método de atualização da própria classe.

Esta classe possui um construtor de instancias randômico bastando informar o número de partículas desejado. Cada instancia possui, além de métodos de troca de informação, dois métodos principais: Projeção da curva e Atualização do fluido.

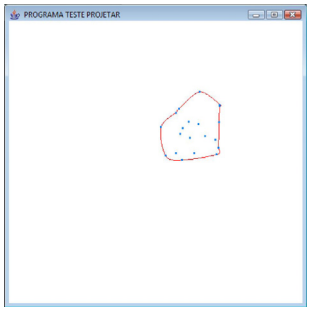
O objetivo do método de Projeção é aproximar a estrutura explicita da implícita, ou seja, projetar a curva em direção às partículas. Assim, molda-se o bordo do fluido em direção a estrutura implícita, esticando ou comprimindo o mesmo.

Cada ponto da curva é deslocado em função das partículas vizinhas. Para tal, estipula-se um valor de raio padrão e, para cada ponto da curva, constrói-se um circulo. Imaginemos agora um ponto p1 e seu respectivo circulo c1. Utiliza-se a arvore de busca quadtree para escolher as partículas mais próximas de p1, ou seja, aquelas que estão no interior de c1.

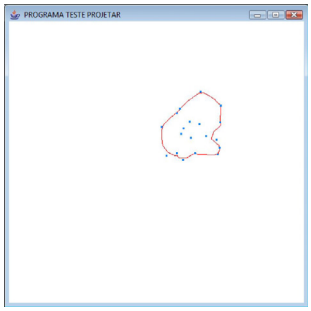
Em seguida, calcula-se o deslocamento de p1 em função das partículas. Chamemos de q1 uma das partículas contidas em c1. Calcula-se o vetor que vai de p1 a q1. Depois, calculam-se as projeções ortogonal e paralela deste vetor com a normal de p1. E o deslocamento final de p1 é dado pela equação:

$$\text{DeslocamentoFinal} = \frac{\sum_{\text{Partículas vizinhas}} \exp\left(-\frac{|ProjParalela|^2}{\sigma^2}\right) \cdot |ProjOrtogonal|}{\sum_{\text{Partículas vizinhas}} \exp\left(-\frac{|ProjParalela|^2}{\sigma^2}\right)}$$

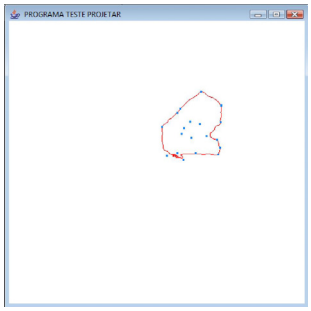
Podemos observar aqui a evolução de uma curva sendo projetada nas partículas:



Fluido exemplo.



1ª projeção da curva nas partículas.



2ª projeção da curva nas partículas.

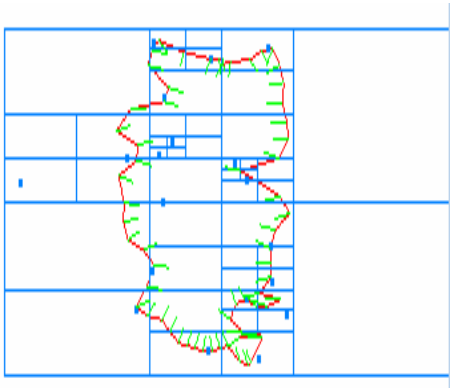
O método de atualização é responsável por deslocar as partículas a cada instante de tempo. Sua função é a partir das leis da física calcular a nova posição de cada partícula após um intervalo de tempo. Para cada partícula, calcula-se a aceleração resultante das leis da física aplicadas a ela e a partir desta aceleração, da velocidade inicial da partícula e do valor do intervalo de tempo, calcula-se a velocidade final. Tendo a velocidade final, a posição inicial (que já estava armazenada como informação na partícula) e o intervalo de tempo, calcula-se a posição final da partícula. Uma vez recalculadas as posições das partículas, atualiza-se a quadtree.

Para avaliar a interação entre cada partícula utilizamos o método de busca da quadtree. Assim o algoritmo que seria de ordem $O(n^2)$ passa a ser ordem $O(\log(n))$ na média. Do contrario seria inviável realizar a simulação com um número muito alto de partículas.

Visualização com OPENGL

Após todas as classes de abstração temos ainda a classe Draw. Esta classe corresponde à interface gráfica do projeto. Apesar de não possuir atributos, nela encontramos os métodos de desenho na tela, como o grid da quadtree, o desenho da curva, das partículas e das normais. Esta classe utiliza o pacote gráfico JOGL.

Resultados



As duas estruturas são coordenadas pela classe Fluído que, por sua vez, une as duas representações através de métodos como Projetar, que projeta cada ponto da curva nas partículas. Ele possui o método Atualizar responsável por calcular as variações do fluído no tempo.

As estruturas estão prontas e implementadas. Cada operação foi validada por uma bateria de testes próprios. Porém as equações usadas nos testes simulam um fluído ainda muito simples (gás em expansão).

Conclusão

O que foi desenvolvido neste projeto pode ser visto como o primeiro passo para o desenvolvimento de uma simulação de fluídos em 3D.

O projeto permitiu uma grande compreensão sobre as diferentes partes em que se divide a simulação. Aprendi muito sobre as diferentes estruturas de dados utilizadas no projeto (estruturas que representam as Partículas, a Curva, o Fluído e estruturas de partição do espaço). Além disto, tive um grande contato com programação orientada a objetos (pois a linguagem de programação utilizada é JAVA) e com o pacote gráfico OPENGL (para JAVA: JOGL).

Agradecimentos

Agradecemos ao CNPQ - PIBIC pela oportunidade de realizar o projeto. Infelizmente eu não terei a possibilidade de dar continuidade imediata ao projeto uma vez que farei o programa de Duplo Diploma Internacional.

Referências

1 - BARGTEIL A.W., GOKTEKIN T.G., O'BRIEN J.F., STRAIN J.A. A Semi-Lagrangian Contouring Method for Fluid Simulation. ACM Transactions on Graphics 25(1), 2006.

2 – CHUNG T. J. Computational Fluid Dynamics. 2002.