5 Sistema Proposto

5.1 Identificação da Interface

Dada a simulação de SPH de dois fluidos incompressíveis, o primeiro desafio é encontrar a interface entre eles. A interface é representada por partículas de um fluido que fazem contato e influenciam diretamente partículas de um segundo fluido (Figura 17).

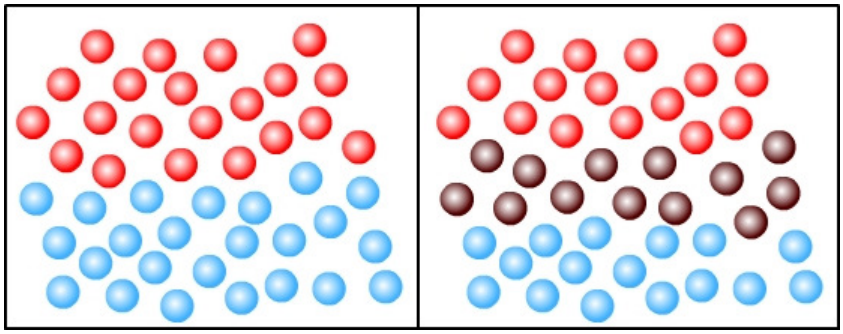

Figura 17 - Identificação de partículas que fazem parte da interface entre os fluidos

Para a identificação destas partículas utilizamos o cálculo do campo de cor, obtido pela equação 3.19. Partículas que fazem parte do fluido de maior densidade recebem valores positivos para o atributo *Cs* e partículas representando fluidos menos densos recebem valores negativos.

No cálculo da equação 3.19, notamos que partículas que pertencem ao interior do fluido mais denso terão apenas a influência de partículas com o valor de *Cs* positivo e no interior do fluido menos denso terão influência de partículas com atributo *Cs* negativo. Já partículas na interface são influenciadas pelos dois polos, fazendo com que seu campo de cor seja um valor próximo de zero (Figura 18).

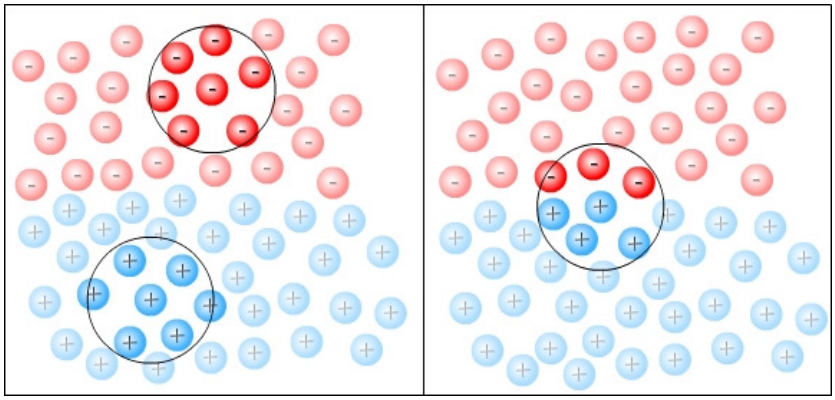

Figura 18 - Influência de partículas no cálculo da equação 3.19. Na esquerda: partículas dentro do fluido influenciam partículas vizinhas de mesmo polo. Na direita: partículas da interface influenciam partículas vizinhas de polos diferenciados

Logo, podemos definir as partículas da interface como sendo:

$$\begin{cases} -\epsilon \leq Cs\,(\vec{x}_i) \leq \epsilon, & partícula\ \vec{x}_i\ pertence\ \text{à}\ interface \\ caso\ contrário, & partícula\ \vec{x}_i\ não\ pertence\ \text{à}\ interface \end{cases}$$

Sendo que ϵ é um valor limiar que caracteriza o desbalanceamento aceitável dos polos para que uma partícula seja considerada como sendo da interface.

5.2 Geração da Malha de Triângulos

Com o campo de cor, conseguimos separar as partículas em três grupos. As que possuem esse campo negativo, as que possuem o campo positivo e as que possuem o valor do campo próximo de zero. Logo, estas últimas representam a iso-superfície (Figura 19).

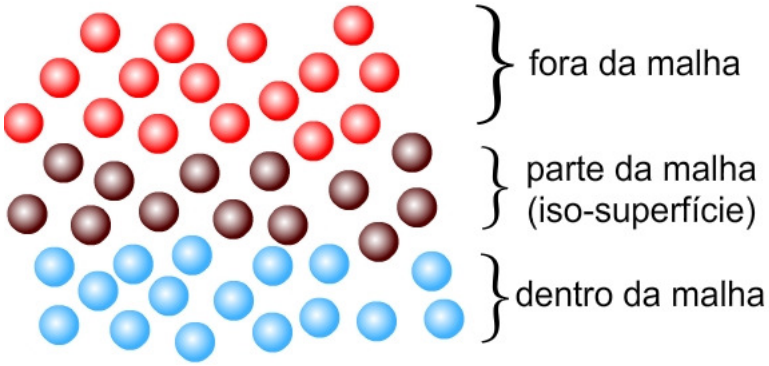

Figura 19 - Representação da separação das partículas e distinção das partículas pertencentes à iso-superfície

Dada esta distinção, é necessário inserí-las num *grid* tridimensional regular com o valor de cada campo de cor para que possa ser gerada a malha através do algoritmo de *Marching Cubes*. Portanto, dado um grid que segmente o espaço de domínio em pequenos cubos, cada partícula pode ser representada por um vértice de cubo (Figura 20). O vértice representante recebe um valor referente ao campo de cor calculado anteriormente.

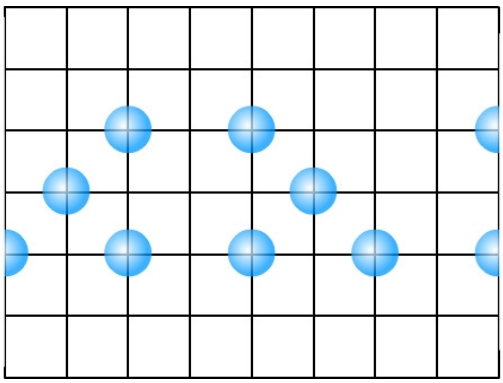

Figura 20 - Representação bidimensional dos vértices representantes das partículas da interface no grid

Há mais vértices no grid do que partículas na simulação. Para manter uma continuidade no grid e evitar "buracos" na malha que gerem incoerência na visualização, cada vértice representante influencia vértices vizinhos de acordo com um raio *r* (Figuras 21 e 22). A influência é inversamente proporcional à distância entre eles.

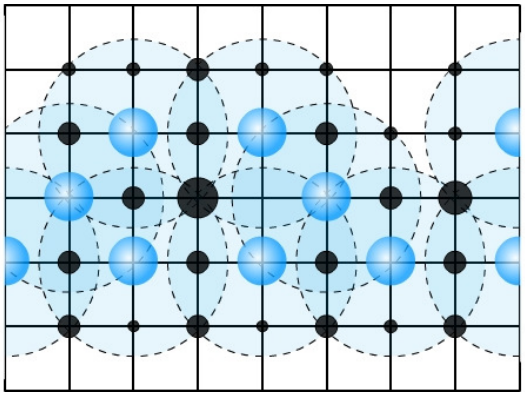

Figura 21 - Representação bidimensaional da influência de vértices representantes em vértices vizinhos

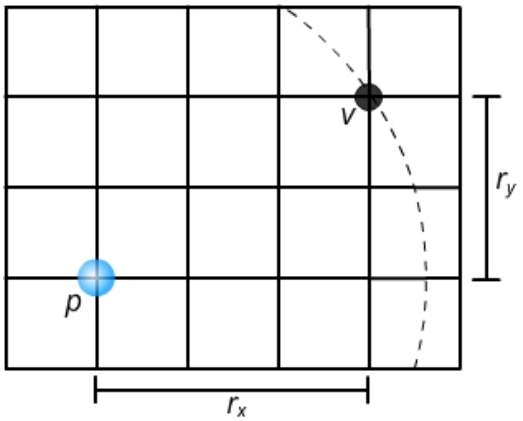

Figura 22 - Detalhe do raio de influência

Cada vértice vizinho recebe uma fração do valor que o vértice representante possui. Para determinarmos o quanto essa influência afeta cada vértice vizinho, foi desenvolvido um fator peso ω, que pode ser calculado como:

$$\omega = \frac{3}{r_x^2 + r_y^2 + r_z^2 + 3} \qquad (5.1)$$

sendo que r_x , r_y e r_z são as projeções da distância nos eixos *x*, *y* e *z*, respectivamente, e são definidos como:

$$r_x = v_x - p_x \qquad (5.2)$$

$$r_y = v_y - p_y \qquad (5.3)$$

$$r_z = v_z - p_z \qquad (5.4)$$

sendo que v_x, v_y e v_z são as posições de um vértice vizinho e p_x, p_y e p_z são posições do vértice representante, ambos nas projeções dos eixos *x*, *y* e *z*,

respectivamente. O número 3 do numerador é dado para que ω tenha um valor unitário caso r_x, r_y e r_z possuam o valor nulo simultaneamente.

Os valores a serem inseridos nos vértices vizinhos são calculados realizando uma média ponderada:

$$V_i = \frac{\sum_j \ \omega_j\left(Cs\left(\vec{x}_j\right)\right)}{\sum_j \ w_j} \qquad (5.5)$$

Em seguida, a matriz tridimensional representando as partículas da interface é passado para o algoritmo de *Marching Cubes* de Lewiner *et al* (2003) apresentando o iso-valor como zero. Logo, como explicado no capítulo 4, obtém-se uma malha de triângulos com as normais dos vértices normalizadas. A malha resultante apresenta uma forma cúbica em suas extremidades (Figura 23). Isso se deve ao fato da integração da matriz espacial com os cubos lógicos do algoritmo de *Marching Cubes*.

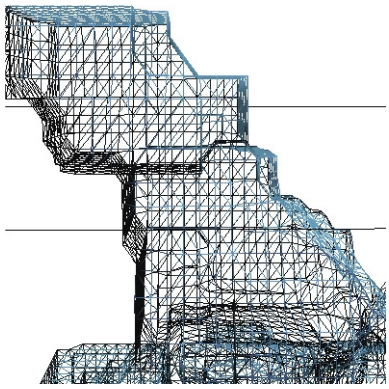

Figura 23 - Malha com extremidades de aparência cúbica

5.3 Suavização da Malha

Para contornar o problema da aparência cúbica (Figura 22), é implementada uma suavização de malha. Na literatura podemos encontrar vários algoritmos de suavização, tais como o filtro Laplaciano, *Implicit Mean Curvature Flow* e *Bilaplacian Smoothing Flow (*Belyaev e Ohtake, 2003). Buscando eficiência em conjunto com bons resultados, utilizamos, nesta dissertação, o algoritmo do filtro Laplaciano (Vollmer *et al*, 1999).

Dada uma malha **M** com uma coleção de vértices **V** = {1, 2, ..., *n*}, definimos **p** como uma função tal que **p** : **V** $\rightarrow \mathbb{R}^3$ que mapeia cada vértice *i* para sua posição espacial $\mathbf{p_i}$. O vértice que procuramos sua nova posição é denominado vértice corrente $\mathbf{V_c}$ e os vértices que compartilham alguma aresta

de algum triângulo da malha é denominado como vértice vizinho $\mathbf{V_z}$ (Figura 24). A princípio, todos os vértices $\mathbf{V_c}$ inciam em suas posições originais $\mathbf{V_o}$.

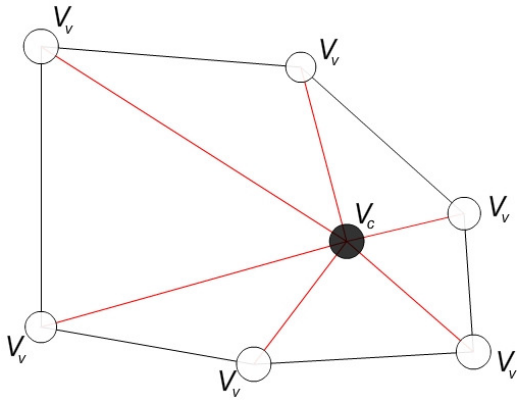

Figura 24 - Representação dos vértices vizinhos numa malha de triângulos

O Laplaciano é aplicado fazendo com que os vértices $\mathbf{V_c}$ mudem de posição de acordo com a média das posições dos vértices vizinhos (Figura 25). Logo, segue a equação:

$$p_{V_c}(i) = \alpha . p_{V_o}(i) + \frac{(1-\alpha)}{|p_{V_v}(j)|} \sum_j p_{V_v}(j) \qquad (5.6)$$

sendo α uma constante entre 0 e 1 que corresponde à intensidade da suavização.

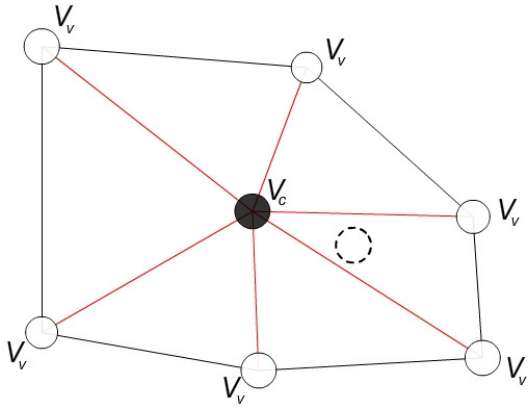

Figura 25 - Representação da nova posição do vértice e a referência de sua posição antiga com uma iteração do Laplaciano

Após aplicarmos este algoritmo, a malha apresenta formas mais suaves (Figura 26).

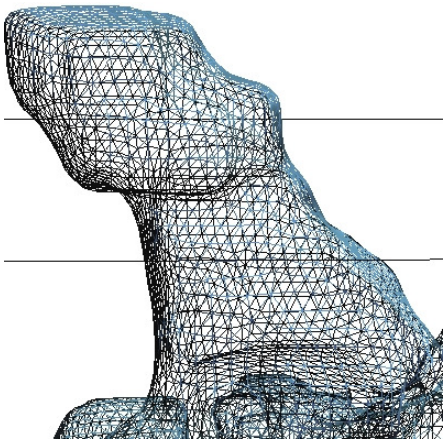

Figura 26 - Malha após o algoritmo de suavização Laplaciano

Assim com a posição do vértice, sua normal também foi calculada levando em consideração as normais dos vértices vizinhos. A equação a seguir é análoga à equação 5.6:

$$n_{V_c}(i) = \alpha . n_{V_o}(i) + \frac{(1-\alpha)}{\left| n_{V_v}(j) \right|} \sum_j n_{V_v}(j) \qquad (5.7)$$

sendo n_{V_c}a posição da normal do vértice corrente, n_{V_o} a localização do vértice original e n_{V_v} a posição da normal de um vértice vizinho. As Figuras 27 e 28 ilustram as normais dos vértices após a suavização.

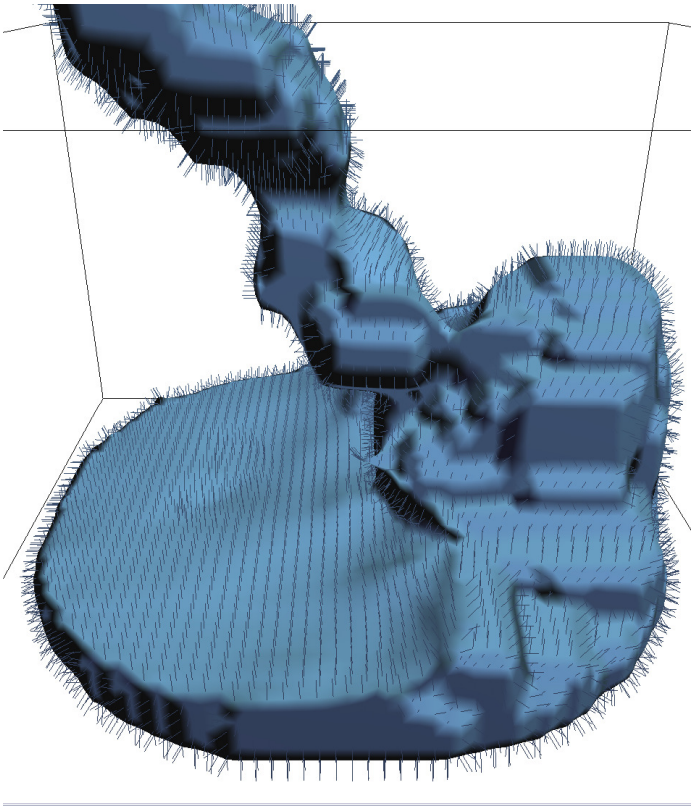

Figura 27 - Apresentação das normais dos vértices da malha de triângulos após a suavização

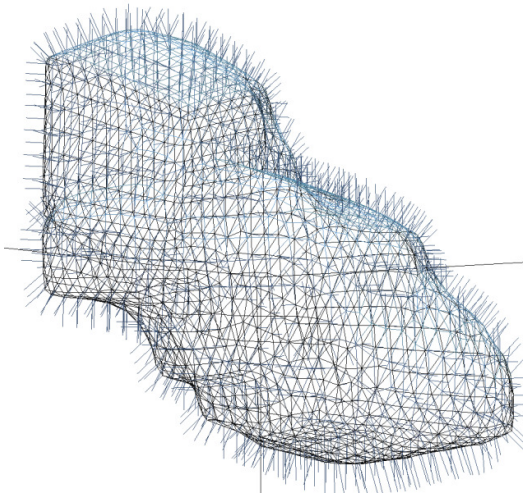

Figura 28- Apresentação detalhada das normais dos vértices da malha de triângulos após a suavização