3
Sistema de Partículas na CPU

Um sistema de partículas pode ser dividido em diferentes etapas: avanço do sistema no tempo; construção da estrutura de subdivisão espacial (geralmente a estrutura de *grid*); e tratamento das restrições e colisões.

3.1.
Avanço do Sistema

O avanço do sistema no tempo é feito com algum método de integração numérica. O método escolhido neste trabalho foi o método de Verlet. Este método calcula a nova posição da partícula com base nas duas posições anteriores, não armazenando explicitamente a velocidade. Além de ter uma precisão maior que o método de Euler, o método de Verlet torna o tratamento das restrições e colisões bastante simples.

A equação de Verlet é dada por:

$$P_{t+\Delta t} = 2P_t - P_{t-\Delta t} + a_t \Delta t^2 \quad (1)$$

E pode ser obtida através da expansão de Taylor.

$$P_{t+\Delta t} = P_t + \Delta t\, P_t^{'} + \frac{1}{2}\Delta t^2\, P''_t + \frac{1}{6}\Delta t^3\, P'''_t + O(\Delta t^4) \quad (2)$$

$$P_{t-\Delta t} = P_t - \Delta t\, P_t^{'} + \frac{1}{2}\Delta t^2\, P''_t - \frac{1}{6}\Delta t^3\, P'''_t + O(\Delta t^4) \quad (3)$$

Sendo a velocidade a primeira derivada da posição e a aceleração a segunda derivada, estas equações podem ser reescritas como:

$$P_{t+\Delta t} = P_t + \Delta t\, V_t + \frac{1}{2}\Delta t^2\, a_t + \frac{1}{6}\Delta t^3\, P'''_t + O(\Delta t^4) \quad (4)$$

$$P_{t-\Delta t} = P_t - \Delta t\, V_t + \frac{1}{2}\Delta t^2\, a_t - \frac{1}{6}\Delta t^3\, P'''_t + O(\Delta t^4) \quad (5)$$

Somando-se a eq. (4) com a eq. (5):

$$P_{t+\Delta t} + P_{t-\Delta t} = 2P_t + \Delta t^2\, a_t + O(\Delta t^4) \quad (6)$$

Rearrumando a eq. (6) e desprezando o termo de quarta ordem, obtêm-se a eq. (1).

Para adicionar um efeito artificial de amortecimento, a eq. (1) é rearrumada da seguinte maneira:

$$P_{t+\Delta t} = P_t + (P_t - P_{t-\Delta t})\delta + a_t \Delta t^2 \quad (7)$$

Onde o termo $(P_t - P_{t-\Delta t})$ é proporcional à velocidade e δ é o fator de amortecimento.

3.2. Construção da Subdivisão espacial

O segundo estágio em um sistema de partículas é a montagem da estrutura de *grid*. Esta estrutura é necessária quando existe algum tipo de interferência entre as partículas como colisão e forças. O problema de interferência entre as partículas nos leva naturalmente a uma solução quadrática. Como o número de partículas pode ser muito grande, esta solução é inaceitável. Com a estrutura de *grid*, este teste passa para $O(p.v)$, onde p é o número de partículas e v é a média de vizinhos por partículas.

Fazendo com que o tamanho das células seja igual ao diâmetro da maior partícula, garantimos que uma partícula só poderá colidir com partículas que estejam em sua própria célula ou em células vizinhas. Com isso, existem duas maneiras para abordar o problema de colisão usando uma estrutura de *grid*. A primeira é inserir a partícula na célula que contém seu centro e fazer o teste de colisão com as partículas inseridas nesta célula e em suas vizinhas. A segunda é inserir a partícula em todas as células tocadas por ela e testar colisão apenas com as partículas inseridas numa mesma célula.

Usando o primeiro método, testamos a colisão de uma partícula contra as partículas inseridas em todas as vizinhas da célula que contém seu centro. Na Figura 1, as células cinza são as consideradas no teste de colisão da partícula em preto. Mesmo sem a partícula em preto tocar na célula onde está inserida a partícula em cinza, o teste de colisão será feito entre as duas partículas.

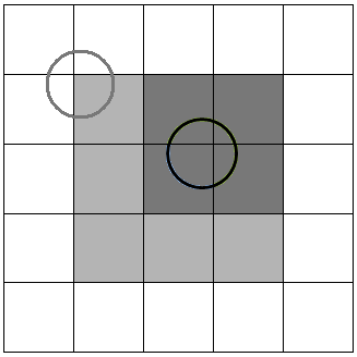

Figura 1 Inserção de uma partícula pelo centro. As células em cinza escuro são tocadas pela partícula em preto, mas todas as células em cinza são consideradas na colisão.

Este método permite que se insira a partícula no *grid* de maneira eficiente, porém a chance de ocorrer testes falsos é muito grande, prejudicando o desempenho da aplicação.

No segundo método, para o mesmo caso, nenhum teste de colisão seria feito (Figura 2). A partícula em preto estaria inserida nas células coloridas de cinza escuro e a partícula em cinza está inserida nas células cinza claro. É um método que resulta em menos testes falsos, porém torna a inserção de uma partícula no *grid* menos eficiente.

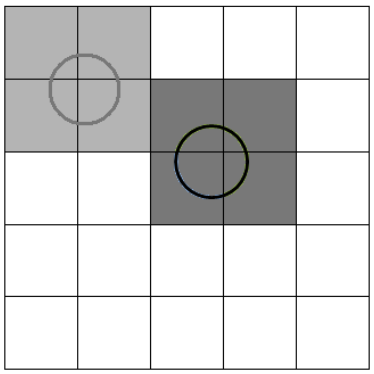

Figura 2 Partículas inseridas em todas as células em que toca. Nenhum teste de colisão seria feito.

Como neste segundo método, duas mesmas partículas podem estar inseridas em mais de uma célula, testes duplicados serão feitos a cada quadro (Figura 3). Para evitar estes testes duplicados, uma tabela é mantida indicando os testes que já foram feitos no quadro corrente. Apesar de este método ter se mostrado mais eficiente em implementações na CPU, não encontramos uma maneira eficiente de mapeá-lo para a GPU.

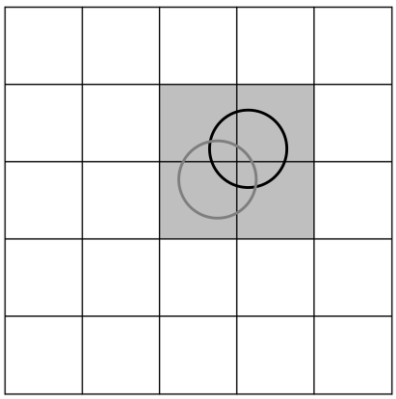

Figura 3 Partículas inseridas em várias células iguais.

Seguindo o primeiro método, para gerenciar a lista de partículas de cada célula, adotamos a estratégia de usar um vetor auxiliar para armazenar as partículas desta lista. A dimensão deste vetor auxiliar é igual ao número de partículas. A construção do *grid* é feita da seguinte maneira: a primeira tentativa é armazenar a partícula na célula que contém seu centro. Se esta célula estiver vazia, a partícula é inserida nesta célula. Se uma partícula já estiver inserida nesta célula, a partícula corrente é colocada no vetor auxiliar. O ID da partícula que está inserida na célula é usado como índice do vetor onde será armazenada a partícula corrente. Como exemplo, considere que a partícula 5 já esteja em uma célula e que a partícula 9 deveria ser inserida na mesma célula. Nesta situação, a partícula 9 seria inserida na posição 5 do vetor auxiliar. Considere ainda que a partícula 13 também seja mapeada na mesma célula. Como a célula já contém a partícula 5 e a partícula 9 já está na posição 5 do vetor auxiliar, a partícula 13 iria ser armazenada na posição 9 do vetor auxiliar. Esta situação é ilustrada pela Figura 4. Esta estratégia é apropriada para a GPU, pois a lista encadeada é montada sem a necessidade de alocação dinâmica de memória ou atualização de ponteiros.

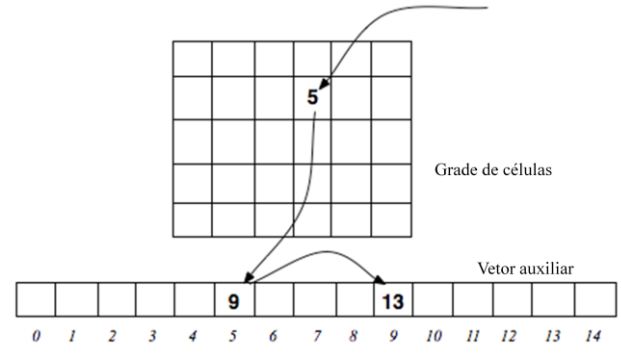

Figura 4 Lista de partículas no vetor auxiliar.

3.3. Tratamento de Restrições e colisões

O terceiro e último estágio usa o método da relaxação para fazer o tratamento de restrições, colisão entre as partículas e colisão das partículas com o ambiente. Usando o método de integração numérica de Verlet, este passo fica bastante simplificado. A idéia geral do tratamento das restrições e das colisões é projetar as partículas para uma posição válida. Isto corresponde a adoção de um modelo físico sem atrito com amortecimento total (restituição igual a zero).

As restrições ao movimento das partículas são geralmente implementadas através de barras rígidas. Este tipo de restrição mantém uma partícula a certa distância de um dado elemento. Este elemento pode ser um plano, uma curva ou outra partícula.

Para resolver múltiplas restrições, o método da relaxação [16] é usado. Em uma iteração, cada restrição é resolvida independente das outras restrições. A iteração é repetida até que a solução seja satisfatória. A Figura 5 mostra o método da relaxação para duas restrições de barra.

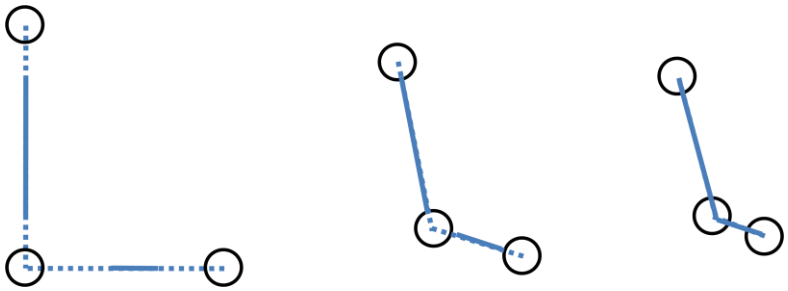

Figura 5 Uso da relaxação para resolver restrições.

No tratamento de colisão, as partículas são projetadas para uma posição válida na direção normal da superfície de colisão, que é calculada na posição da partícula. No caso de colisão entre partículas esta normal é dada pelo vetor definido pelo centro das duas partículas em colisão, e a metade do deslocamento é aplicado para cada partícula. Mais uma vez, para resolver múltiplas colisões, o método de relaxação é usado.