5
Simulação de Grãos na GPU

O objetivo de viabilizar a implementação de um sistema de partículas na GPU é aproveitar o alto poder de processamento paralelo oferecido pelos vários processadores existentes nas placas gráficas. A estratégia é desenvolver algoritmos capazes de realizar as etapas de simulação processando cada partícula de forma independente.

Os resultados de cada etapa são armazenados em estruturas dentro da própria GPU, evitando a transferência de informações entre a CPU e a GPU. As primitivas básicas de processamento são tratadas por *threads* independentes, escrevendo seus resultados de forma não concorrente na mesma estrutura de armazenamento.

A simulação de partículas na GPU pode ser resumida em quatro etapas principais:

1. Construção da Grade Uniforme
2. Detecção de Colisão
3. Cálculo das forças de contato através da Lei Força-Deslocamento
4. Integração no tempo através das Leis de Movimento

Duas soluções de simulação de partículas foram implementadas. A primeira considera apenas as forças normais de contato enquanto que a segunda considera as forças normais e tangenciais. As duas simulações utilizam as *threads* independentes do hardware gráfico para executar os algoritmos de construção da grade uniforme, detectar e calcular as forças de resposta ao contato e avaliar as equações de movimento. A *Grade 1-Partícula:1-Célula* é utilizada como estratégia de construção de grade uniforme nas duas implementações. A principal diferença entre as duas soluções é o algoritmo de detecção de contatos. No caso da simulação apenas com forças normais é possível realizar a detecção de contato e o cálculo da força normal de contato na mesma etapa de simulação. Por outro lado, a simulação com força tangencial exige o armazenamento do histórico das forças tangenciais de cada par de contatos, sendo necessária a construção de uma estrutura adicional para ser usada no próximo passo de integração.

O desempenho dos dois algoritmos de construção da grade uniforme, *Grade 1-Partícula:1-Célula* e *Grade 1-Partícula:N-Células*, é comparado na simulação apenas com forças normais e com variação de raio das partículas.

5.1 Grade Uniforme

Uma solução de construção em paralelo da grade uniforme, é utilizar ordenação na estrutura de células que compõem a grade. Uma representação válida para a estrutura da grade uniforme é formada por um vetor de pares [célula id, partícula id], e um vetor de endereçamento, como mostra a Figura 5.1. O vetor de pares possui a dimensão do número de partículas existentes na simulação, ordenado de forma crescente quanto ao id da célula. O vetor de endereçamento possui a dimensão do número de células existentes na grade uniforme onde, em cada posição é armazenado o endereço de início dos pares [célula id, partícula id] da célula, no vetor de pares da grade, e o número de partículas armazenadas na célula.

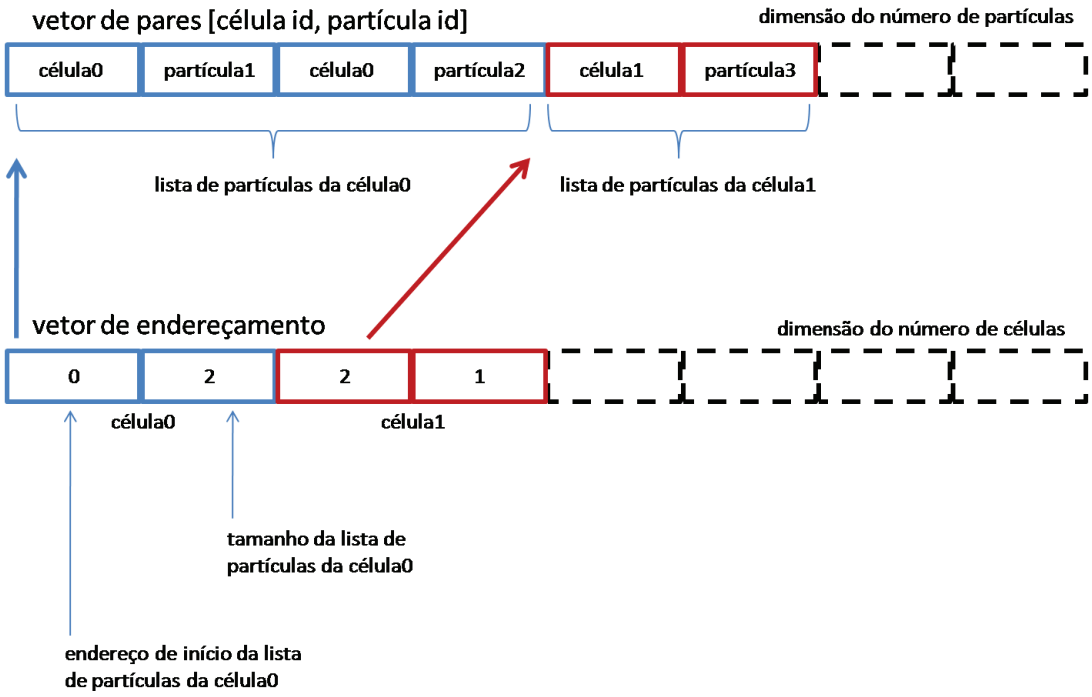

Figura 5.1: Representação da grade uniforme formada pelo vetor de pares e o vetor de endereçamento.

5.1.1 Grade Uniforme 1-Partícula:1-Célula

O algoritmo para construção da *Grade 1-Partícula:1-Célula* é uma simplificação da solução proposta por Ivson *et al.* (8). Pelo fato de cada partícula

ser armazenada apenas na célula do seu centro geométrico, algumas etapas do algoritmo proposto por Ivson podem ser eliminadas.

A construção desta grade uniforme pode ser resumida nos seguintes passos:

1. Preencher o vetor de pares [célula id, partícula id]
2. Ordenar o vetor do Passo 1, em ordem crescente quanto ao id da célula
3. Preencher o vetor de endereçamento com o endereço e o tamanho do bloco de cada célula

Passo 1: Preencher o vetor de pares [célula id, partícula id]

Neste Passo, cada *thread* processa uma partícula. Este passo utiliza como dado de entrada a posição da partícula e algumas informações das dimensões da grade uniforme como: o ponto mínimo do domínio, o tamanho da célula e o número de células existentes na grade. A saída deste passo é o vetor de pares [célula id, partícula id] ordenados pelo id da partícula.

Para cada partícula, dada sua posição, calcula-se a célula que contém o seu centro geométrico, identificando o id da célula que irá armazená-la (Figura 5.2). Este par [célula id, partícula id] é escrito na estrutura do vetor de pares na posição correspondente da partícula. Ao final do processamento de todas as partículas, o vetor de pares da grade uniforme possuirá todos os pares ordenados, por construção, pelo id da partícula.

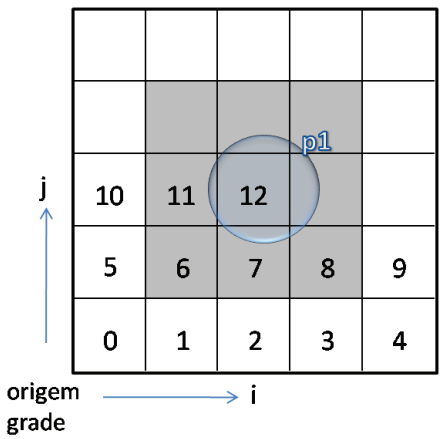

Figura 5.2: Cálculo do id da célula que contém o centro geométrico de $p1$, em relação a origem da grade.

A Figura 5.3 mostra a estrutura de armazenamento do vetor de pares com o resultado do Passo 1, utilizando como exemplo as 4 partículas da Figura 4.2.

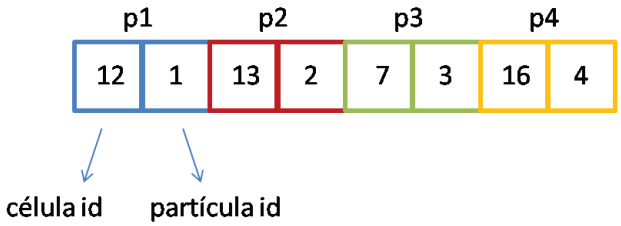

Figura 5.3: Resultado do Passo1 para o exemplo de 4 partículas.

Passo 2: Ordenar os pares do Passo 1, crescente quanto ao id da célula

No Passo 2, uma operação de ordenação é realizada para organizar os pares, em ordem crescente quanto ao id da célula. A entrada deste passo é o vetor de pares preenchido no Passo 1. A saída deste passo é o mesmo vetor de pares, porém agora, ordenado de forma crescente quanto ao id da célula. Este procedimento é necessário pois a aplicação precisa acessar a lista de partículas de uma dada célula. O procedimento deste passo é feito através da biblioteca de ordenação CUDPP (20), baseada no algoritmo de *radix-sort*, e implementada em CUDA. A Figura 5.4 mostra o vetor de pares resultante deste passo.

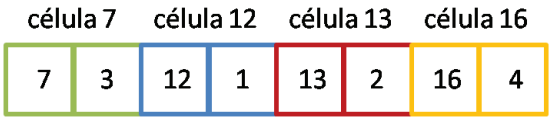

Figura 5.4: Resultado do Passo2 com a grade ordenada do exemplo de 4 partículas.

Passo 3: Preencher o vetor de endereçamento com o endereço e o tamanho do bloco de cada célula

Neste passo, cada *thread* processa uma célula. A entrada deste passo é formada pela posição da partícula e o vetor de pares resultante do Passo 2. A saída é o vetor de endereçamento preenchido com o endereço de início e o tamanho do bloco da célula. A Figura 5.5 mostra o processamento da célula de id 12, do exemplo de 4 partículas da Figura 4.2. O algoritmo executado em cada *thread* realiza uma busca binária no vetor de pares da grade uniforme para encontrar o bloco correspondente ao id da célula. Uma vez encontrado o bloco, o par [endereço bloco, tamanho bloco] é escrito no vetor de endereçamento da grade, na posição correspondente da célula. As células que não possuem partículas terão endereço zero e tamanho zero.

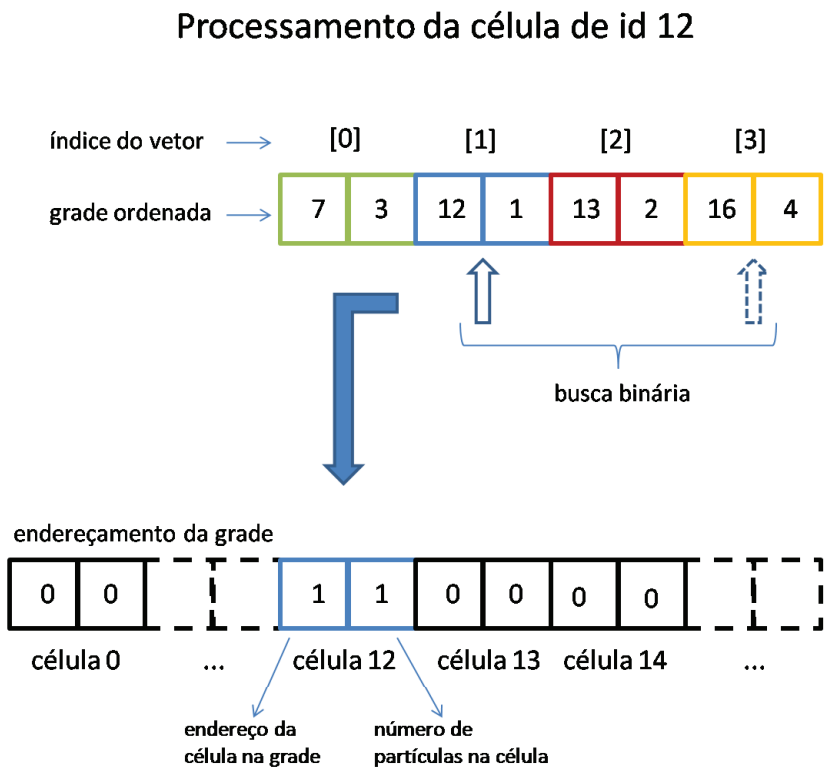

Figura 5.5: Algoritmo para processar o endereço da célula de id 12 no vetor de endereçamento da grade uniforme.

A Figura 5.6 ilustra o resultado do endereçamento das células que possuem partículas, na grade do exemplo da Figura 4.2.

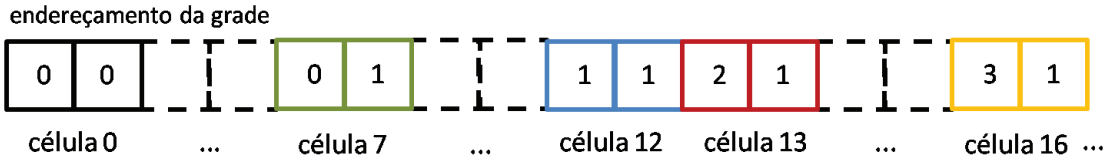

Figura 5.6: Vetor de endereçamento da grade uniforme do exemplo de 4 partículas.

5.1.2
Grade Uniforme 1-Partícula:N-Células

O algoritmo de construção da *Grade Uniforme 1-Partícula:N-Células* foi proposto por Ivson *et al.* (8) para o armazenamento de triângulos, no procedimento de traçado de raios. O principal objetivo da estratégia adotada neste algoritmo é o tratamento de elementos de diferentes tamanhos. No caso do presente trabalho, o algoritmo será utilizado para simular partículas com variação de raio. Assim como a *Grade 1-Partícula:1-Célula*, a representação da *Grade 1-Partícula:N-Células* também é formada por um vetor de pares [célula id, partícula id], em ordem crescente quanto ao id da célula, e um vetor de endereçamento.

5.2 Simulação Apenas com Força Normal

Na simulação apenas com forças normais, o cálculo da velocidade angular, o momento e o atrito aplicado sobre as partículas são desconsiderados. Do modelo descrito no Capítulo 3, somente as variáveis de translação são calculadas. O fluxo de processamento para a simulação apenas com forças normais possui uma pequena diferença no tratamento das colisões entre as partículas. A simulação constrói a *Grade 1-Partícula:1-Célula*, porém, executa as etapas de detecção e de cálculo das forças de contato unificadas na mesma etapa de simulação. Na etapa final, as Leis de Movimento são utilizadas para avançar o sistema para o próximo passo de integração. Em termos de força de contato, apenas a força resultante exercida sobre cada partícula é armazenada para aplicação no próximo passo de integração, como mostra a Figura 5.7.

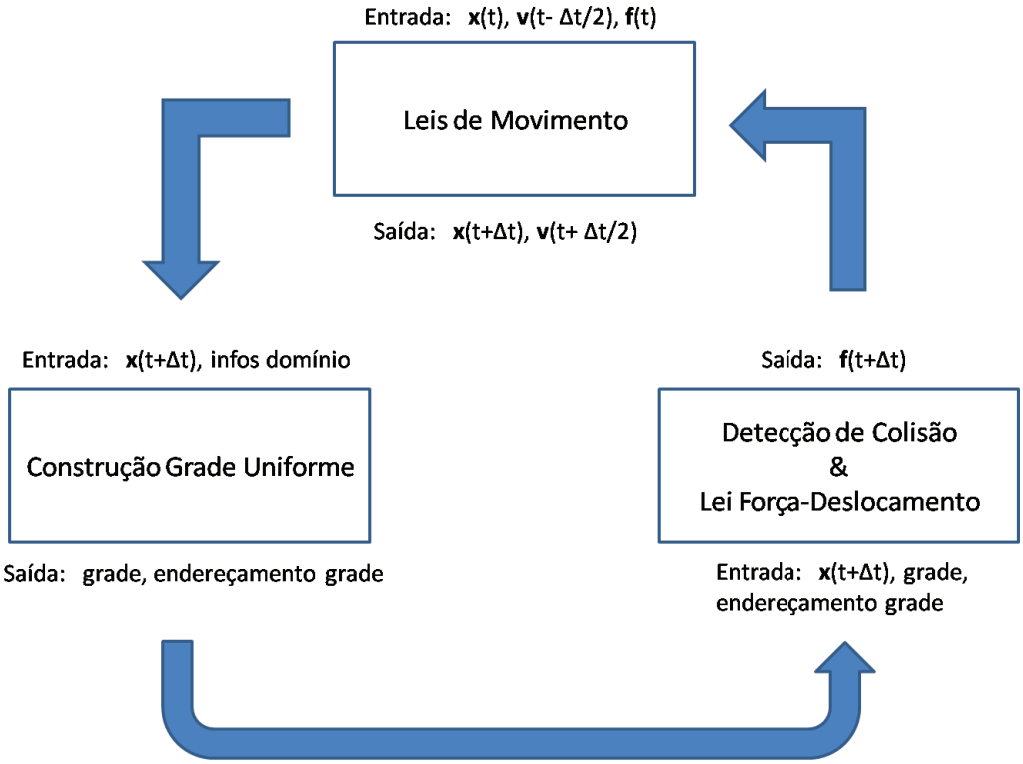

Figura 5.7: Fluxo de dados entre as etapas da simulação apenas com força normal.

5.2.1 Implementação da Lei Força-Deslocamento

A força resultante exercida sobre cada partícula é calculada nesta etapa da simulação. A entrada desta etapa de simulação é formada pela grade uniforme, atualizada com o posicionamento corrente das partículas, a posição de cada partícula e o valor da rigidez normal (k_n) do material da partícula. A saída desta etapa de

simulação é uma estrutura preenchida com a força resultante que deve ser aplicada a cada partícula. Cada *thread* processa uma partícula. Dado o id da partícula, sua posição é consultada e a célula que contém seu centro geométrico é calculada. O vetor de endereçamento da grade é utilizado para acessar as listas de partículas contidas nesta própria célula e nas 27 células vizinhas. Caso seja detectada colisão entre partículas, a força normal de contato é calculada e somada à força resultante da partícula em processamento.

Após a verificação de colisão apenas entre partículas, uma nova passada é realizada verificando a existência de colisão entre cada partícula e cada obstáculo presente no ambiente. Para isso, cada *thread* processa novamente uma partícula, consultando a equação do plano de cada parede plana. A força de contato com o obstáculo também é calculada e somada à força resultante.

O resultado desta etapa é uma estrutura formada pela força resultante que será aplicada sobre cada partícula no próximo passo de integração. O armazenamento somente da força resultante de cada partícula é a principal diferença entre a simulação apenas com forças normais e a simulação com forças normais e tangenciais.

5.2.2 Implementação das Leis de Movimento

A integração numérica das Leis de Movimento é a última etapa da simulação. Seu objetivo é calcular a velocidade e a posição de cada partícula em cada passo de integração. A entrada desta etapa de simulação é formada pela posição e velocidade do passo de integração anterior e a força resultante calculada na etapa da Lei Força-Deslocamento. A saída são as estruturas que armazenam a nova posição e a nova velocidade de cada partícula. Considerando as equações descritas na Seção 3.1, são utilizadas duas estruturas de armazenamento, com dimensão do número de partículas do modelo, para guardar a posição corrente, $\mathbf{x}(t)$, e a posição do passo seguinte, $\mathbf{x}(t+\Delta t)$. Outro par de estruturas de armazenamento, também com dimensão do número de partículas, é utilizado para guardar a velocidade no passo $(t-\Delta t/2)$ e no passo $(t+\Delta t/2)$.

O procedimento de execução deste passo é o processamento de cada partícula por cada *thread*, com as informações de posição, velocidade e força resultante da partícula sendo consultadas, além de informações de raio e massa da partícula. As Equações 3-11, 3-9 e 3-8 (repetidas abaixo apenas para facilitar a leitura) são calculadas, nesta ordem, para atualizar as informações de posição e velocidade da partícula para o novo passo de integração

$$\mathbf{f}^d = -\alpha\|\mathbf{f}\|\frac{\mathbf{v}}{\|\mathbf{v}\|},$$

$$\mathbf{v}^{(t+\Delta t/2)} = \mathbf{v}^{(t-\Delta t/2)} + \left(\frac{\mathbf{f}}{m} + g + \frac{\mathbf{f}^d}{m}\right)\Delta t,$$

$$\mathbf{x}^{(t+\Delta t)} = \mathbf{x}^t + \mathbf{v}^{(t+\Delta t/2)}\Delta t.$$

Os resultados de posição e de velocidade serão escritos nas posições correspondente de cada partícula, nas estruturas de armazenamento de saída.

5.3 Simulação com Força Normal e Tangencial

A simulação de grãos com forças normais e tangenciais, como descrito no Capítulo 3, possui grandes desafios para seu mapeamento na GPU, ainda mais com o objetivo de alto desempenho. A principal dificuldade é a necessidade de armazenar o histórico das forças tangenciais para cada contato entre as partículas, a cada novo passo de integração. Isto gera a necessidade de separar a etapa de detecção de contato da etapa de cálculo das forças de contato. A Figura 5.8 mostra a sequência de dados que são calculados em cada etapa da simulação.

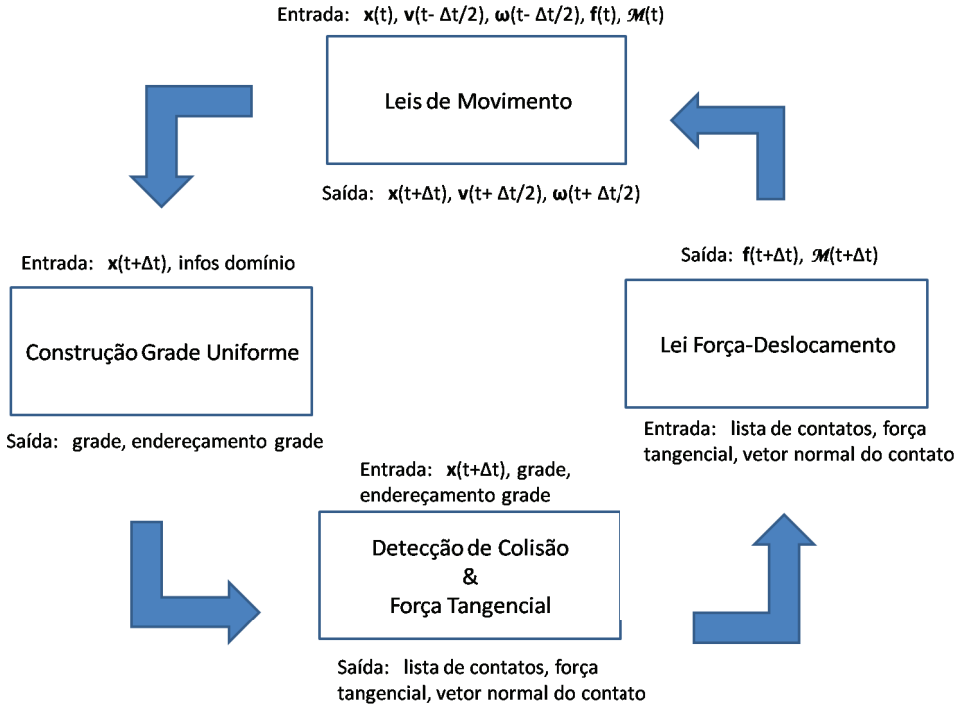

Figura 5.8: Fluxo de grandezas que são calculadas em cada etapa da simulação com forças normais e tangenciais.

O algoritmo de detecção de contato realiza o mesmo procedimento feito para a simulação com apenas forças normais, detalhada na Seção 5.2.1. Porém, além de

pesquisar pelas colisões de uma determinada partícula, os ids dos pares de partículas em colisão são armazenados em uma estrutura adicional, juntamente com a direção normal e a força tangencial do contato. Esta estrutura adicional será utilizada no passo de integração seguinte para calcular o valor absoluto da nova força tangencial sobre a partícula.

O algoritmo de construção da grade uniforme utilizada nesta simulação é o mesmo da simulação apenas com força normal. A *Grade 1-Partícula:1-Célula* é construída após a atualização da posição das partículas por meio das Leis de Movimento.

Nesta simulação com forças normais e tangenciais, tanto as grandezas de translação, quanto as de rotação, são calculadas e aplicadas sobre as partículas. Isto aumenta a complexidade de dados que devem ser consultados e calculados por cada *thread*. A simulação completa possui um custo computacional muito mais elevado do que a simulação com apenas força normal. Isto faz aumentar ainda mais o desafio por ganho de desempenho com a GPU.

5.3.1
Construção da Lista de Contatos

O desenvolvimento do algoritmo de construção da lista de contatos é um dos mais importantes para o desempenho do sistema na GPU. O objetivo desta etapa é construir a estrutura necessária para armazenar o histórico da força tangencial de cada contato entre as partículas. A entrada, desta etapa de simulação, é a posição atualizada de cada partícula e a estrutura de armazenamento da grade uniforme. A saída é uma estrutura com os pares de contato entre as partículas, a força tangencial e o vetor direção normal de cada contato entre cada par de partículas. A solução adotada, neste trabalho, para o algoritmo de construção da lista de contatos é análoga à solução de construção da *Grade 1-Partícula:N-Células*, proposta por Ivson *et al*. (8).

Apesar do objetivo desta etapa ser a construção dos pares de contato entre as partículas, foi adicionada a este algoritmo a tarefa de calcular a direção normal e a força tangencial necessárias para o próximo passo de integração. Ao final de todo o algoritmo a estrutura estará dividida por blocos de contato de cada partícula, onde são escritos, em duas estruturas separadas, o id da outra partícula em contato, a força tangencial e a direção normal do contato, viabilizando o armazenamento do histórico da força tangencial.

O algoritmo proposto possui os seguintes passos, detalhados posteriormente:

1. Para cada partícula, contabilizar todos os seus contatos

2. Acumular o Passo 1 para obter o endereço de início do bloco de contatos de cada partícula

3. Identificar os pares de partículas em contato

4. Calcular e armazenar a força tangencial e o vetor normal de cada contato

Passo 1: Contabilizando o total de contatos de cada partícula

O objetivo deste passo é dimensionar o tamanho necessário para armazenar a lista de contatos entre todas as partículas. Cada *thread* processa uma partícula. A entrada deste passo é a posição atualizada de cada partícula e a estrutura da grade uniforme. A saída é um vetor do tamanho do número de partículas, onde na posição correspondente a cada partícula será escrito seu número total de contatos.

Para contabilizar o número de contatos de cada partícula, é realizado o mesmo procedimento de detecção de contato usado na simulação apenas com forças normais. Dado o id de uma partícula, a célula que contém seu centro geométrico é calculada e a lista de partículas desta célula e de suas 27 células vizinhas é acessada para verificação de contato. Porém, ao invés de calcular a força de contato, apenas são contabilizados o número total de contatos. A Figura 5.9 mostra o procedimento feito por uma *thread* para contabilizar 1 contato da partícula $p1$.

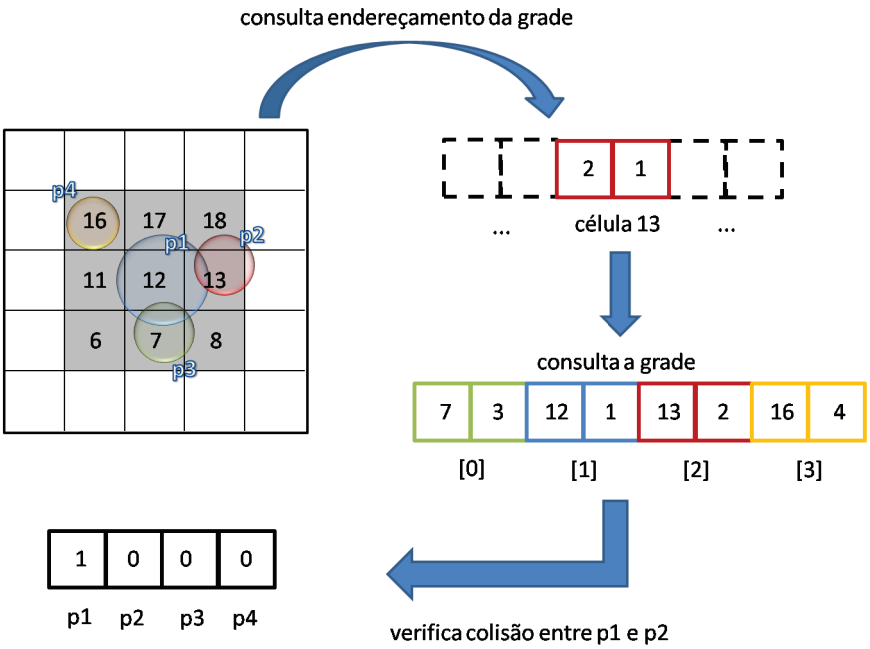

Figura 5.9: Etapas para contabilizar o contato entre p1 e p2.

No caso do exemplo apresentado na Figura 5.9, a célula de id número 13 possui a partícula $p2$ armazenada. Ao consultar a célula 13 no vetor de endereçamento da grade uniforme, obtém-se a posição número 2 para o endereço do bloco da célula

13 no vetor de pares, e o tamanho de apenas uma partícula armazenada na célula. Consultando o vetor de pares na posição 2 obtém-se o par [13, 2]. Neste caso, verifica-se a colisão entre a partícula $p1$ e a partícula $p2$, contabilizando 1 contato para a partícula $p1$. A Figura 5.10 mostra o resultado final deste passo do algoritmo com o número de contatos do exemplo acima, para todas as 4 partículas.

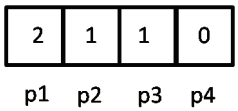

Figura 5.10: Total de contatos por partícula.

Passo 2: Acumulando o resultado do Passo 1 para obter o endereço de início do bloco de contatos de cada partícula

Os valores do vetor resultante do Passo 1, utilizado como entrada deste passo, são acumulados sequencialmente, de forma a se obter a posição de início de cada bloco de contatos de uma determinada partícula. O último valor deste vetor acumulado corresponde ao número total de contatos na cena, e será utilizado para dimensionar o tamanho necessário para armazenar a estrutura de contatos. A saída deste passo é o mesmo vetor de entrada, porém, com seus valores acumulados sequencialmente, como na Figura 5.11. O bloco de $p1$ começa na posição 0, o bloco de $p2$ começa na posição 2, já que $p1$ possui dois contatos, e assim, sucessivamente. O total indica que existem 4 contatos na cena, lembrando que o contato de $p1$ com $p3$, por exemplo, também é contabilizado como contato de $p3$ com $p1$. O algoritmo é realizado através do uso da biblioteca CUDPP (20).

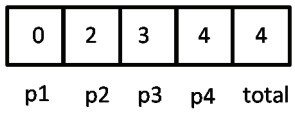

Figura 5.11: Contagem de contatos acumulada.

Passo 3: Identificando os pares de partículas em contato

Esta etapa realiza o mesmo procedimento de detecção de contato feito no Passo 1. Porém, o objetivo deste passo é armazenar na saída os ids das partículas em contato. Cada *thread* processa um contato. A entrada deste passo é a posição atualizada das partículas, a estrutura de grade uniforme e o vetor resultante do Passo 2. A saída deste passo são dois vetores com o id da partícula j em contato, escrito na posição correspondente da partícula i, sendo um vetor destinado a armazenar a força tangencial do contato e o outro para armazenar a direção normal do contato, como mostra a Figura 5.12.

O id do contato a ser processado é utilizado em uma busca binária feita sobre o vetor acumulado do Passo 2. Ao encontrar no vetor o valor imediatamente abaixo ao id do contato, o índice desta posição no vetor corresponde ao id da partícula *i* em contato. O valor armazenado no vetor, nesta posição, corresponde ao endereço do início do bloco de contatos da partícula *i* na estrutura final de armazenamento deste passo. A diferença entre o valor da posição seguinte no vetor acumulado, e o valor desta posição, corresponde ao número de contatos da partícula *i*.

A informação do id da partícula *i* é utilizada para realizar o mesmo procedimento feito no Passo 1. A célula onde a partícula *i* está armazenada é calculada, de forma a possibilitar a consulta na grade uniforme da lista de partículas em potencial colisão. Ao encontrar a partícula *j* em contato, seu id será escrito na posição correspondente do contato, no endereço do bloco da partícula *i*, calculado anteriormente. O resultado deste passo para o exemplo das 4 partículas da Figura 4.2 pode ser observado na Figura 5.12.

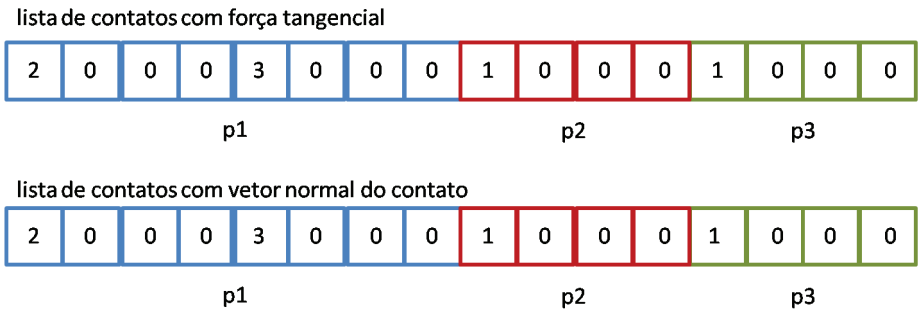

Figura 5.12: Lista de contatos com as estruturas de força tangencial e vetor normal do contato.

Passo 4: Calculando e armazenando a força tangencial e o vetor normal de cada contato

Neste passo do algoritmo, a força tangencial e o vetor normal do contato são calculados. A entrada deste passo é a lista de contatos construída no passo anterior, a posição e as velocidades linear e angular de cada partícula. A saída será a mesma lista de contatos, porém, com os valores da força tangencial e do vetor normal preenchidos. Assim como no passo anterior, o processamento neste passo também é feito para cada contato. Consultando a lista de contatos do passo anterior, os ids das partículas em contato são identificados e suas posições e velocidades lineares e angulares são consultadas para calcular o incremento da força tangencial e o vetor normal do contato. Abaixo são repetidas as equações descritas na Seção 3.3 e utilizadas neste passo do algoritmo.

$$\mathbf{v}_c = \{\mathbf{v}^B + [\omega^B \times (\mathbf{x}_c - \mathbf{x}^B)]\} - \{\mathbf{v}^A + (\omega^A \times (\mathbf{x}_c - \mathbf{x}^A)]\}$$

$$\mathbf{v}_{ct} = \mathbf{v}_c - \mathbf{v}_{cn} = \mathbf{v}_c - (\mathbf{v}_c \cdot \mathbf{n})\mathbf{n},$$

$$\Delta\mathbf{u}_t = \mathbf{v}_t \Delta t,$$

$$\Delta\mathbf{f}_t = -k_t \Delta\mathbf{u}_t.$$

Para calcular o valor absoluto da força tangencial entre as duas partículas, é necessário obter a força tangencial e o vetor normal do contato no passo de integração anterior. Se o contato só teve início neste passo de integração da simulação, então significa que a força tangencial no passo de integração anterior era nula. Sendo assim, a força tangencial resultante do passo atual é igual ao seu incremento. Porém, se o contato já existia, então a força tangencial anterior e o vetor normal do contato são consultados para realizar os cálculos de rotação para atualizar a força tangencial, através das equações abaixo, como descrito na Seção 3.3.

$$\mathbf{f}_{t\ rot1} = \mathbf{f} + (\mathbf{c} \times \mathbf{f})$$

$$\mathbf{f}_{t\ rot2} = \mathbf{f}_{t\ rot1} + (\overline{\omega} \times \mathbf{f}_{t\ rot1})$$

$$\mathbf{c} = \mathbf{n}^{old} \times \mathbf{n},$$

$$\overline{\omega} = \frac{1}{2}[(\omega^A + \omega^B) \cdot \mathbf{n}]\mathbf{n},$$

$$\mathbf{f}_t = \{\mathbf{f_t}\}_{rot2} + \Delta\mathbf{f}_t.$$

A Figura 5.13 mostra o resultado deste passo do algoritmo, ou seja, a lista de contatos, com o histórico da força tangencial e do vetor normal preenchido. No próximo passo de integração, essas duas estruturas serão utilizadas neste mesmo passo do algoritmo para calcular o valor absoluto da nova força tangencial de contato.

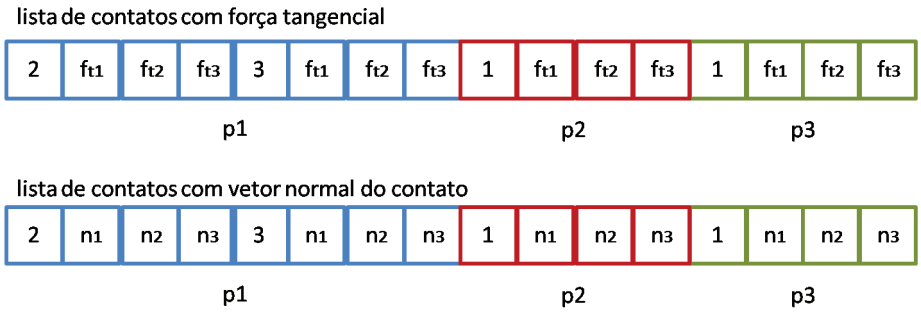

Figura 5.13: Lista de contatos com o histórico da força tangencial e do vetor normal do contato.

Todo o procedimento feito neste passo para calcular e armazenar a força tangencial no contato entre as partículas, é também realizado para considerar a força tangencial entre as partículas e os obstáculos do ambiente. As informações da equação do plano de cada parede são consultadas pela *thread* e a força tangencial é armazenada da mesma forma que na colisão entre as partículas. Ao final do procedimento de colisão com obstáculos, duas novas estruturas, com o mesmo formato das estruturas da força entre partículas, são armazenadas para uso na força resultante da partícula.

5.3.2 Implementação da Lei Força-Deslocamento

A etapa de cálculo da Lei Força-Deslocamento na simulação com forças normais e tangenciais, assim como na simulação apenas com força normal, tem por objetivo calcular as forças resultantes que devem ser aplicadas sobre cada partícula. A força e o momento calculados nesta etapa serão usados na etapa de Leis de Movimento para atualizar a posição e as velocidades lineares e angulares das partículas para o próximo passo de integração.

A entrada desta etapa é formada pela lista de contatos, que possui na sua estrutura a força tangencial e o vetor normal de cada contato, construída na etapa anterior, além da posição atualizada das partículas. A saída é formada por duas

estruturas, do tamanho do número de partículas, onde são armazenadas as forças resultantes e os momentos angulares resultantes.

Cada *thread* processa uma partícula. Dado o id de uma determinada partícula, este é utilizado para consultar o endereço correspondente de seu bloco de contatos, no vetor acumulado que foi construído no Passo 2 do algoritmo. Após a identificação das partículas em colisão, a força normal do contato é calculada e somada à força tangencial, calculada na etapa anterior.

Após calcular a força de contato entre partículas, são verificadas as colisões entre as partículas e os obstáculos presentes no ambiente, através de uma nova passada. A lista de contatos entre partículas e obstáculos, construída na etapa anterior, é consultada para somar a força tangencial de contato à força normal calculada. A soma das forças de contato devido à colisão entre partículas e obstáculos e devido à colisão apenas entre partículas, é usada para formar apenas uma força resultante. O momento angular resultante da partícula também é calculado considerando as partículas e os obstáculos. Apenas a força resultante e o momento angular resultante de cada partícula serão armazenados nas estruturas de saída ao final desta etapa.

5.3.3
Implementação das Leis de Movimento

A etapa de Leis de Movimento para a simulação com forças normais e tangenciais é bastante similar ao da simulação apenas com forças normais. Porém, na simulação com força normal e tangencial são calculadas todas as grandezas descritas na Seção 3.1. A entrada desta etapa são as estruturas de posição, velocidades lineares e angulares, força resultante e momento angular resultante, referentes ao passo de integração anterior. A saída são as estruturas com a nova posição, novas velocidades lineares e angulares de cada partícula para o novo passo de integração. O processamento de cada *thread* é feito sobre cada partícula. Além das equações mencionadas nesta etapa da simulação apenas com força normal, as Equações 3-10 e 3-12 também são calculadas para armazenar o novo estado das partículas.

$$\mathcal{M}^d = -\alpha \|\mathcal{M}\| \frac{\omega}{\|\omega\|}.$$

$$\omega^{(t+\Delta t/2)} = \omega^{(t-\Delta t/2)} + \left(\mathcal{M} + \mathcal{M}^d\right)\frac{\Delta t}{I},$$

O resultado das grandezas calculadas é escrito, de forma simultânea, por cada *thread* na posição correspondente de cada partícula, em várias estruturas diferentes de armazenamento.