UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DIEGO VASCONCELOS SCHARDOSIM DE MATOS

Simulação Física Simplificada em Web

RIO DE JANEIRO
2025

DIEGO VASCONCELOS SCHARDOSIM DE MATOS

Simulação Física Simplificada em Web

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Cláudio Esperança

RIO DE JANEIRO

2025

CIP - Catalogação na Publicação

R484t

Ribeiro, Tatiana de Sousa
Titulo / Tatiana de Sousa Ribeiro. -- Rio de Janeiro, 2018.
44 f.

Orientador: Maria da Silva.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2018.

1. Assunto 1. 2. Assunto 2. I. Silva, Maria da, orient. II. Titulo.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a).

DIEGO VASCONCELOS SCHARDOSIM DE MATOS

Simulação Física Simplificada em Web

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em ___ de _____ de _____

BANCA EXAMINADORA:

Cláudio Esperança
Titulação (Instituição)

Nome do Professor1
Titulação (Instituição)

Nome do Professor2
Titulação (Instituição)

"The display is the computer."

Jen-Hsun Huang

RESUMO

Neste trabalho é descrito elementos básicos para um esquema de modelagem baseado em física de objetos rígidos e deformáveis bem adequado para aplicações interativas que é simples, rápida e bastante estável. Estes corpos são representados por um grupo de partículas que devem satisfazer um conjunto de restrições lineares por um método de relaxamento e a simulação de seu movimento é usado um esquema de integração Verlet. A detecção das colisões são tratadas em duas fases: de *Broad Phase* responsável por reduzir o número de candidatos à colisão com estruturas de divisão espacial e uso de testes rápidos e baratos; de *Narrow Phase* responsável por usar algoritmos mais sofisticados para detectar colisão como o Teorema do Eixo Separador (Separating Axis Theorem - SAT) e algoritmo de Gilbert, Johnson e Keerthi (1988) (GJK). Para a resposta a colisão é usado as técnicas descritas por Jakobsen (2001) pela projeção das posições das partículas envolvidas através do Vetor de Translação Mínimo (minimum translation vector - MTV). Diferente das abordagens tradicionais, a simulação física é obtida sem se computar explicitamente matrizes de orientação, torques ou tensores de inércia.

Palavras-chave: Computação gráfica; Simulação Física; Detecção de Colisão; Resposta a Colisão; Corpos rígidos e deformáveis; web.

ABSTRACT

Abstract in english. The text should be typed in a single paragraph with **single spacing** and contain between 150 and 500 words. Use the third person singular, the verbs in the active voice and avoid the use of symbols and contractions that are not of current use. The keywords must appear right below the abstract, preceded by the expression **Keywords:**, separated by a semicolon (;) and ending with a period. They must be written with the initials in lowercase, with the exception of proper nouns and scientific names.

Keywords: artificial intelligence; cryptography; data mining; Sociedade Brasileira de Computação; neural network.

SUMÁRIO

1 INTRODUÇÃO

A computação gráfica é matemática e arte. Esta ferramenta proporciona um maior poder de abstração, ajudando na criação de imagens complexas e em muitos casos não imaginadas. A computação gráfica pode ser encarada como uma ferramenta não convencional que permite ao artista transcender das técnicas tradicionais de desenho ou modelagem (AZEVEDO, 2003).

A computação gráfica vista como ferramenta indicaria que temos um artista responsável pela arte gerada. Mesmo as imagens geradas a partir de equações podem ser consideradas arte, se essas equações forem fruto da criatividade e da capacidade do descobridor que manifesta sua habilidade e originalidade inventiva. A habilidade de simular a natureza em computadores tem sido objeto de atenção e curiosidade de toda a comunidade científica.

De acordo com Moller, Haines e Hoffman (2018) renderização em tempo real refere-se à criação rápida de imagens no computador. É a área mais interativa da computação gráfica. Uma imagem aparece na tela, o usuário interage ou reage, e esse feedback afeta o que será gerado em seguida.

Esse ciclo de reação e renderização acontece em uma velocidade suficientemente alta para que o usuário não veja imagens individuais, mas sim se sinta imerso em um processo dinâmico. A taxa na qual as imagens são exibidas é medida em quadros por segundo (FPS) ou Hertz (Hz). Com um quadro por segundo, há pouca sensação de interatividade; o usuário percebe claramente a chegada de cada nova imagem. A partir de cerca de 6 FPS, a sensação de interatividade começa a aumentar.

Uma taxa de quadros mais alta é importante para minimizar o tempo de resposta. Um atraso temporal de apenas 15 milissegundos pode prejudicar e interferir na interação. Como exemplo, os óculos de realidade virtual geralmente exigem 90 FPS para minimizar a latência.

Modelagem e animação baseada em física vêm sendo pesquisadas desde início desse século, encontrando aplicação em todas áreas de entretenimento, simulação, desenho assistido por computador e várias outras áreas.

Uma animação realista requer que os objetos em movimento obedeçam a leis físicas. Para tanto, vários aspectos precisam ser considerados como, por exemplo: no mundo real, dois objetos não podem ocupar o mesmo lugar no espaço ao mesmo tempo. Isto significa que objetos podem empurrar outros objetos dependendo de suas massas e velocidades; podem ser empilhados uns sobre os outros; não podem atravessar o chão, e assim por diante. Para incorporar um mecanismo que permita tratar estas situações uma tarefa importante para este mecanismo é detectar configurações onde haja interpenetração entre objetos, as quais são chamadas de "colisões".

Porém, sendo o processo de detecção de colisões geralmente muito custoso computacionalmente, a simulação de ambientes interativos dificilmente pode ser alcançada em tempo real. Isto se deve ao fato de que, a cada instante de tempo da simulação, diversas características físicas têm que ser computadas, tais como velocidades, forças, torques, momentos e outros.

O tratamento de colisões pode ser dividido em duas fases: na detecção de colisões, objetos que se interpenetram ou que estão em vias de o fazer são identificados, na resposta a colisões envolve a modificação dos diversos parâmetros físicos dos objetos envolvidos – tipicamente posição, orientação e velocidade – de tal forma que uma configuração fisicamente plausível seja obtida.

Há muitas abordagens para esse problema na literatura que envolve uso de métodos precisos que requerem computar diversas equações que regem as leis físicas, particularmente as da dinâmica como método do impulso, método da penalização

Jakobsen (2001) propôs um esquema simplificado de simulação baseada em física, conseguindo simular ambientes com objetos deformáveis e rígidos sem a necessidade de calcular explicitamente matrizes de orientação, torques ou tensores de inércia. Este esquema é baseado principalmente na implementação de restrições lineares e a com binação de diversas técnicas que se complementam:

- A dinâmica das partículas é simulada usando o integrador de Verlet.
- Esquema de resposta a colisão que consiste em projetar vértices que penetram uma determinada superfície para fora desta
- As restrições lineares são mantidas constantes usando um processo de relaxamento.
- É usada uma raiz quadrada aproximada em vez de uma exata para os cálculos de distância.
- Objetos (rígidos e deformáveis) são representados por sistemas de partículas com restrições lineares.

Entretanto, vários detalhes importantes foram suprimidos. Em particular, a ligação entre o mecanismo de detecção de colisão e os algoritmos de resposta a colisões é descrita apenas superficialmente, nenhuma estratégia é sugerida para tratar objetos que requeiram um número maior de restrições lineares e não é apresentado estruturas de otimizações para lidar com muitos objetos. Isto é relevante, já que o emprego de muitas restrições lineares pode comprometer o desempenho do sistema.

Neste trabalho estaremos focando em um protótipo de sistema baseado no Jakobsen apresentando soluções para detecção e resposta a colisão. Ele é adequado para situações de aplicações em tempo real e beneficia áreas que exigem alta imersão entre a simulação e o humano.

O restante deste trabalho é dividido nos seguintes capítulos: O Capítulo 2 apresenta as técnicas principais descritas por JAKOBSEN para animação baseada em física. No Capítulo 3 mostra as diversas representações de objetos aceitas pelos métodos anteriores. O Capítulo 4 aborda as principais metodologias para detecção de colisões e seus casos degenerados. O Capítulo 5 apresenta como calcular de forma simplificada as fronteiras dos objetos. No Capítulo 6 mostra como realizar a resposta a colisão por projeção das posições. O Capítulo 7 apresenta as otimizações usados neste trabalho. No Capítulo 8 os experimentos e comparação de resultados com outras bibliotecas. Finalmente, o Capítulo 9 aborda alguns comentários finais e sugestões para trabalhos futuros.

2 ANIMAÇÃO BASEADO EM FÍSICA

Na maioria dos casos, é preciso usar métodos sofisticados e exatos para simular a dinâmica. Porém, em aplicações de jogos interativos, a precisão não é realmente o mais importante, mas sim que o resultado final tenha uma aparência realista e que possa ser executado rapidamente.

Jakobsen (2001) apresentou um conjunto de métodos e técnicas que, unidas, conseguem atingir em grande parte estes objetivos. Estes métodos são relativamente simples de implementar (comparados com outros esquemas) e têm um bom desempenho. O algoritmo é iterativo e permite aumentar a precisão do método sacrificando parte de sua rapidez: se uma pequena fonte de imprecisão é aceita, o código pode conseguir uma execução mais rápida. Esta margem de erro ainda pode ser ajustada de forma adaptativa em tempo de execução.

Em geral, o sucesso deste método vem da combinação de varias técnicas que se beneficiam umas das outras, principalmente o uso do integrador Verlet, o tratamento de colisões usando projeção, e a resolução de restrições de distância usando relaxamento. A seguir serão descritas as componentes mais importantes da abordagem proposta por Jakobsen.

2.1 INTEGRADOR VERLET

O núcleo da simulação é um sistema de partículas, sua implementação mais simples é para cada passo calcular sua nova posição **x** e nova velocidade **v** com o sistema abaixo:

$$\begin{cases} x' = x + v \cdot \Delta t \\ v' = v + a \cdot \Delta t \end{cases}$$

Esse é o método de Euler sendo Δt o tamanho do passo, **a** é aceleração calculado a partir da lei de newton $F = ma$ onde F é a força acumulada agindo na partícula.

Existem muitos outros esquemas de integração como Runge-Kutta e outras variantes. Um método que é bem estável, sua velocidade é calculada implicitamente o quê mantém sua posição e velocidade em sincronia é o integrador de Verlet, muito popular em simulação de dinâmica molecular.

$$\begin{cases} x' = 2x - x^* + a \cdot \Delta t^2 \\ x^* = x \end{cases}$$

Sendo **x** sua posição corrente e x^* sua posição anterior, essa é chamada representação sem velocidade. Ao fim de cada passo conseguimos representar o sistema de partículas como:

$$X = \begin{pmatrix} x_1 \\ x_1^* \\ a_1 \\ \vdots \\ x_n \\ x_n^* \\ a_n \end{pmatrix}$$

O algoritmo para o integrador verlet segue como

Algorithm 1: Verlet

Input: Tamanho do passo dt

for $i \leftarrow 0$ **to** n **do**

$x \leftarrow X[i]$

$x^* \leftarrow X[i+1]$

$a \leftarrow X[i+2]$

$x_{tmp} \leftarrow x$

$X[i] \leftarrow 2 * x - x^* + a * dt$

$X[i+1] \leftarrow x_{tmp}$

$X[i+2] = 0$ `// Limpa as forças para próxima iteração`

Como dito anteriormente, as vantagens de usar esse esquema proposta por JAKOBSEN serão evidenciadas quando combinadas com as próximas técnicas.

2.2 RESTRIÇÃO LINEAR

Anteriormente foi um método numérico, porém o sucesso de Jakobsen (2001) foi conectar ideias simples para construir corpos maiores e complexos. Isso é feito atribuindo as partículas algo em comum, um conjunto de propriedades que devem ser respeitadas para atingir o efeito desejado.

A primeira propriedade - também é a base para as próximas - é chamada restrição linear ou restrição de distância. Ela apenas dita que duas partículas p_0 e p_1 devem sempre estar a uma distância d entre si. Dessa forma o conjunto de partículas devem satisfazer a todo instante uma coleção de inequações unilaterais representadas na forma:

$$|x_2 - x_1| = d \tag{2.1}$$

Mesmo que as posições das partículas estejam inicialmente corretas, depois de um passo de simulação a distância entre elas pode se tornar inválidas (através de forças externas, por exemplo). Para corrigir sua distância devemos mover (projetar) elas de tal forma que satisfaça 2.1.

Inserir figura

A essência de uma restrição é a projeção. Deve-se encontrar o movimento mínimo que a satisfaça. O efeito de uma restrição linear pode representar conectar duas partículas com uma haste rígida mas também projetar o ponto em um círculo ao redor do ponto de ancoragem.

Algorithm 2: Restrição Linear

$\vec{\delta} \leftarrow x_2 - x_1$
$\epsilon \leftarrow \frac{|\vec{\delta}|-d}{|\vec{\delta}|}$
$x_1 \leftarrow x_1 - \vec{\delta} * 0.5 * \epsilon$
$x_2 \leftarrow x_2 + \vec{\delta} * 0.5 * \epsilon$

O pseudocódigo 2 irá separar ou aproximar as partículas de tal forma que satisfaçam a distancia d. Essa situação é comparável a um sistema de molas interconectadas entre partículas de rigidez que tendem para o infinito.

2.3 RESTRIÇÃO DE REVOLUÇÃO

Na animação física queremos muitas vezes que uma partícula gire em torno de um eixo. Isso feito de forma simples, basta ter uma partícula em comum cuja função seja ser um eixo de rotação.

Inserir figura

Uma outra forma de fazer isso é usar uma restrição linear com $d = 0$.

2.4 RESTRIÇÃO ANGULAR

Em muitas situações em animação física é desejável que o ângulo formado entre duas partículas esteja restrito a um intervalo válido. Isso pode ser feito de forma simples aplicando uma restrição linear caso a distância entre duas partículas seja menor que um limiar. Ou seja, satisfazer a inequação abaixo

$$|x_2 - x_1| > 100$$

A rotina para essa restrição é tão simples quanto usar um condicional junto com o algoritmo de restrição linear.

Algorithm 3: Restrição Angular

$\vec{\delta} \leftarrow x_2 - x_1$
if $|\vec{\delta}| < d$ **then**
    $\epsilon \leftarrow \frac{|\vec{\delta}|-d}{|\vec{\delta}|}$
    $x_1 \leftarrow x_1 - \vec{\delta} * 0.5 * \epsilon$
    $x_2 \leftarrow x_2 + \vec{\delta} * 0.5 * \epsilon$

Um outro método de restringir os ângulos é satisfazer a restrição de produto interno

$$(x_2 - x_0) \cdot (x_1 - x_0) < \alpha$$

2.5 RESOLVENDO RESTRIÇÕES CONCORRENTES POR RELAXAMENTO

Na prática, uma simulação pode conter muitas restrições de todos os tipos vistos anteriormente. Para satisfazer todas elas devemos resolver todas as inequações sequencialmente, isso seria o equivalente a encontrar a solução de um sistema de equações.

Entretanto Jakobsen (2001) propõe uma abordagem de solução indireta por iteração local, basta satisfazer todas restrições e repetir um número N vezes a cada passo da simulação

Algorithm 4: Satisfazer Restrições

Input: n: número de repetições
for $i \leftarrow 0$ **to** n **do**
  ResolverRestrição(i)

Apesar dessa abordagem parecer ingênua, ao resolver todas restrições localmente e repetir, o sistema global do sistema converge para uma configuração que satisfaça todas restrições. Quanto maior o número de iterações mais rápido o sistema irá convergir para solução e também a animação irá parecer mais rígida para o usuário.

3 MODELANDO CORPOS

4 DETECÇÃO DE COLISÕES

4.1 POLÍGONOS CONVEXOS

4.1.1 Teorema do Eixo Separador (SAT)

4.1.2 Algoritmo Gilbert–Johnson–Keerthi (GJK)

4.2 POLÍGONOS CÔNCAVOS

4.3 TRATANDO CASOS DEGENERADOS

5 O FECHO CONVEXO

5.1 CAIXA DELIMITADORA ALINHADA AO EIXO COORDENADO (AABB)

5.2 QUICKHULL

6 RESPOSTA A COLISÃO

6.1 EPA

6.2 PROJEÇÃO DA POSIÇÃO PELO MÉTODO JAKOBSEN

7 OTIMIZAÇÕES

8 EXPERIMENTOS

9 CONCLUSÕES

REFERÊNCIAS

AZEVEDO, A. C. E. **Computação gráfica - Teoria e prática**. [S.l.]: Editora Campus, Ltda, 2003.

GILBERT, E.; JOHNSON, D.; KEERTHI, S. A fast procedure for computing the distance between complex objects in three-dimensional space. **IEEE Journal on Robotics and Automation**, v. 4, n. 2, p. 193–203, 1988.

JAKOBSEN, T. Advanced character physics. In: **Proceedings of the Game Developer's Conference 2001**. San Jose, CA: Game Developers Conference, 2001. Presented at Game Developer's Conference 2001.

MOLLER, T.; HAINES, E.; HOFFMAN, N. **Real-time rendering**. 4. ed. Boca Raton: CRC Press, 2018.