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 é 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.

Além disso para o 2 a constante 0,5 também tem o efeito de aumentar o passo local de convergência para solução ideal. Fisicamente pode ser interpretado como um coeficiente de restituição. Pode ser usado para representar quão abrupto as partículas se aproximam ou afastam.

3 MODELANDO CORPOS

Em animação física, convencionalmente são usadas funções paramétricas para representar sólidos simples, tais como cubos, cilindros, cones, esferas, toros, etc. Isto é feito frequentemente em ambientes interativos complexos, como jogos, onde é preciso animar milhares de objetos simples.

Um objeto geométrico é um conjunto fechado de pontos, limitado e não vazio. É fechado, pois a borda faz parte do objeto e é limitado significa que existe uma esfera de raio finito que limita o objeto. Assim, por exemplo, um plano é fechado mas não limitado. Tais objetos geométricos podem ser de dois tipos: objetos convexos e objetos côncavos. Um objeto é convexo se contém todos os segmentos de reta que conectam pares de pontos, caso contrário é chamado côncavo.

Objetos complexos podem ser compostos de objetos mais simples – tipicamente convexos.

3.1 CORPOS RÍGIDOS

Usando as ferramentas vistas no capítulo anterior a representação de corpos rígidos torna-se um passo quase natural a ser feito. Basta decompor a geometria desejada em partes menores e mais simples usando partículas para representar seus vértices e restrições lineares com coeficiente de restituição igual a 1 para representar suas arestas. No caso de 2D um quadrado pode ser representado com 4 partículas, 4 restrições lineares e mais 2 restrições internas para garantir rigidez.

Dessa forma ao colocar o objeto na simulação a integração Verlet das partículas e o relaxamento das restrições são responsáveis por mover o corpo de forma plausível, conservando momento e torque.

3.2 CORPOS DEFORMÁVEIS

A modelagem de um corpo deformável é análoga a um corpo rígido, com a principal diferença que o coeficiente de restituição é menor que 1. Dessa forma será necessário alguns passos da simulação para a restrição convergir para configuração ideal, gerando efeito de deformação do corpo.

3.3 CORPOS ARTICULADOS

Essa abordagem permite construir um modelo completo de corpo articulado bem realista usando uma combinação de: restrição linear, restrição de revolução e restrição angular.

3.4 TECIDOS

O tecido pode ser facilmente representado como uma grade uniforme de partículas conectadas por restrições lineares com coeficientes de restituição menor que 1 e apenas um passo de relaxamento.

4 DETECÇÃO DE COLISÕES

O mecanismo de detecção de colisão é um componente fundamental para um sistema de simulação baseada em física [26, 40, 29, 32, 33]. Este componente permite efetuar consultas rápidas de proximidade geométrica entre objetos durante a simulação. Isso não somente inclui verificações de como dois objetos se intersectam, mas também quando e onde a colisão ocorreu.

Neste capítulo introduzimos conceitos relevantes dentro do contexto deste trabalho e discutimos métodos que geralmente são usados para representar objetos, assim como algoritmos para fazer consultas de proximidade e interseção entre objetos.

4.1 POLÍGONOS CONVEXOS

Um objeto geométrico é um conjunto fechado de pontos, limitado e não vazio. É fechado, pois a borda faz parte do objeto e é limitado significa que existe uma esfera de raio finito que limita o objeto. Assim, por exemplo, um plano é fechado mas não limitado.

(INSERIR IMAGEM FORMA CONVEXA X NAO CONVEXA)

Uma forma é considerada convexa se, para qualquer linha desenhada através da forma, essa linha cruzar apenas duas vezes. Formas não convexas podem ser representadas por uma combinação de formas convexas.

4.1.1 Teorema do Eixo Separador (SAT)

O Teorema do Eixo Separador (do inglês Separating Axis Theorem, SAT) é um método para determinar se duas formas convexas estão se cruzando. O algoritmo também pode ser usado para encontrar o vetor de penetração mínimo que é útil para simulação de física e uma série de outras aplicações. O SAT é um algoritmo genérico rápido que pode remover a necessidade de ter código de detecção de colisão para cada par tipo de forma, reduzindo assim o código e a manutenção.

Teorema 1. *Se dois objetos convexos não estiverem penetrando, existe um eixo para o qual a projeção dos objetos não se sobreporá.*

Um plano de separação (PS) é um plano que está localizado entre dois objetos convexos, separando-os. Eventualmente, A está situado no lado positivo e B no lado negativo ou vice-versa. Matematicamente, um PS é definido por $H(v, \delta)$,onde v é chamado de eixo de separação (ES).

Se v é um eixo de separação, então existe um escalar δ tal que o plano de separação possa ser definido.

INSERIR IMAGEM POLIGONO SEPARADO E POLIGONO EM COLISÃO

Isso pode ser verificado calculando a linha perpendicular à linha que separa as duas formas e projetarmos as formas nessa linha. Caso haja sobreposição das projeções os objetos estão em interseção, caso contrário não há colisão. Uma linha onde as projeções (sombras) das formas não se sobrepõem é chamada de eixo de separação.

O SAT deve testar todos os eixos candidatos a separação para determinar se há ou não sobreposição. Devido a isso ele não muito prático em ambientes 3D. No entanto, o teorema 1 nos garante que o primeiro eixo onde as projeções não estão sobrepostas, o algoritmo pode sair imediatamente determinando que as formas não estão se cruzando.

Algorithm 5: SAT 1

```
Input: objetoA e objetoB
foreach eixo do objetoA do
    p1 ← Projetar shapeA no eixo
    p2 ← Projetar shapeB no eixo
    if p1 sobrepoe p2 then
        Result: False
foreach eixo do objetoB do
    p1 ← Projetar shapeA no eixo
    p2 ← Projetar shapeB no eixo
    if p1 sobrepoe p2 then
        Result: False
Result: True
```

O algoritmo SAT precisa responder duas perguntas fundamentais:

1. **Quais eixos de separação testar?** deve-se testar a normal de todas faces (3D) ou arestas (2D) dos objetos que estão participando da colisão.

2. **Como projetar um geometria no eixo?** Para projetar um polígono em um eixo é relativamente simples; loop sobre todos os vértices e faça o produto interno com o eixo, armazenando o mínimo e máximo.

Dessa forma o algoritmo apenas retorna verdadeiro ou falso se as duas formas estão se sobrepondo. Além disso, o SAT pode retornar um vetor de translação mínima (MTV). A MTV é o vetor usado para empurrar as formas para fora da colisão, na direção de separação com tamanho da penetração.

Para o SAT a direção é equivalente ao eixo de separação e a penetração (magnitude do MTV) é o tamanho da menor projeção do polígono com o eixo de separação. Podemos adaptar nosso pseudocódigo como

Algorithm 6: SAT 2

Input: objetoA e objetoB
penetracao $\leftarrow$ 0
separacao $\leftarrow$ Vector.Zero
foreach *eixo do objetoA* **do**
    p1 $\leftarrow$ Projetar shapeA no eixo
    p2 $\leftarrow$ Projetar shapeB no eixo
    $\delta \leftarrow$ projeção p1 com p2
    if $\delta \leq TOLERANCIA$ **then**
        Result: False
    else
        penetracao $\leftarrow \delta$
        separacao $\leftarrow$ eixo
foreach *eixo do objetoB* **do**
    p1 $\leftarrow$ Projetar shapeA no eixo
    p2 $\leftarrow$ Projetar shapeB no eixo
    $\delta \leftarrow$ projeção p1 com p2
    if $\delta \leq TOLERANCIA$ **then**
        Result: False
    else
        penetracao $\leftarrow \delta$
        separacao $\leftarrow$ eixo
Result: penetracao, separacao

4.1.2 Algoritmo Gilbert–Johnson–Keerthi (GJK)

A GJK, como a SAT, só opera em formas convexas. GJK é um método iterativo, mas converge muito rápido. É uma alternativa melhor para o SAT para ambientes 3D devido ao número de eixos que o SAT deve testar.

O algoritmo GJK depende muito de um conceito chamado ***soma de Minkowski***. A soma de Minkowski de dois objetos convexos A e B é definida por:

$$A + B = \{\vec{x} + \vec{y} \mid \vec{x} \in A, \vec{y} \in B\}, \tag{4.1}$$

O objeto A + B é o conjunto de pontos obtido por um processo de varredura que translada o centro de massa de B para cada ponto de A, ou seja, faz-se uma cópia do objeto B centrado em cada ponto de A

(INSERIR FIGURA SOMA DE MINKOWSKI)

Uma propriedade muito útil da soma de Minkowski é o fato de que a soma de dois objetos convexos é um objeto convexo. O algoritmo GJK se beneficia dessas propriedades usando uma operação informalmente chamada de *diferença de Minkowski*

$$A - B = \{\vec{x} - \vec{y} \mid \vec{x} \in A, \vec{y} \in B\}, \tag{4.2}$$

Essa operação continua sendo a soma de Minkowski (a soma da diferença). Mas neste trabalho usaremos esse termo para referir a essa operação quando necessário.

A diferença de Minkowski pode ser pensado como um processo de varredura que calcula o vetor distancia para cada ponto de B em A. dessa forma, caso a distância entre dois pontos seja zero (interseção), podemos confirmar a colisão entre os dois objetos. Esse vetor também coincide com a origem, logo elaboramos nosso próximo teorema:

Teorema 2. *Se duas formas estiverem sobrepostas/em interseção a Diferença de Minkowski dessas formas conterá a origem.*

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

7.1 BROAD PHASE E NARROW PHASE

7.2 DIVISÃO ESPACIAL EM GRADE UNIFORME

7.2.1 Uso na Broad Phase

7.3 SEPARANDO SIMULAÇÃO DA THREAD PRINCIPAL

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.