Motor de jogos e Física

Etapas na simulação física de uma *game engine*

Slides por: Gustavo Ferreira Ceccon (gustavo.ceccon@usp.br)

Este material é uma criação do
Time de Ensino de Desenvolvimento de Jogos Eletrônicos (TEDJE)
Filiado ao grupo de cultura e extensão Fellowship of the Game (FoG), vinculado ao ICMC - USP

Objetivos

- Introduzir física e a matemática por trás dos jogos eletrônicos
- Mostrar o processo da simulação física
  - Simulação e métodos de integração
  - *Broadphase*, *Midphase e Narrowphase*
  - Colisão e organização dos dados
  - Algoritmos básicos de colisão
  - Algoritmo de Impulso Sequencial

Índice

1. Introdução

1. Introdução

- Física na vida real vs Física em jogos
  - Contínua vs Discreta
  - Contato vs Intersecção
- Como fazer tudo isso realisticamente e em tempo real?

1. Introdução

- ➔ Física 2D vs Física 3D
  - ◆ Colisores
  - ◆ Transformações
    - Posição
    - Rotação

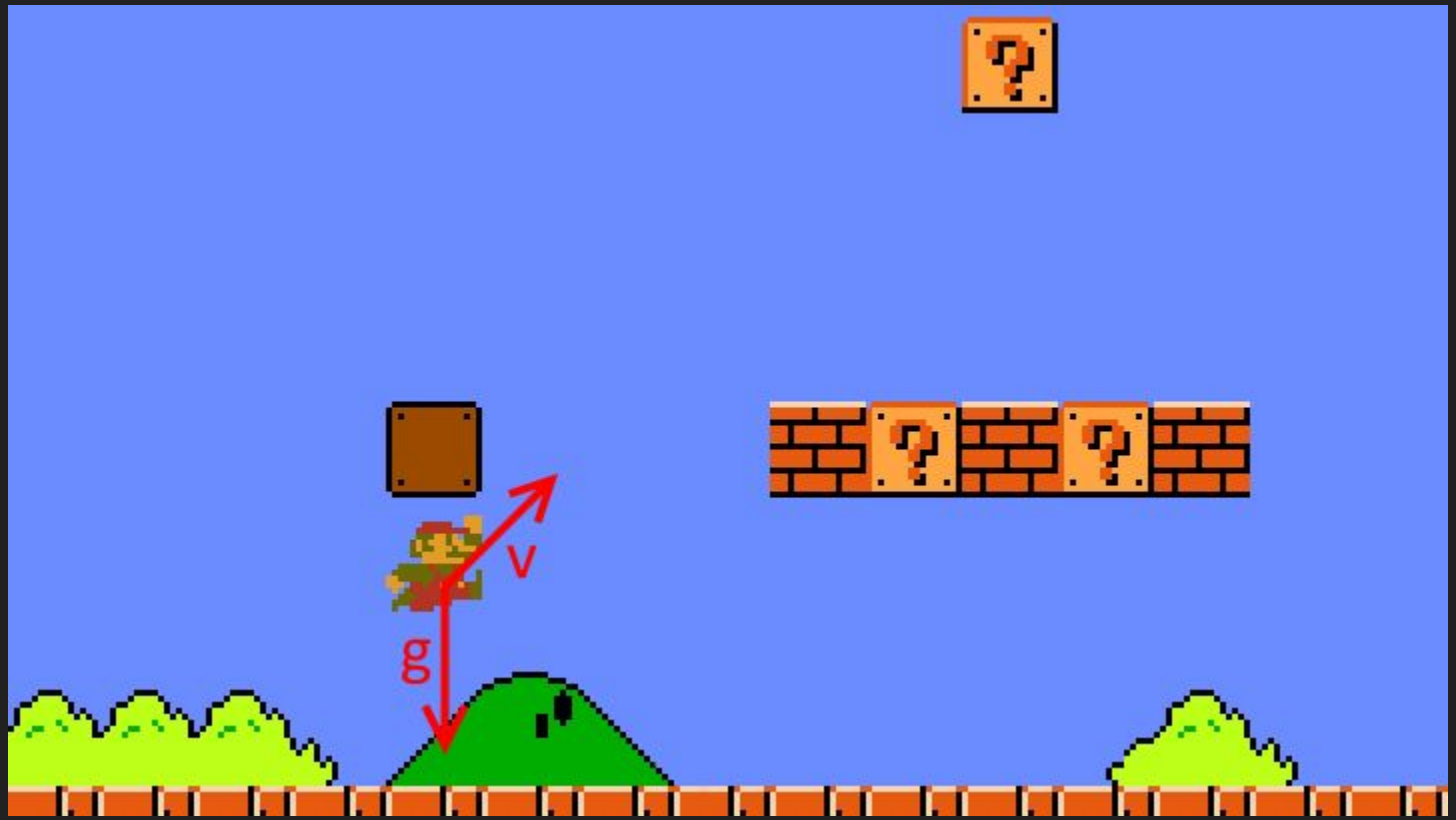
2. Etapas

2. Etapas

- Etapas do motor de física
  - Simulação
  - Detecção de Colisão
  - Resposta

2. Etapas

Simulação



Simulação

Detecção de Colisão

Resposta

Resposta

3. Simulação

3. Simulação

- ➔ Aplicar física newtoniana
- ➔ Aplicamos as forças para descobrir as acelerações
- ➔ Aplicamos aceleração para descobrir a velocidade
- ➔ Aplicamos a velocidade para descobrir a posição

3. Simulação

- ➔ Não conseguimos aplicar as equações conhecidas
  - ◆ Forças não são constantes
- ➔ É preciso usar integrais para fazer os cálculos
  - ◆ Cálculo numérico
  - ◆ Métodos de integração

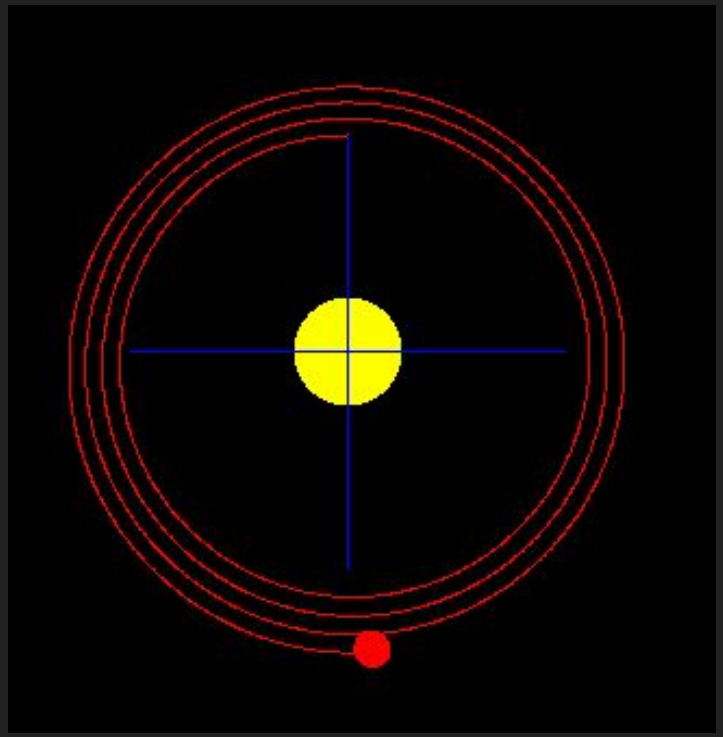
3. Simulação

- Tipos de objetos
  - Dinâmico com colisão: Mario
  - Dinâmico sem colisão: Moeda
  - Estático com colisão: Blocos
  - Estático sem colisão: Fundo

3. Simulação

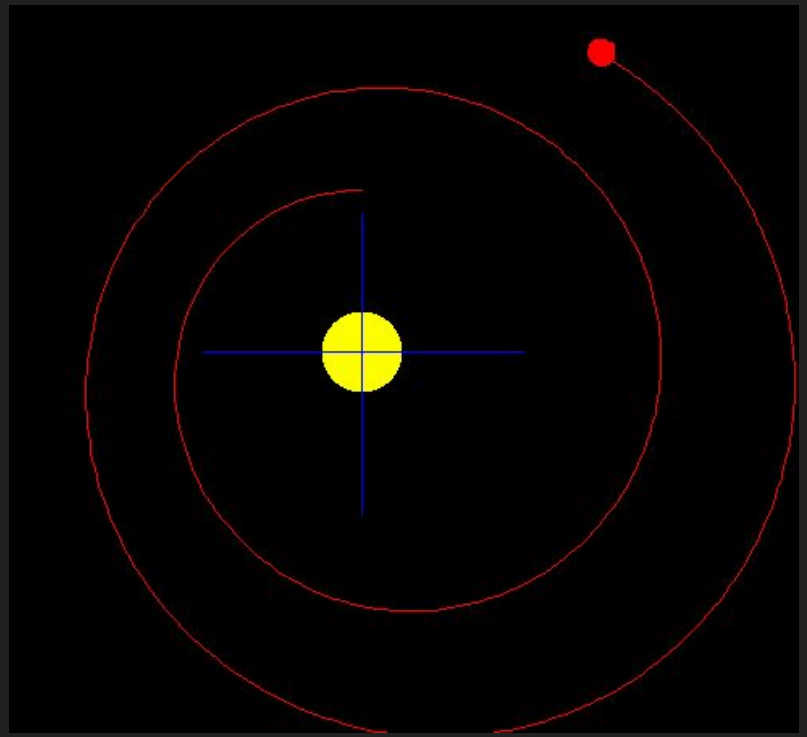
- Integração numérica
  - Diferentes métodos para diferentes tipos físicos
  - Euler e *Semi-Implicit* Euler
  - Verlet e *Time-Corrected* Verlet
  - Runge-Kutta 4 (ordem)

3. Simulação



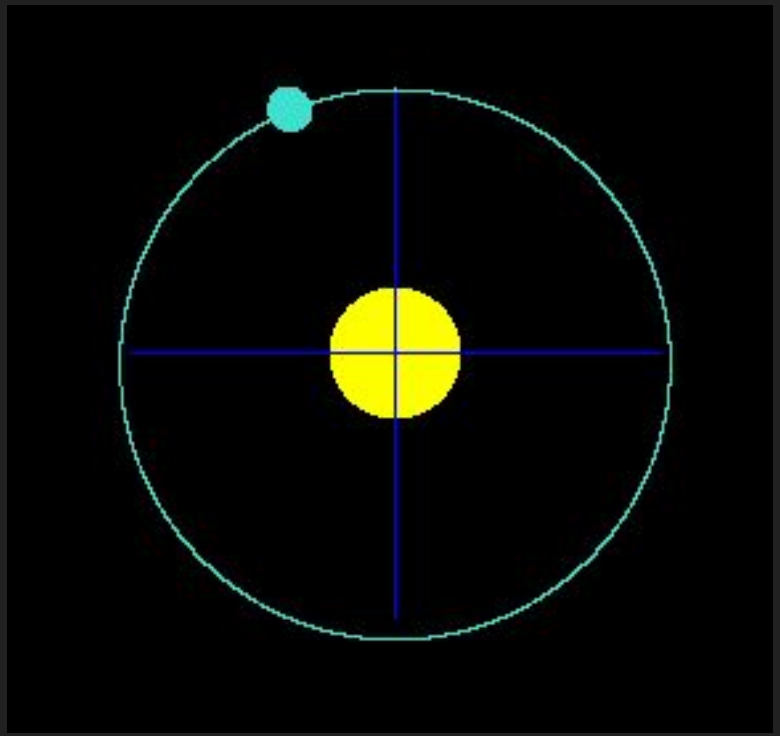
Euler 10 UPS

3. Simulação



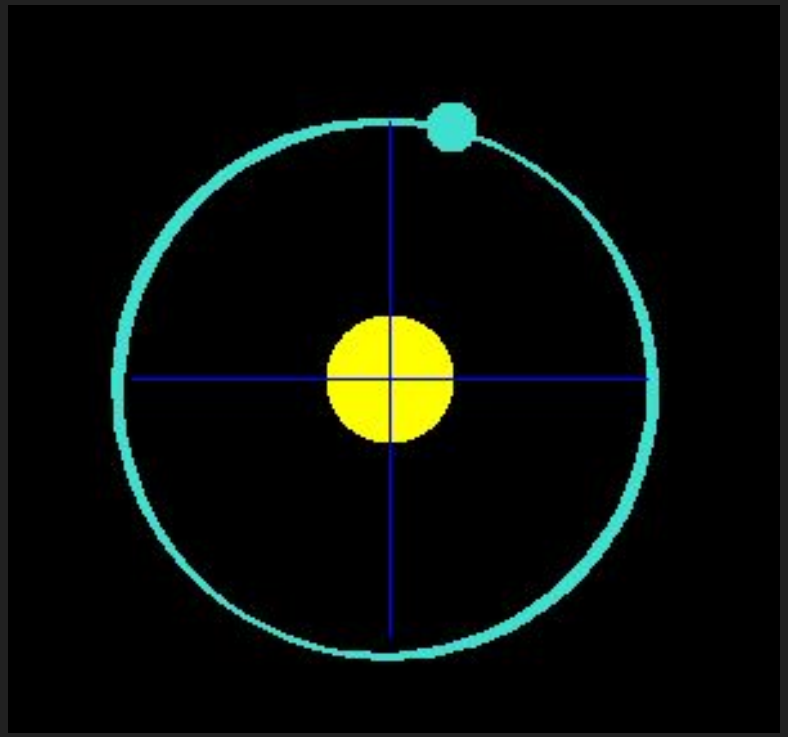
Euler 1 UPS

3. Simulação



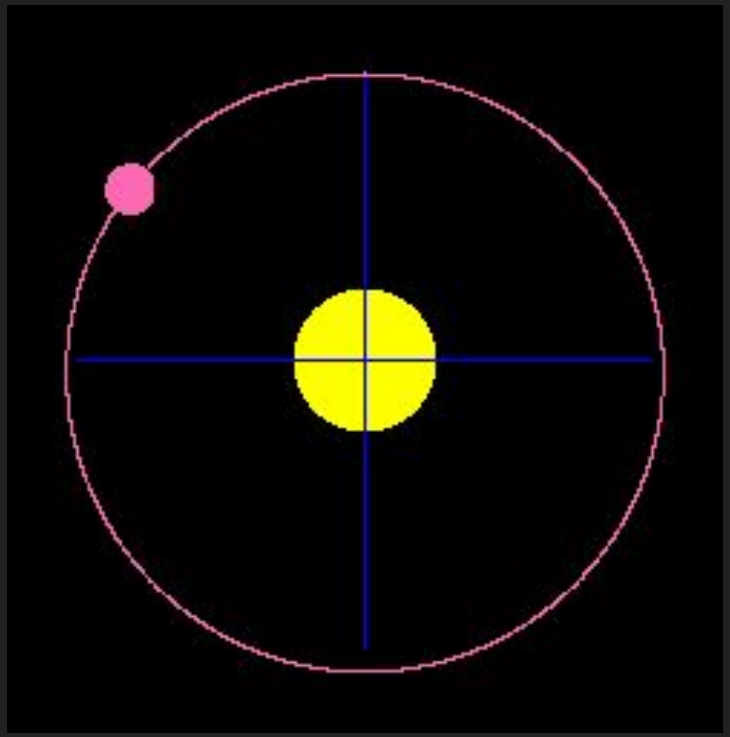
Verlet 10 UPS

3. Simulação



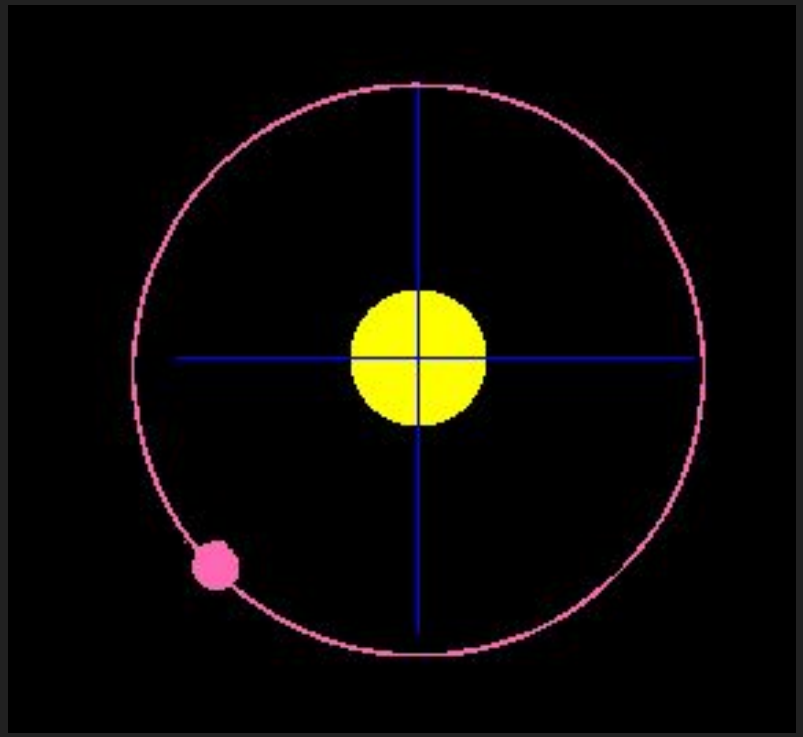
Verlet 1 UPS

3. Simulação



RK4 10 UPS

3. Simulação



RK4 1 UPS

3. Simulação

- Problemas possíveis
  - Forças variáveis
  - dt variável
  - Causam inconsistência na física
  - Quanto maior a ordem do método, melhor a aproximação

4. Colisão

4. Colisão

- *Broad-phase*
  - Separar objetos possíveis de colidir
- *Mid-phase*
  - Testar colisão rápida entre objetos
- *Narrow-phase*
  - Testar colisão precisa entre objetos

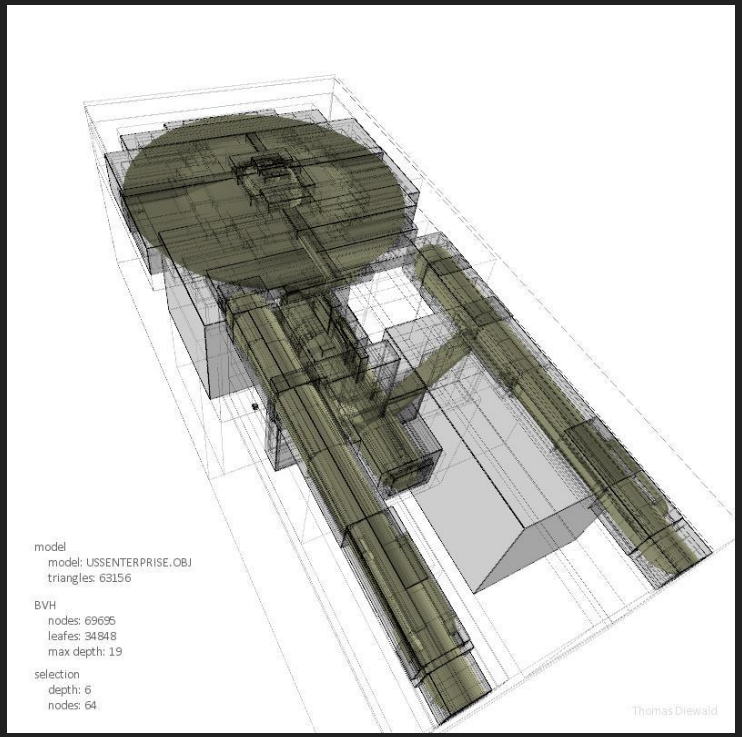
4. Colisão

- ➔ *Broad-phase*
  - ◆ BVH - *Bounding Volume Hierarchy*
  - ◆ *Spatial Partitioning*
    - ● *Quadtree* e *Octree*
    - ● *k-d tree*
    - ● BSP - *Binary Space-Partitioning Tree*

4. Colisão

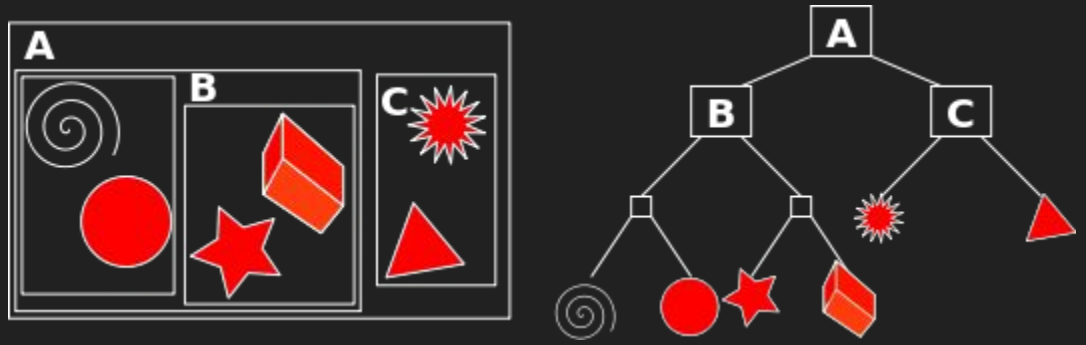
- BVH
  - Árvore de colisores de *bounding volumes* (como uma caixa ou uma esfera)
  - Nós são colisores aproximados de todos os colisores filhos

4. Colisão



Bounding Volume Hierarchy

4. Colisão



Bounding Volume Hierarchy

4. Colisão

- BVH
  - Estática: usada geralmente em um objeto para subdividir os colisores
  - Dinâmica: usada em múltiplos objetos (geralmente dinâmicos), calculada em tempo de execução

4. Colisão

- ➔ *Spatial Partitioning*
  - Dividir o espaço em partes onde pode haver colisão
  - Objetos dentro do mesmo espaço são checados posteriormente

4. Colisão

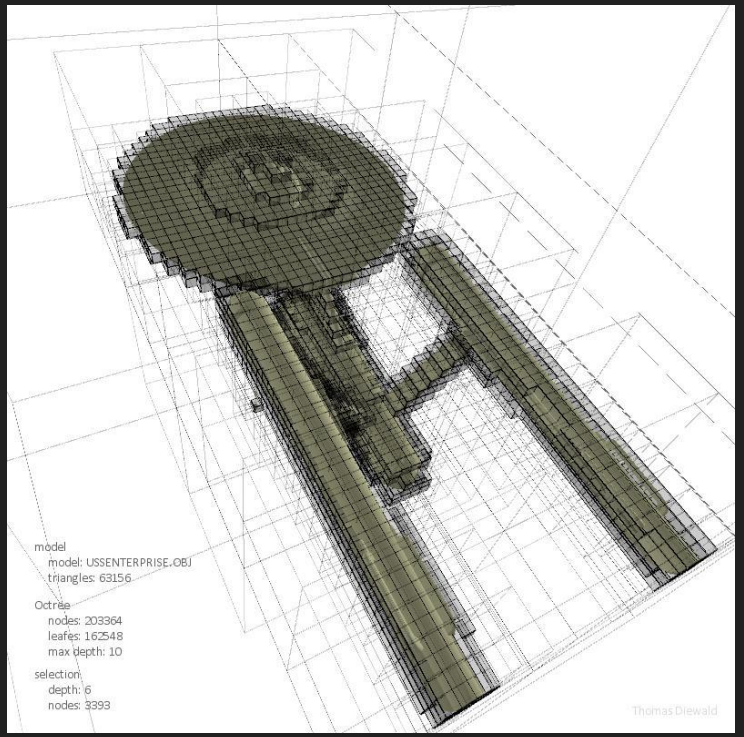
- ➔ *Quadtree e Octree*
  - São relativamente rápidas de calcular
  - Diminuem consideravelmente o número de comparações necessárias

4. Colisão

Quadtree

Quadtree 2

4. Colisão



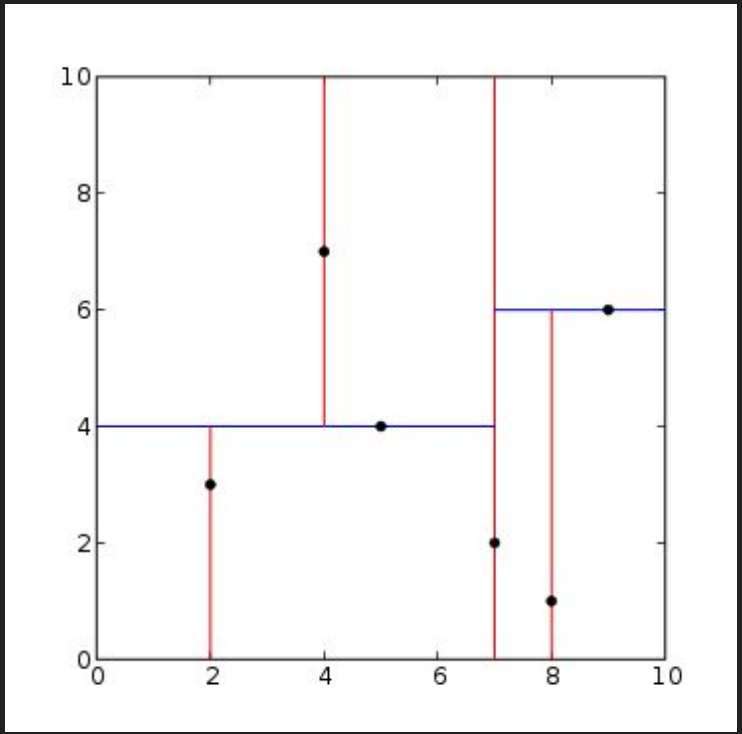
Octree

4. Colisão

➔ *k-d tree*

- Generalização da *quadtree* e da *octree*
- Dividir o espaço em 2 (binariamente) de forma arbitrária
- Divisão arbitrária torna mais custoso

4. Colisão

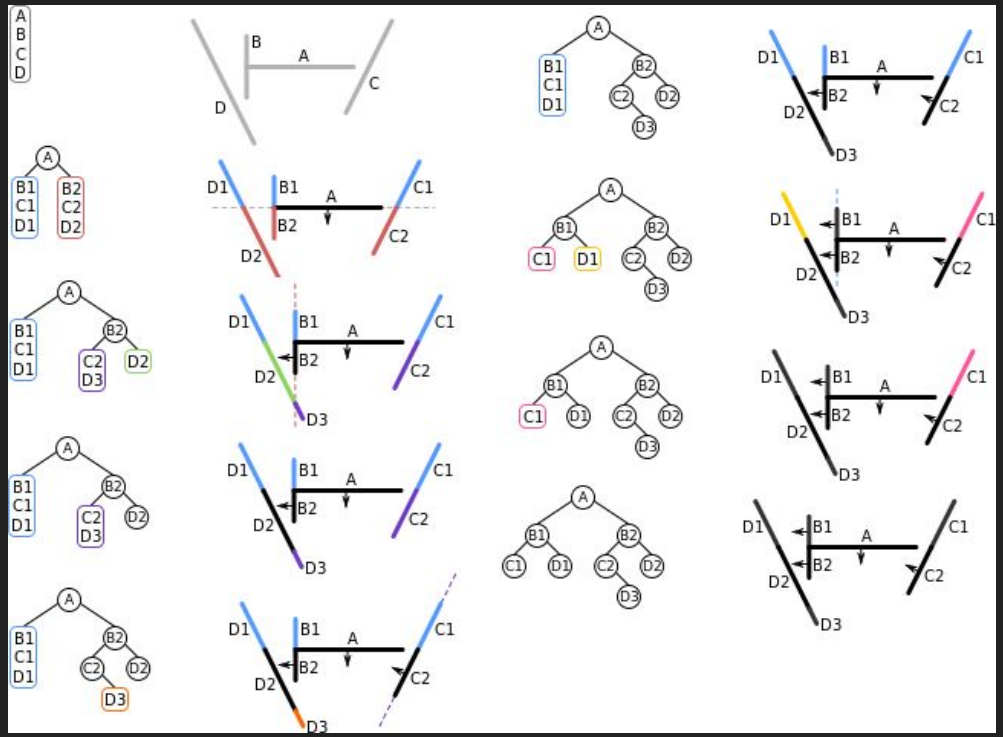


k-d tree

4. Colisão

- BSP *tree*
  - Generalização ainda maior da *k-d tree*
  - Muito pesada de calcular, porém muito eficaz
  - Usada em objetos estáticos, como um cenário urbano
  - Usada em renderização (ordenação dos objetos), ao invés do Z-buffer

4. Colisão



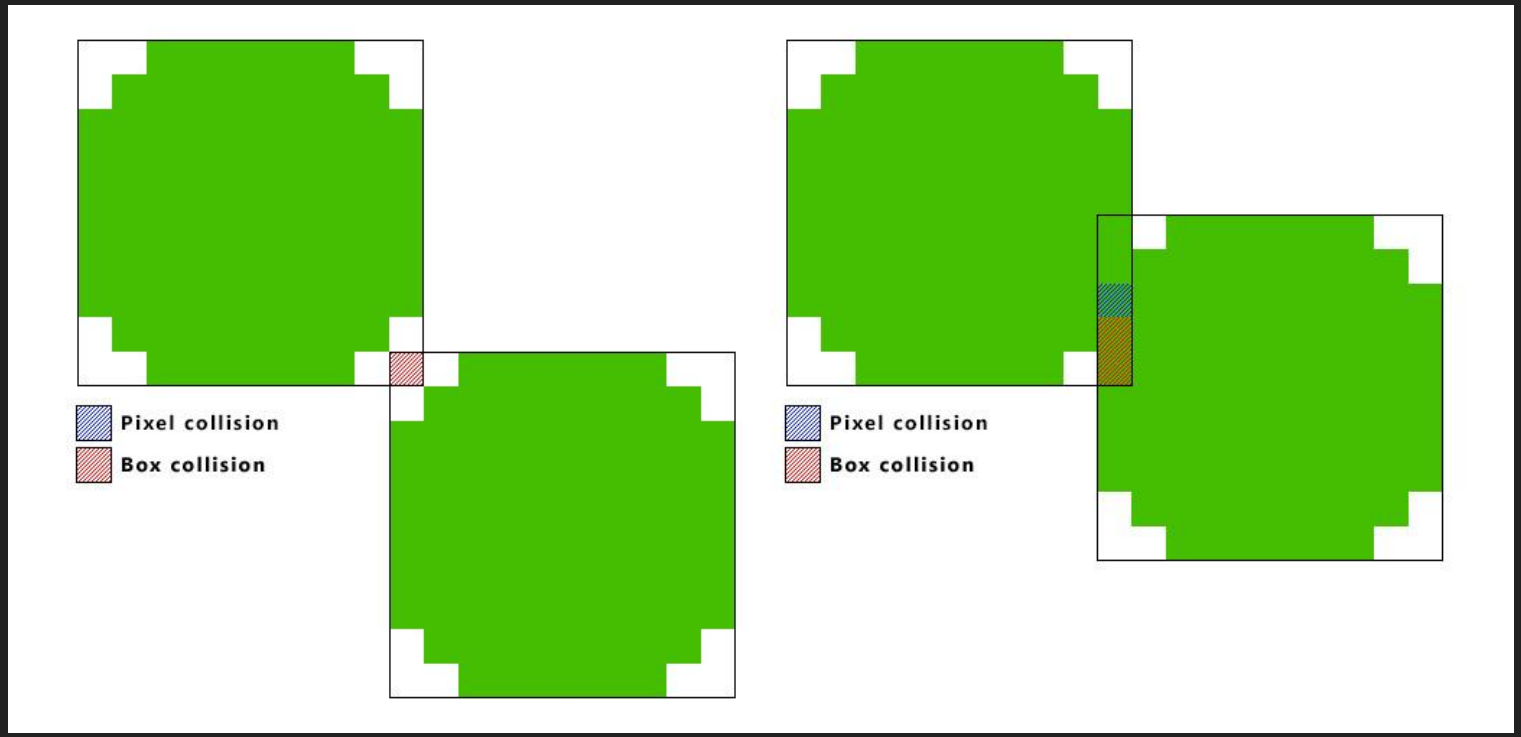
BSP *tree*

5. Tipos de Colisão

5. Tipos de Colisão

- Objeto 2D ou 3D
- *Pixel-level*
- Formas primitivas
- Formas côncavas e convexas
- *Raycast*
- SAT - *Separating Axis Theorem*
- GJK - Gilbert–Johnson–Keerthi

5. Tipos de Colisão



Colisão *pixel-level*

5. Tipos de Colisão

- Formas primitivas
  - Esfera-Esfera
  - AABB - *Axis Aligned Bounding Box*
  - AABB-AABB
  - AABB-Esfera

5. Tipos de Colisão

Exemplo

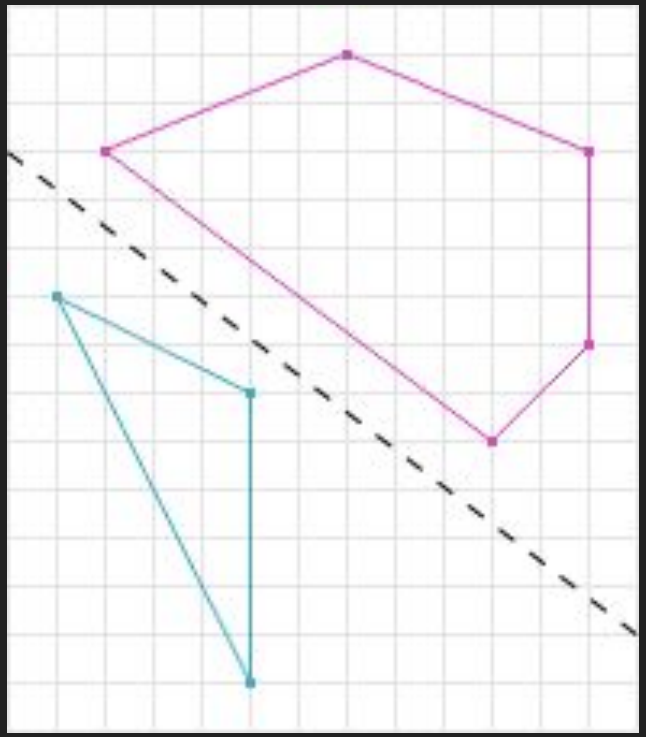
5. Tipos de Colisão

- Formas côncavas
  - Dividir em formas convexas
  - Não funciona com os algoritmos usados
  - Côncavo para convexo
    - Decomposição e triangulização

5. Tipos de Colisão

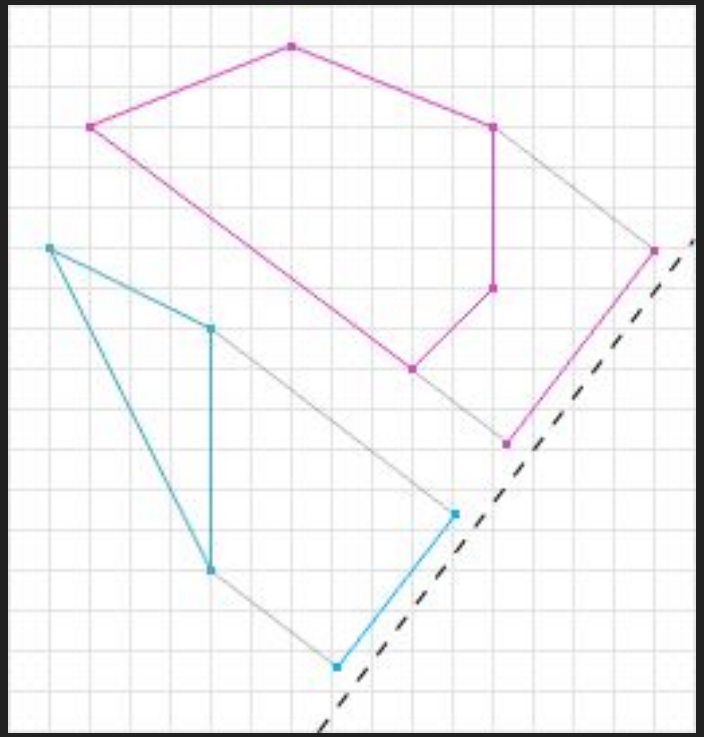
- ➔ *Raycast*
  - Raio de colisão
  - Útil em aproximação de objetos rápidos
    - Projétil
  - Muito pesado normalmente
    - *Broadphase* é necessário para reduzir o tempo

5. Tipos de Colisão



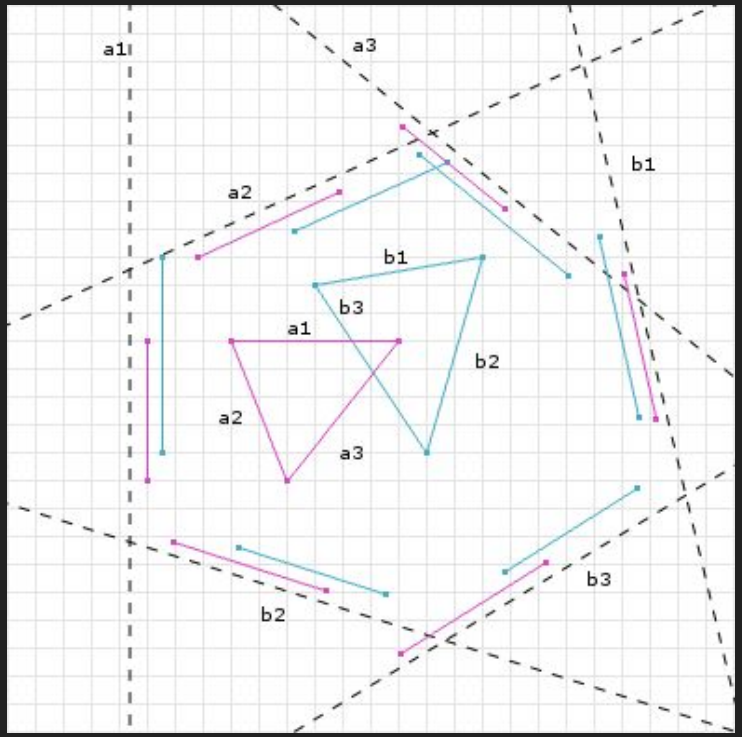
SAT - *Separating Axis Theorem*

5. Tipos de Colisão



SAT - *Separating Axis Theorem*

5. Tipos de Colisão

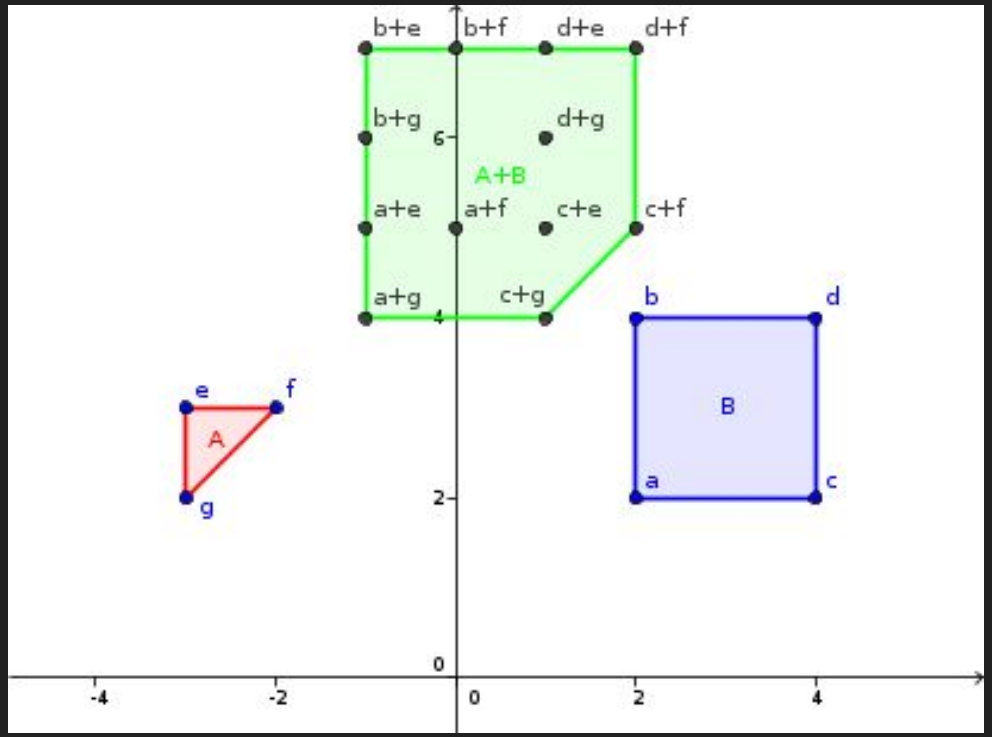


SAT - *Separating Axis Theorem*

5. Tipos de Colisão

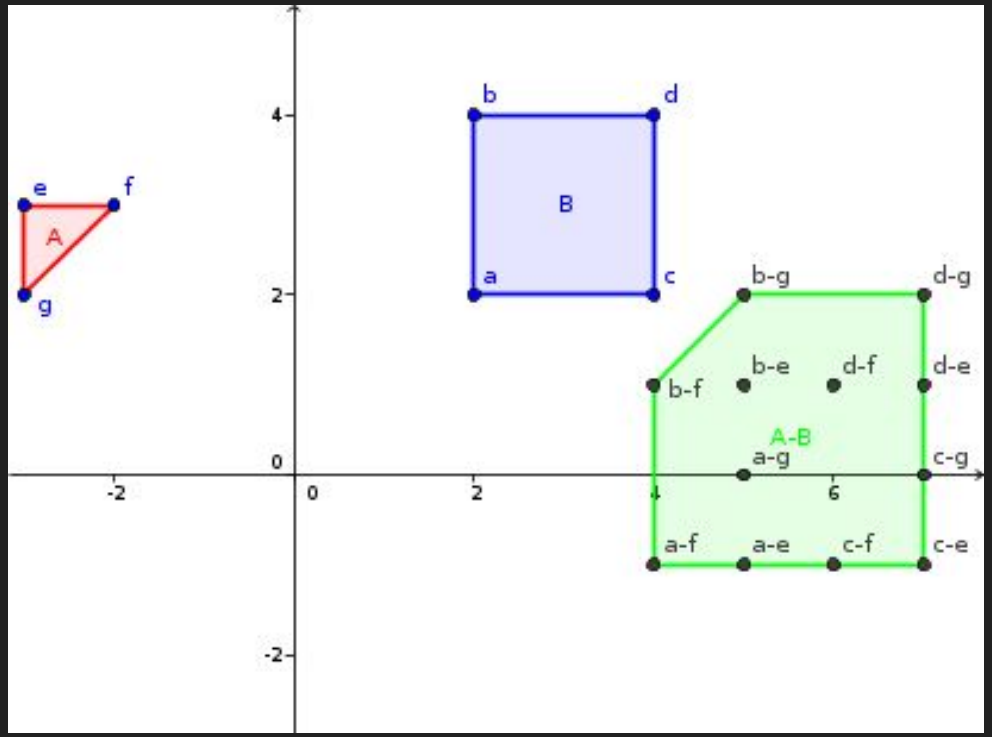
- SAT é fácil de implementar, porém é muito devagar para polígonos complexos
- GJK é rápido na maior parte dos casos
- GJK pode ser reaproveitado para descobrir mais informações
  - Distância e pontos mais próximos
  - Informações de colisão: EPA

5. Tipos de Colisão



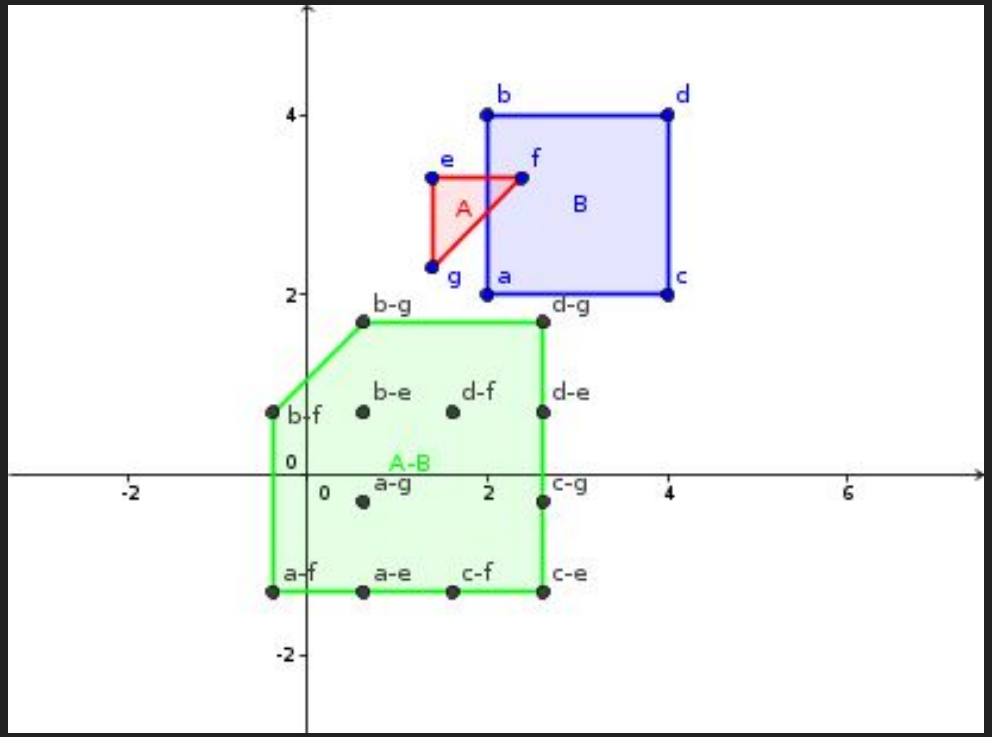
Soma de Minkowski

5. Tipos de Colisão



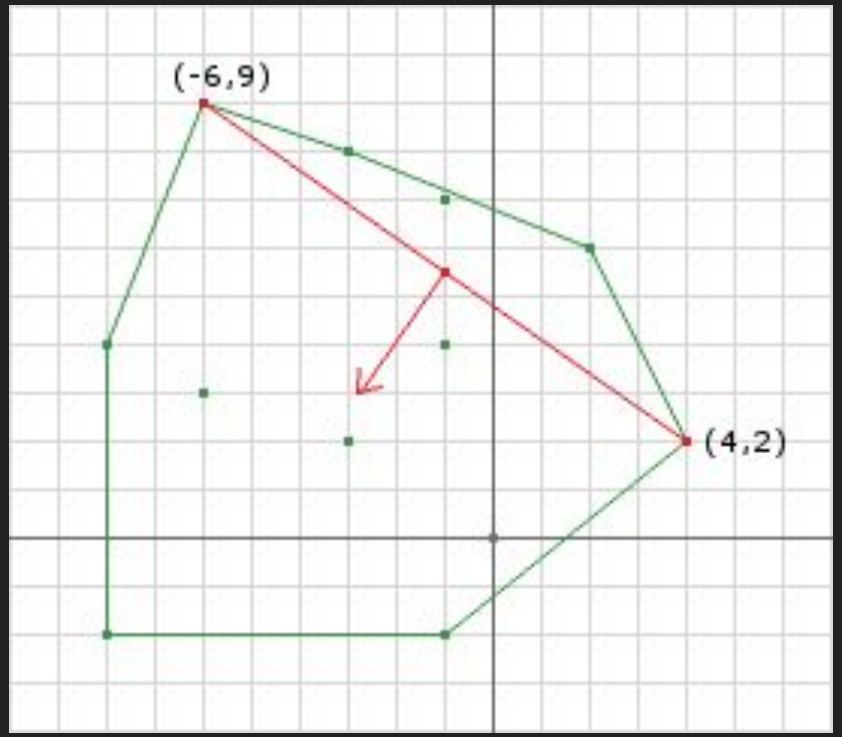
Subtração de Minkowski

5. Tipos de Colisão



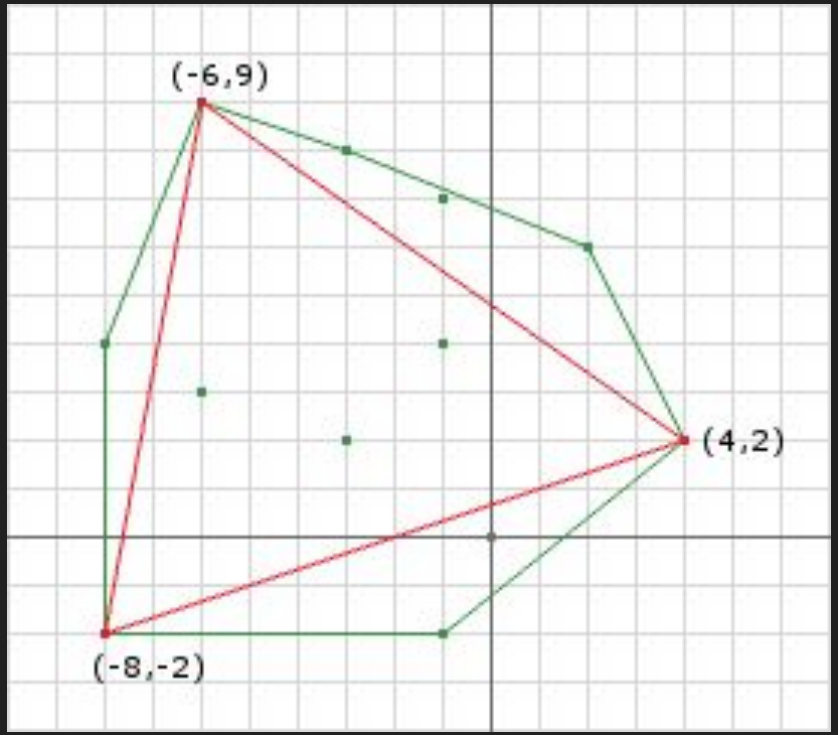
Subtração de Minkowski

5. Tipos de Colisão



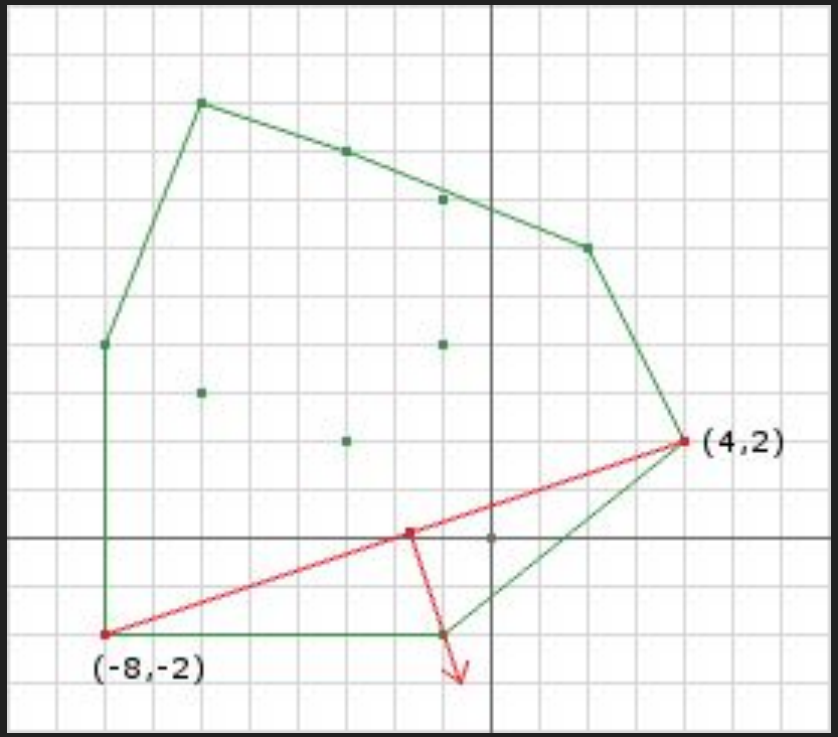
GJK - Gilbert–Johnson–Keerthi

5. Tipos de Colisão



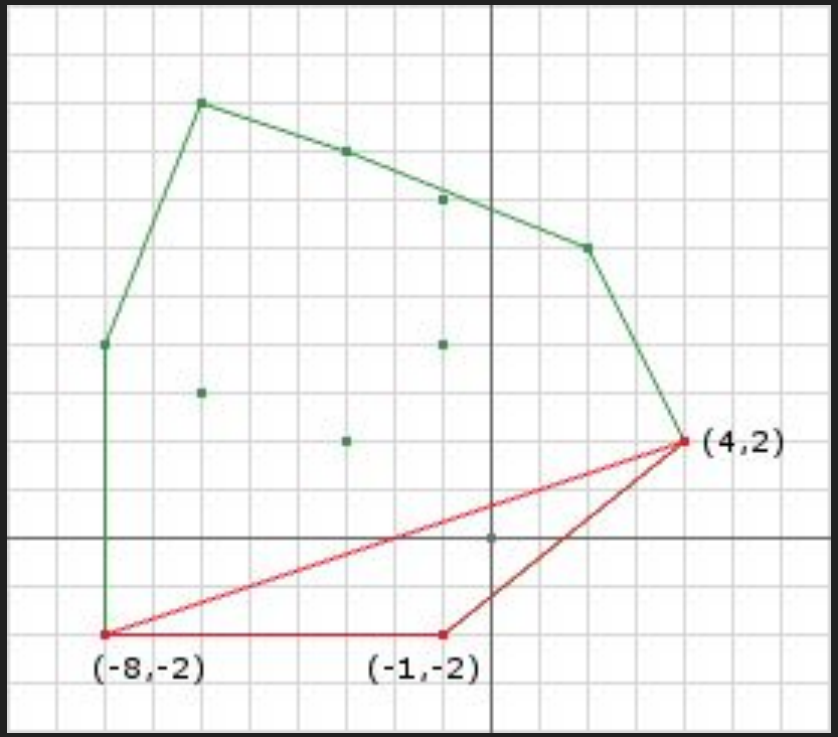
GJK - Gilbert–Johnson–Keerthi

5. Tipos de Colisão



GJK - Gilbert–Johnson–Keerthi

5. Tipos de Colisão



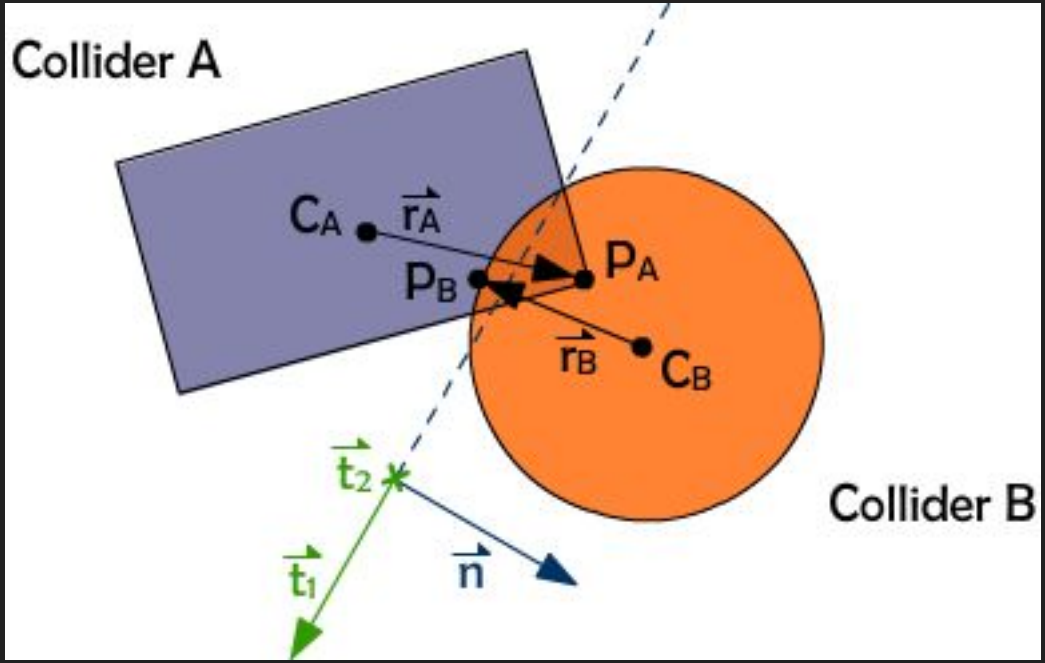
GJK - Gilbert–Johnson–Keerthi

6. Resposta

6. Resposta

- Informação da colisão
  - Normal
  - Ponto de Contato
  - Profundidade
- EPA - *Expanding Polytope Algorithm*

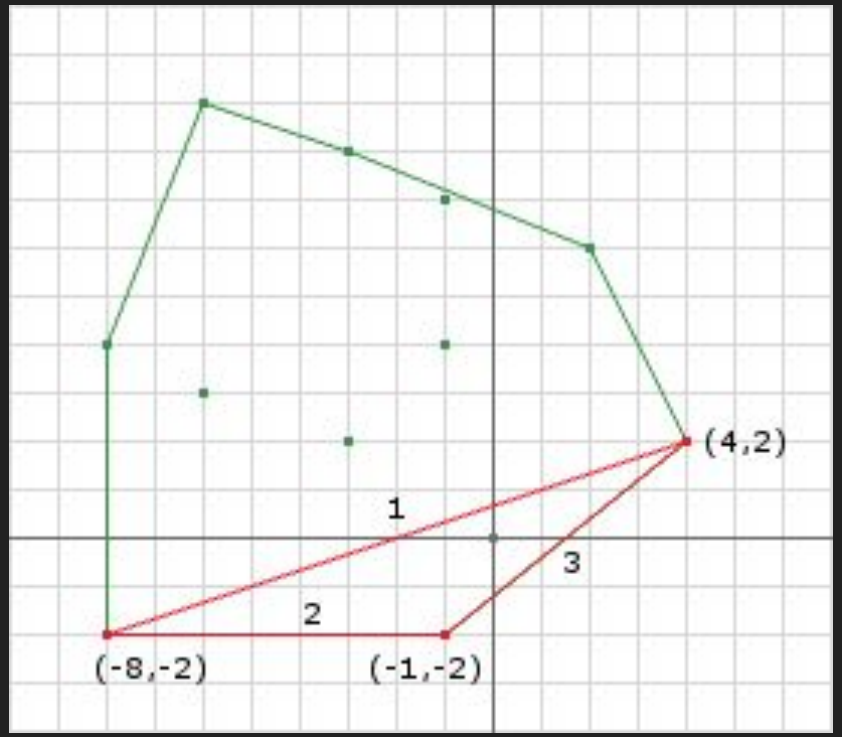
6. Resposta


Collider A
C_A
$\vec{r_A}$
P_B
P_A
$\vec{r_B}$
C_B
$\vec{t_2}$
Collider B
$\vec{t_1}$
$\vec{n}$

6. Resposta

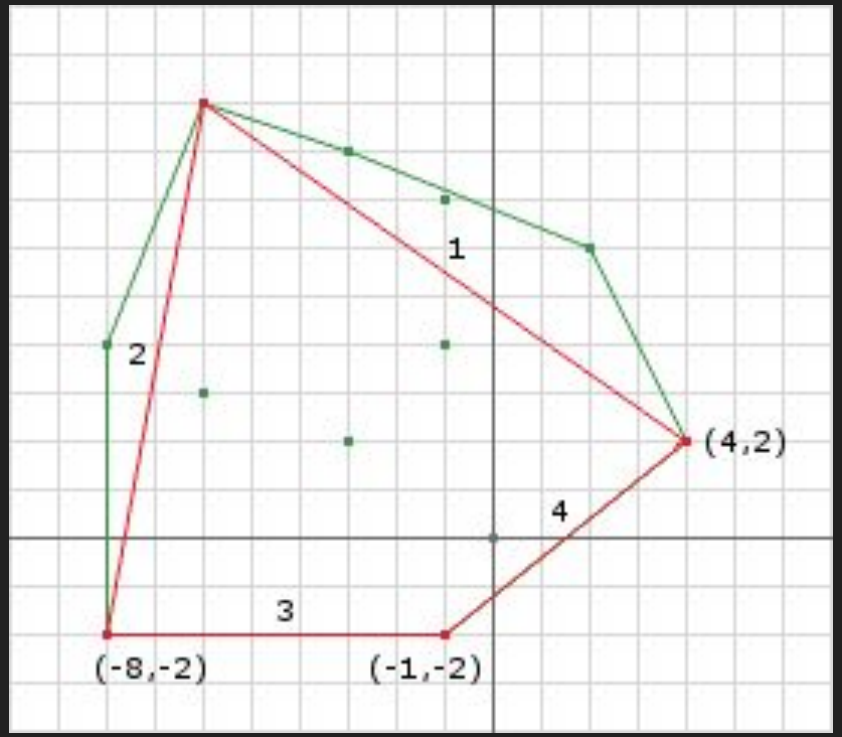
- EPA - *Expanding Polytope Algorithm*
  - Pega o *simplex* final do GJK e expande
  - A aresta ou plano mais próximo do polígono diz:
    - Profundidade da intersecção
    - Normal da intersecção

5. Tipos de Colisão



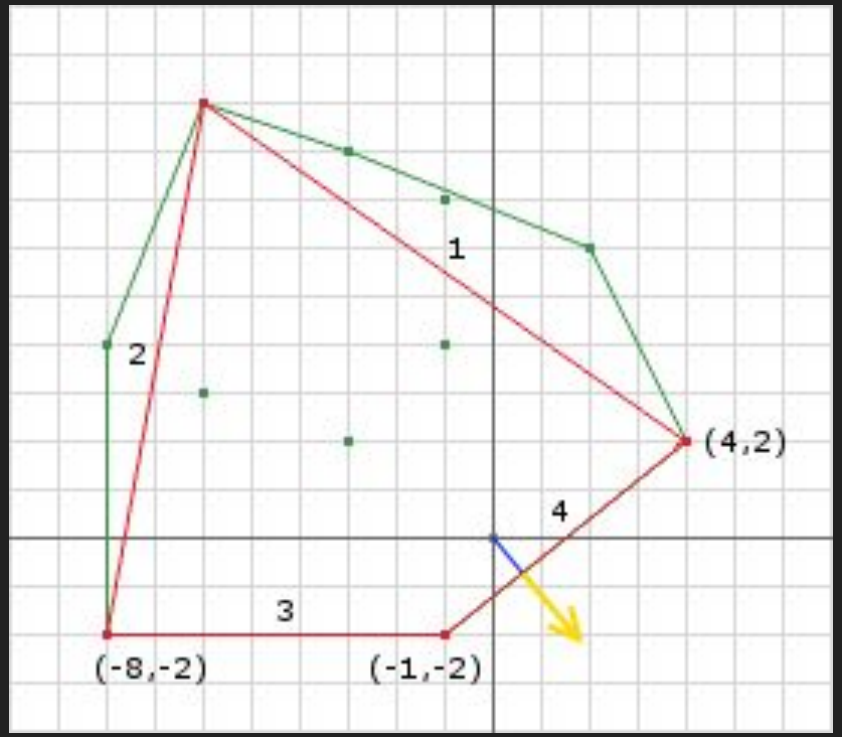
EPA - *Expanding Polytope Algorithm*

5. Tipos de Colisão



EPA - *Expanding Polytope Algorithm*

5. Tipos de Colisão



EPA - *Expanding Polytope Algorithm*

6. Resposta

- ➔ *Sequential Impulse Solver*
  - *Constraints* - Restrições
  - Impulso
  - Encontra nova velocidade
  - Resolve a sobreposição
  - Atrito

6. Resposta

Exemplo

7. Colisão Contínua

7. Colisão Contínua

- ➔ Objetos muito rápidos podem quebrar a física: projéteis
  - ◆ Detectando colisão
    - ● Múltiplas iterações
    - ● Estender o colisor
  - ◆ Determinando tempo de impacto
    - ● “Voltar no tempo”

Dúvidas?

Referências

Referências

[1]http://www.dyn4j.org/
[2]http://allenchou.net/game-physics-series/
[3]http://codeflow.org/entries/2010/aug/28/integration-by-example-euler-vs-verlet-vs-runge-kutta/
[4]https://benjaminhorn.io/code/pixel-accurate-collision-detection-with-javascript-and-canvas/
[5]www.jeffreythompson.org/collision-detection/table_of_contents.php
[6]https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects
[7]https://research.ncl.ac.uk/game/mastersdegree/gametechnologies/physics8collisionmanifolds/Tutorial%208%20-%20Collision%20Manifolds.pdf
[8]http://judis.me/wordpress/sequential-impulse-solver/
[9]http://box2d.org/
[10]http://thomasdiewald.com/blog/?p=1488
[11]https://en.wikipedia.org/wiki/Binary_space_partitioning
[12]http://www.gdcvault.com/play/1020583/Animation-Bootcamp-An-Indie-Approach
[13]https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects
[14] (The Morgan Kaufmann Series in Interactive 3-D Technology) Christer Ericson-Real-Time Collision Detection-Morgan Kaufmann (2005)