

Motor de jogos e Física

Simulação e colisão física

Slides por:

Gustavo Ferreira Ceccon (gustavo.ceccon@usp.br)

Adaptados por:

Fabrício Guedes Faria (fabricao.guedes.faria@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

Este material possui licença CC By-NC-SA. Mais informações em:
<https://creativecommons.org/licenses/by-nc-sa/4.0/>



Objetivos

- Introduzir física e a matemática de um motor de jogos
- Mostrar o processo da simulação física
 - ◆ Simulação e métodos de integração
 - ◆ Broadphase, Midphase e Narrowphase
 - ◆ Colisão e organização dos dados
- Não serão abordados todos algoritmos nessa aula



Índice

1. Introdução
2. Etapas
3. Simulação
4. Colisores
5. Colisão
6. Resposta



1. Introdução



1. Introdução

→ Física no mundo real

- ◆ Contínua, não há saltos de tempo
- ◆ Contato, os objetos se tocam e não intersectam

→ Física no mundo virtual

- ◆ Discreta, a cada frame passa um Δt
- ◆ Intersecção, os objetos podem ocupar mesmo espaço



1. Introdução

→ Física 2D

- ◆ Colisores: quadrados, círculos
- ◆ Transformações
 - \mathbb{R}^2
 - Rotação com um grau de liberdade

→ Física 3D

- ◆ Colisores: cubos, esferas
- ◆ Transformações
 - \mathbb{R}^3
 - Rotação com três graus de liberdade



1. Introdução

→ Tipos de objetos

- ◆ Dinâmico com colisão: Mario
- ◆ Dinâmico sem colisão: Moeda
- ◆ Estático com colisão: Blocos
- ◆ Estático sem colisão: Fundo

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, que é suficiente para velocidades baixas
- ◆ Aplicamos as forças para descobrir as acelerações
- ◆ Aplicamos aceleração para descobrir a velocidade
- ◆ Aplicamos a velocidade para descobrir a posição

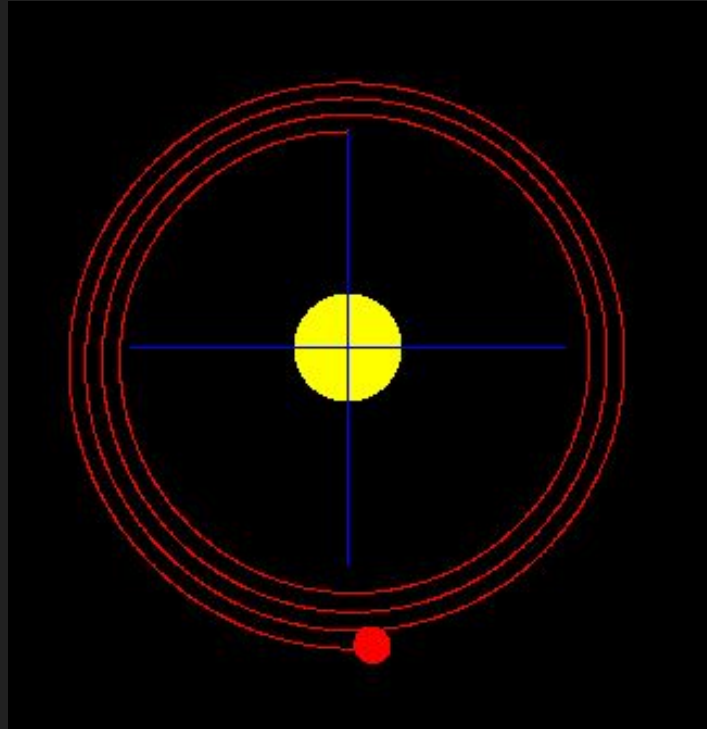
3. Simulação

- Não conseguimos aplicar as equações de ensino médio, porém elas são parecidas
 - ◆ Forças não são constantes, mundo discreto
- É preciso usar integrais para fazer os cálculos
 - ◆ Cálculo numérico aprendemos alguns métodos de integração, que são bem parecidos com esses

3. Simulação

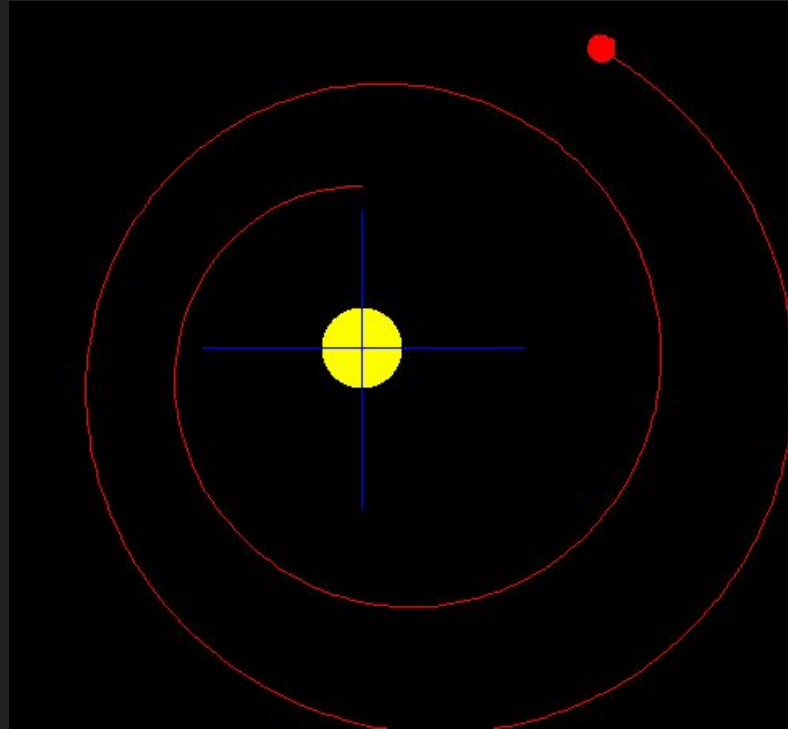
- Integração numérica é aproximar um valor de uma integral usando métodos computacionais (passos)
- ◆ Diferentes métodos para diferentes tipos físicos
- ◆ Euler e *Semi-Implicit* Euler (parecidos com ensino médio)
- ◆ Verlet e *Time-Corrected* Verlet (pulam uma etapa)
- ◆ Runge-Kutta 4ª ordem (mais acurado)

3. Simulação



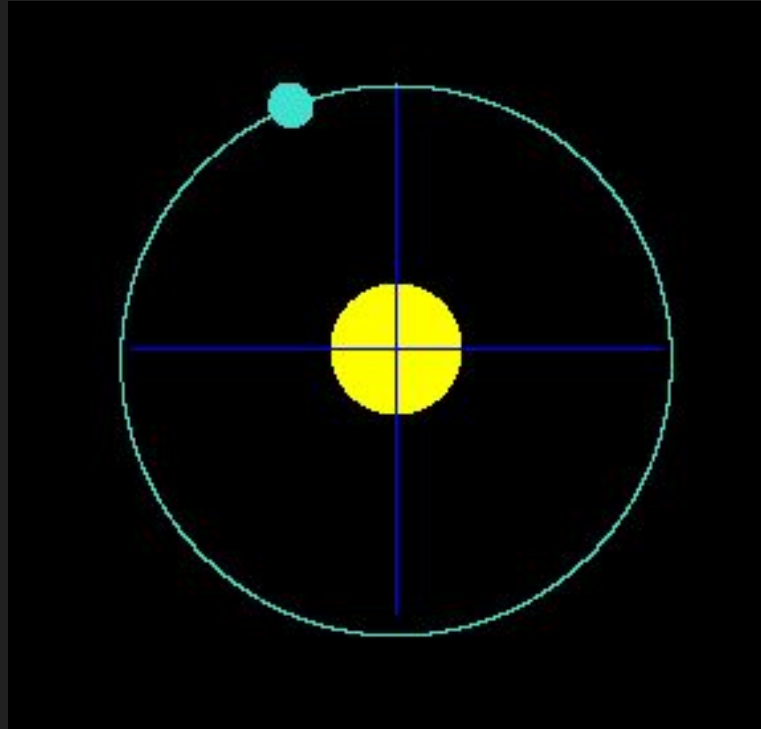
Euler 10 UPS

3. Simulação



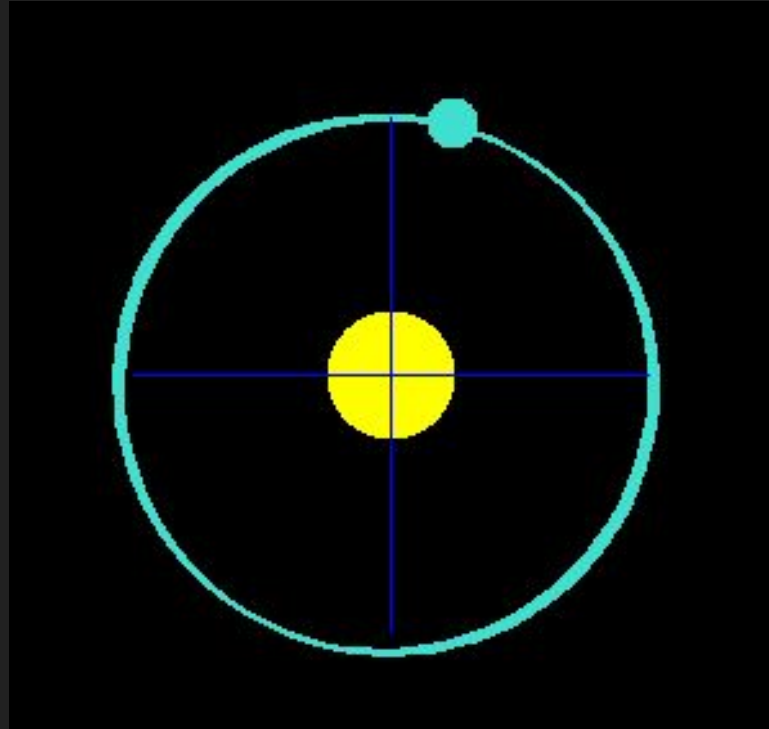
Euler 1 UPS

3. Simulação



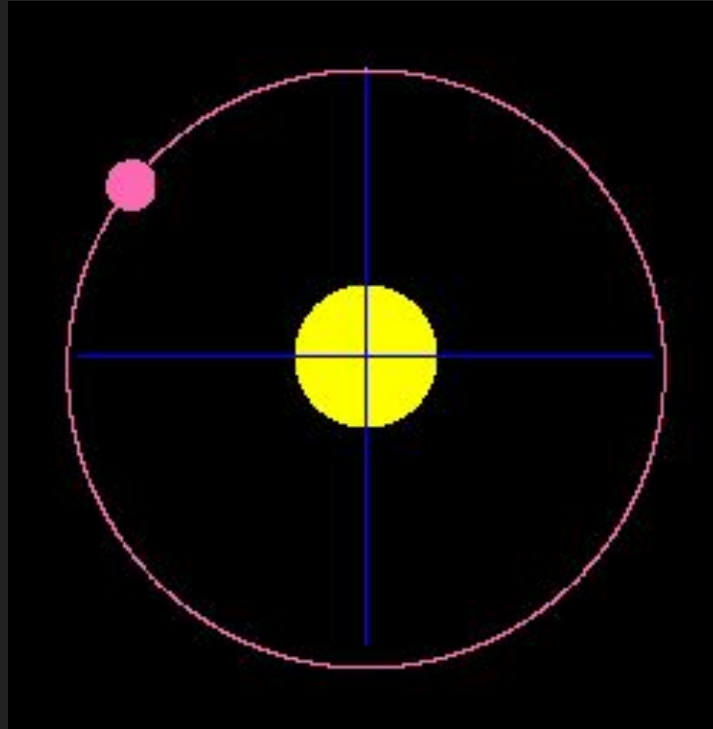
Verlet 10 UPS

3. Simulação



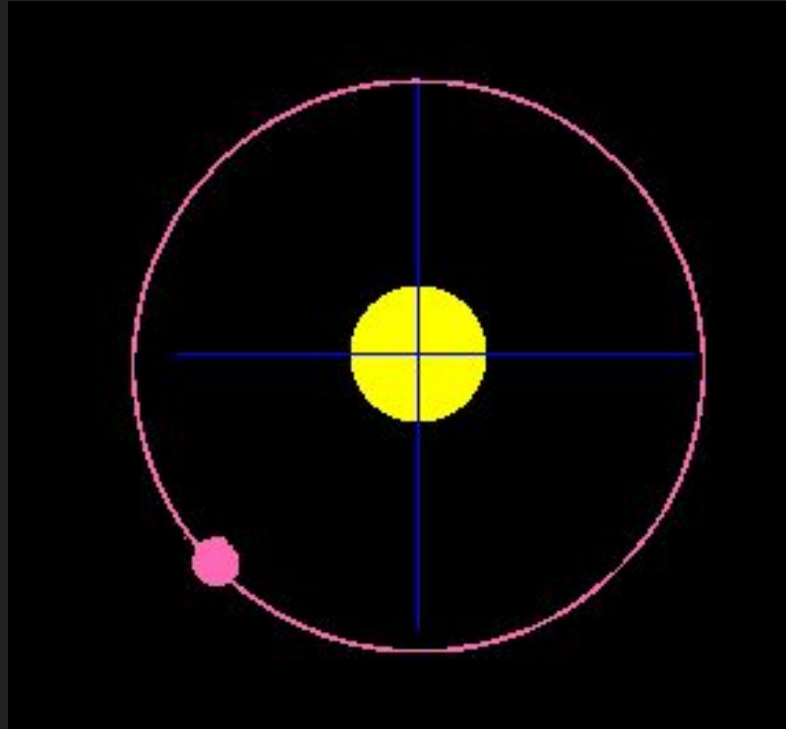
Verlet 1 UPS

3. Simulação



RK4 10 UPS

3. Simulação

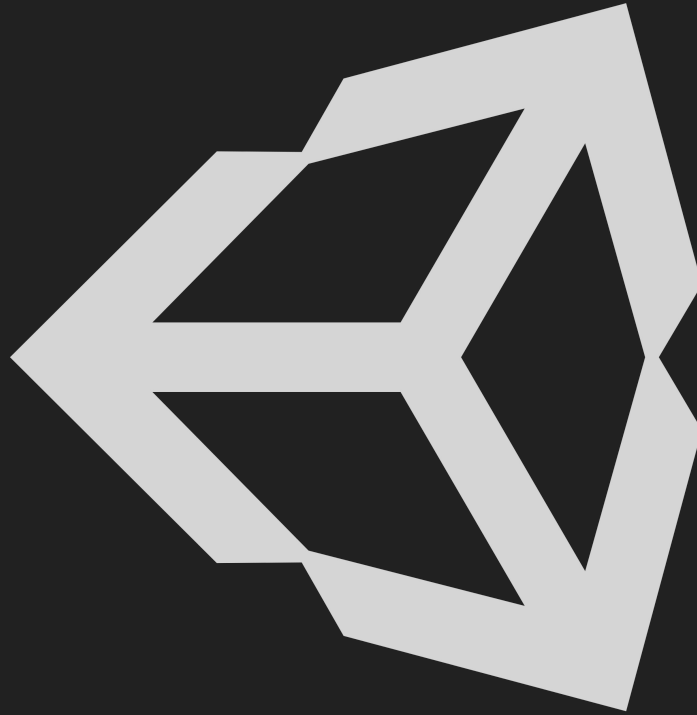


RK4 1 UPS

3. Simulação

- Problemas possíveis
 - ◆ Forças variáveis
 - ◆ dt variável
- Problemas causam inconsistência na física
- Quanto maior a ordem do método, melhor a aproximação, porém maior custo computacional

UNITY TIME !!!! - Cubo/Rigidbody



4. Colisores

4. Colisores

- Sprite 2D - pixel-level
- Malha 3D - triângulos
- Formas primitivas
 - ◆ AABB - Axis Aligned Bounding Box
- Formas côncavas e convexas
- Raycast

4. Colisores

Exemplo



4. Colisores

- Formas côncavas
 - ◆ Não funciona com os algoritmos usados
- Dividir em formas convexas
 - ◆ Côncavo para convexo
 - ◆ Decomposição e triangulação
 - ◆ Convex hull



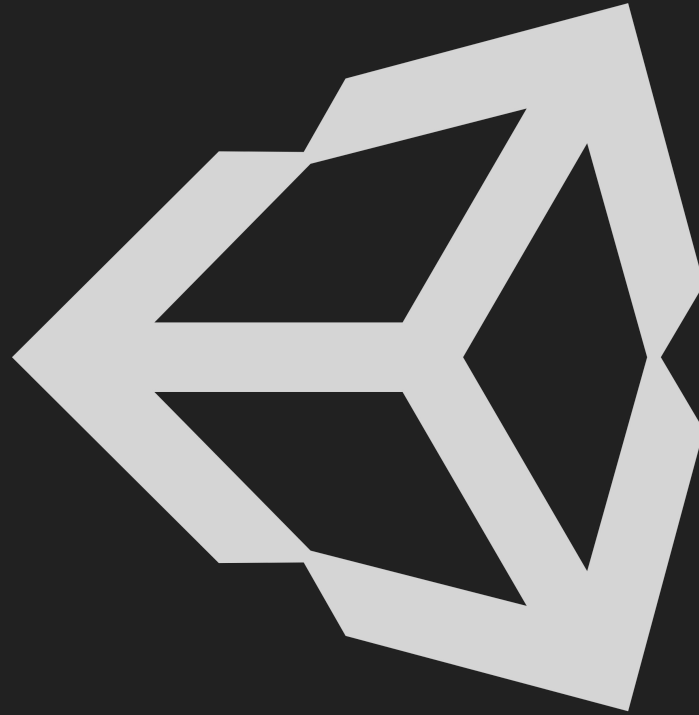
4. Colisores

→ Raycast

- ◆ Raio de colisão
- ◆ Útil em aproximação de objetos rápidos
 - Projétil
- ◆ Muito pesado normalmente
- ◆ Pode testar até o primeiro alvo ou todos os objetos



UNITY TIME !!!! - Colisores/Matrix



5. Colisão

5. Colisão

- Precisamos testar a colisão de n objetos
 - ◆ Força bruta demoraria $O(n^2)$ para testar a colisão entre si todos os objetos
 - ◆ Precisamos diminuir o número de testes, além disso precisamos coletar as informações das colisões

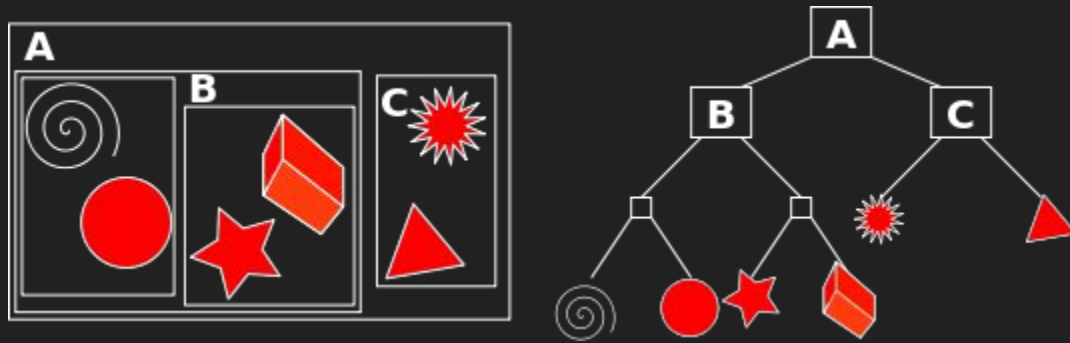
5. Colisão

- Dividimos em três etapas a colisão
- Broad-phase
 - ◆ Separação brusca dos objetos impossíveis de colidir
- Mid-phase
 - ◆ Teste rápido de colisão entre objetos, usando colisores simples
- Narrow-phase
 - ◆ Teste preciso de colisão entre objetos, usando diferentes tipos de colisores (compostos) e algoritmos de colisão

5. Colisão

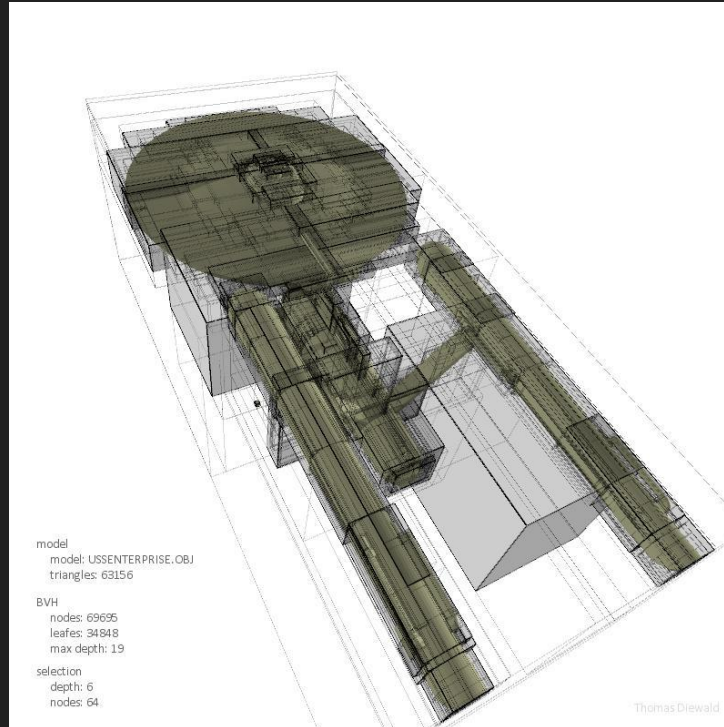
- BVH - Bounding Volume Hierarchy
 - ◆ Árvore de colisores de bounding volumes (como uma caixa ou uma esfera)
 - ◆ Nós são colisores aproximados de todos os colisores filhos
 - ◆ Quanto mais alta hierarquia, pior a aproximação
 - ◆ Usados tanto para teste quanto para representação

5. Colisão



Bounding Volume Hierarchy

5. Colisão



Bounding Volume Hierarchy

5. Colisão

- BVH - Bounding Volume Hierarchy
 - ◆ Estática: usada geralmente em um objeto para subdividir os colisores
 - ◆ Dinâmica: usada em múltiplos objetos (geralmente dinâmicos), calculada em tempo de execução
 - São pesadas para calcular todo frame

5. Tipos de Colisão

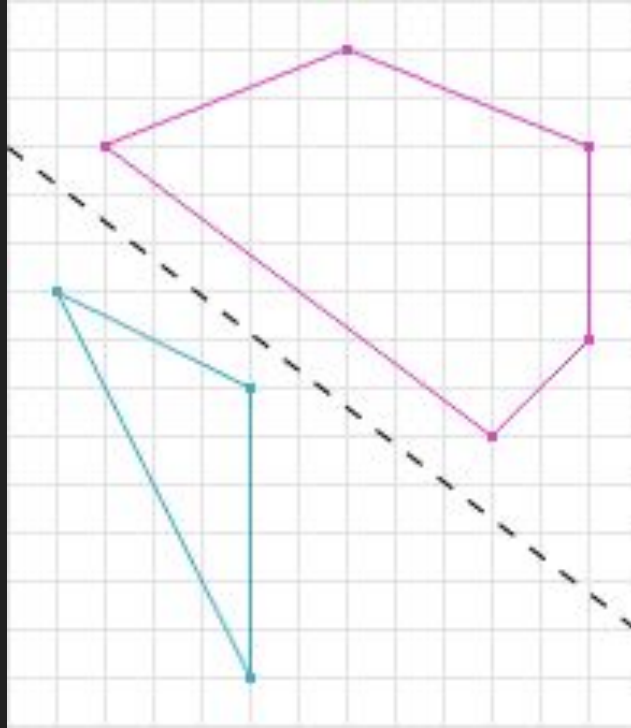
→ SAT - Separating Axis Theorem

- ◆ A ideia é testar a colisão em diferentes eixos dos objetos, projetando eles em uma reta ou em um plano
- ◆ Algumas bibliotecas de física 2D (Box2D) implementam o SAT e deixam o usuário escolher qual algoritmo usar
- ◆ Complexidade aumenta com a quantidade de vértices

[Exemplo de implementação](#)



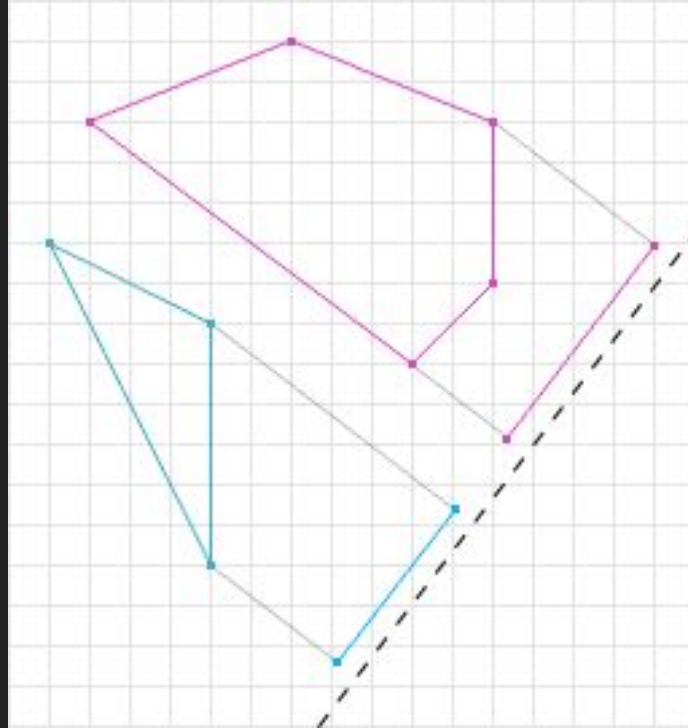
5. Tipos de Colisão



SAT - Separating Axis Theorem



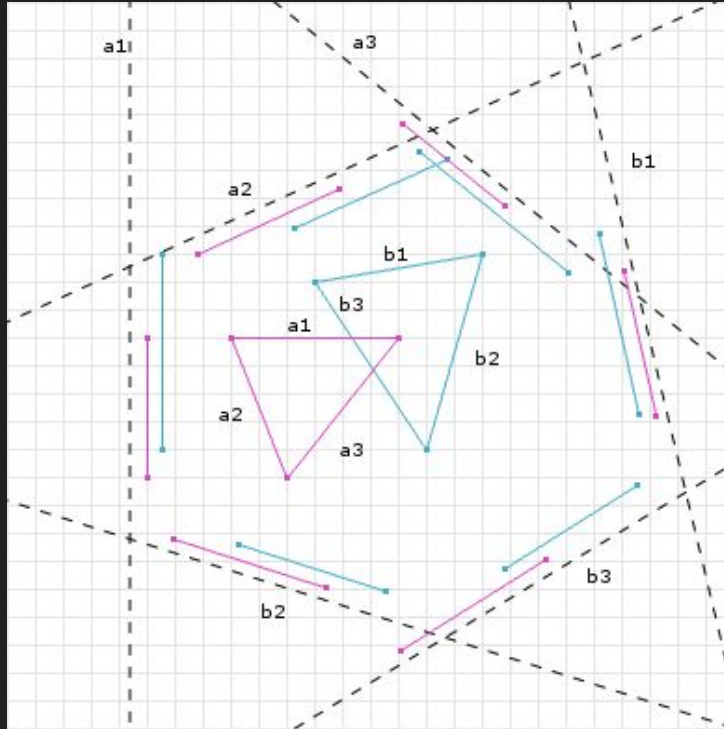
5. Tipos de Colisão



SAT - Separating Axis Theorem



5. Tipos de Colisão



SAT - Separating Axis Theorem



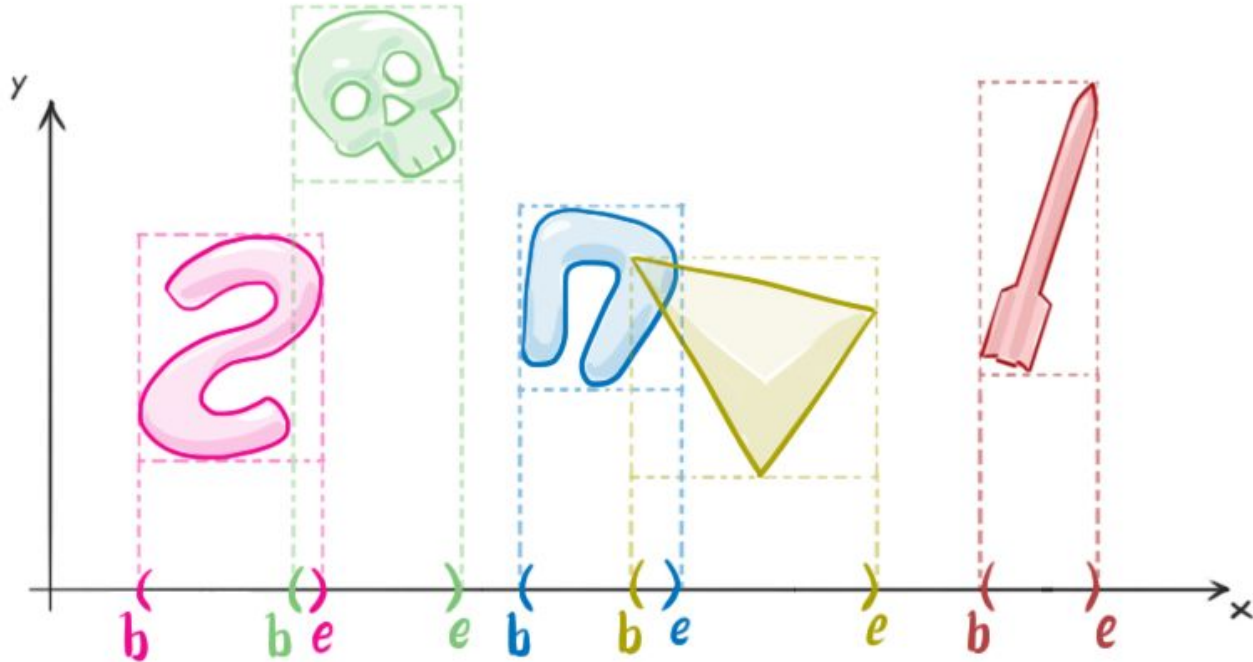
5. Tipos de Colisão

→ Sweep and Prune (SAP)

- ◆ A ideia é ordenar os objetos num eixo e fazer pares de possíveis colisões
 - Só pegamos o máximo e o mínimo do colisor
- ◆ Em 2D temos eixos X ou Y
- ◆ Em 3D temos planos X, Y ou Z



5. Tipos de Colisão



5. Tipos de Colisão

→ Sweep and Prune

- ◆ É preciso ordenar a lista quando atualizar as colisões, então entra a complexidade de computação e memória
- ◆ Um eixo pode ter muitas possíveis colisões e no outro eixo não, por exemplo, Mário
- ◆ Calcular o mínimo e o máximo do colisor pode ser difícil, como malhas 3D



5. Tipos de Colisão

Colisores na Unity

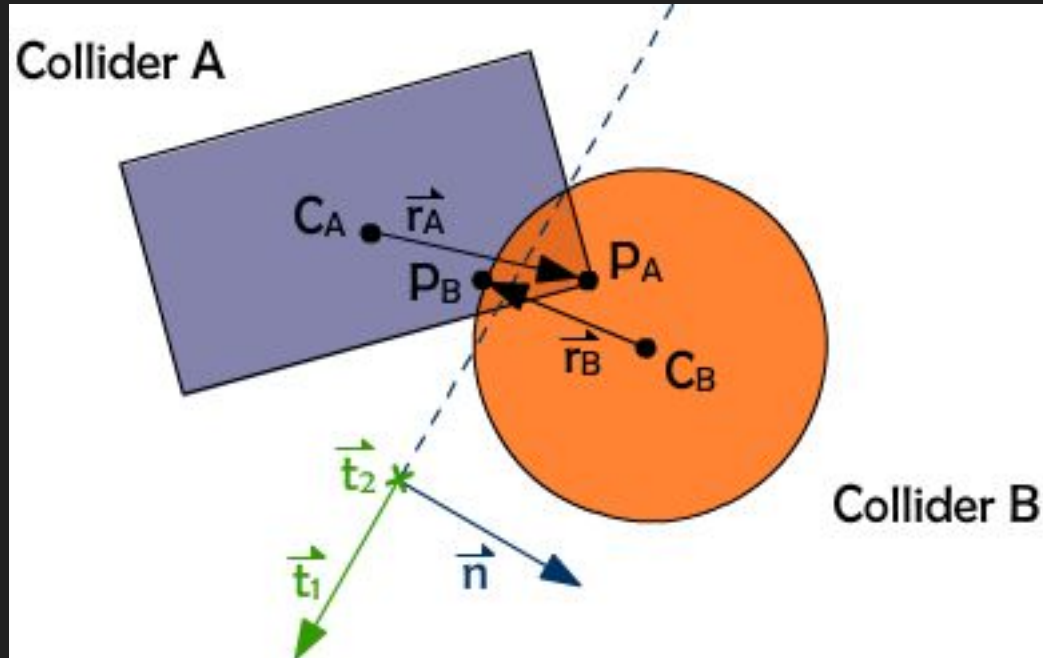


6. Resposta

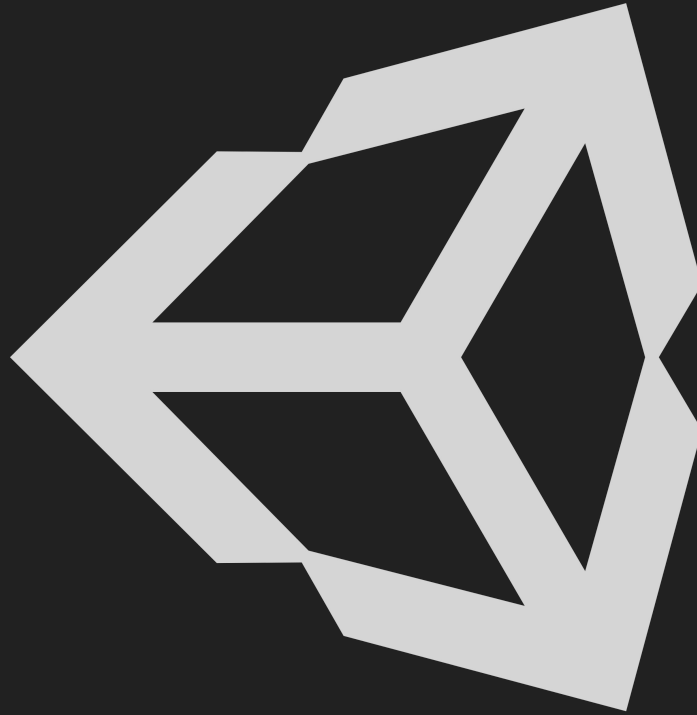
→ Informação da colisão

- ◆ Normal
- ◆ Ponto de Contato
- ◆ Profundidade

6. Resposta



UNITY TIME !!!! - Settings/Colisão/Raycast



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] (The Morgan Kaufmann Series in Interactive 3-D Technology) Christer Ericson-Real-Time Collision Detection-Morgan Kaufmann (2005)