Física para Jogos Digitais – Introdução Conceitos Básicos

Rossana Baptista Queiroz

Assuntos

- Apresentação Geral da Disciplina
- Física para Jogos?!?
- Introdução às Engines Físicas
- Alguns conceitos básicos...
- Exercício!! ☺

Apresentação Geral

- Eu! – fellowsheep@gmail.com
  - Bacharel em Ciência da Computação pela Unisinos ☺
  - Doutoranda (*ad æternum)* em Ciência da Computação pela PUCRS
    - Área de Pesquisa: Animação Facial em Tempo Real

fellowsheep on Steam

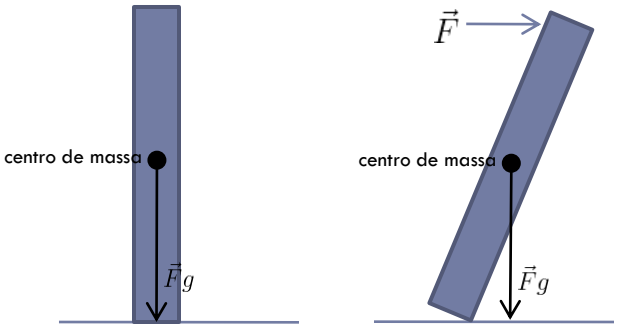
- Nosso canal de comunicação: Moodle!!
  - www.moodle.unisinos.br

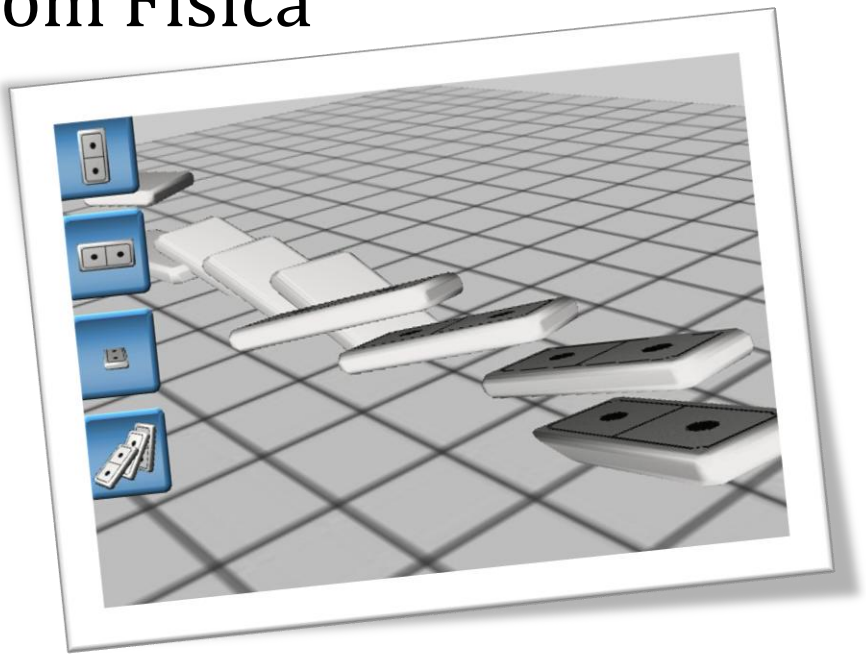
Apresentação Geral

- Vocês!
  - Semestre? (teoricamente 3º)
    - Estão fazendo ou fizeram Processamento Gráfico?
      - OpenGL...
    - Estão fazendo CG?
  - Já desenvolveu algo em/para jogos **COM FÍSICA**?
    - O que?
    - Portfólio pessoal...
    - Expectativas na área...

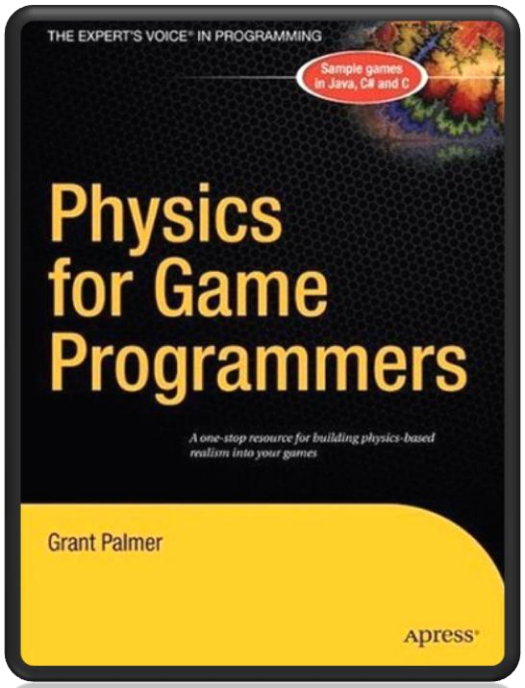
Que tipo de coisas vamos fazer?

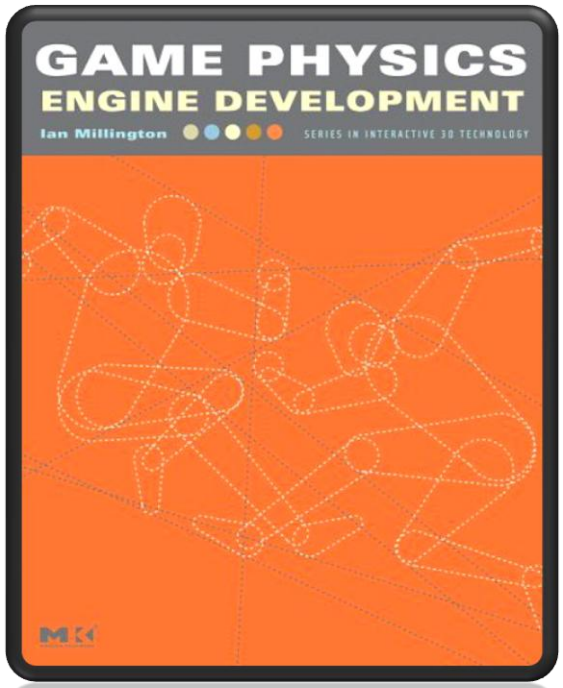
- Como utilizar Física em jogos?
- Conhecer as principais *engines*
- Experimentar a *engines* **Box2D** e ~~talvez~~ **PhysX** em aula
- Criar um **projeto** de jogo com Física
  - Portfólio ☺

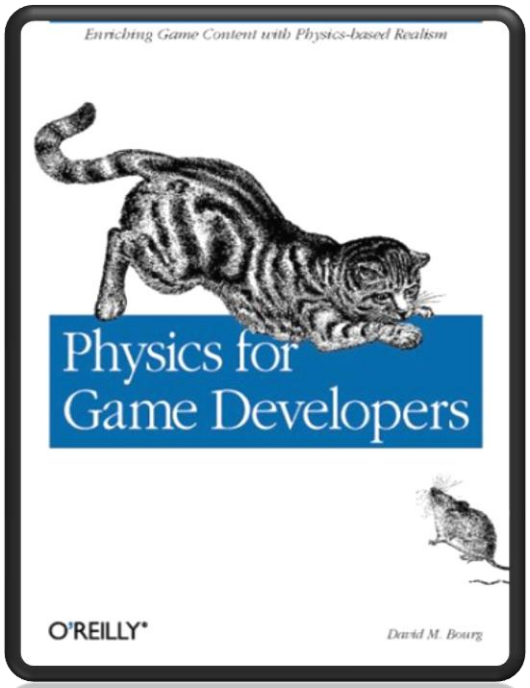




Referências Recomendadas

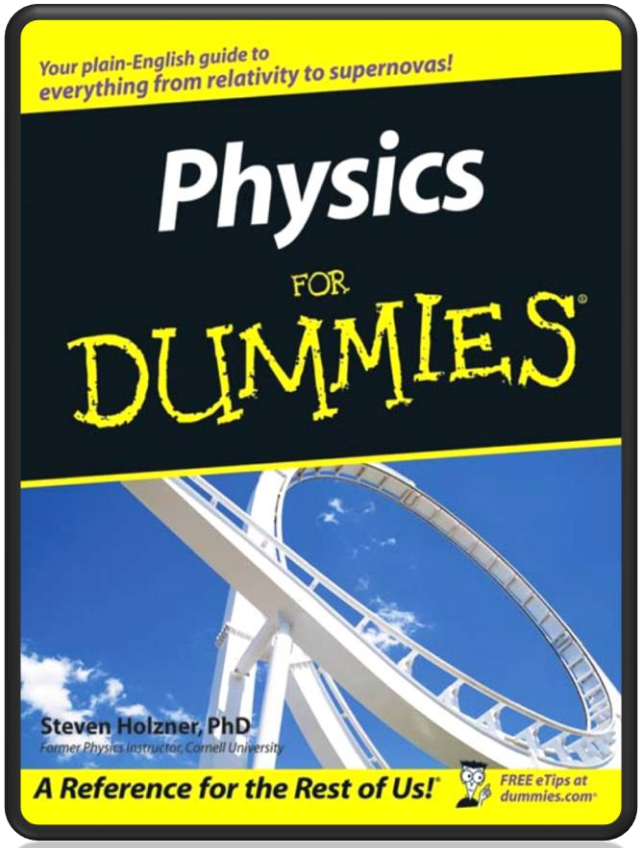






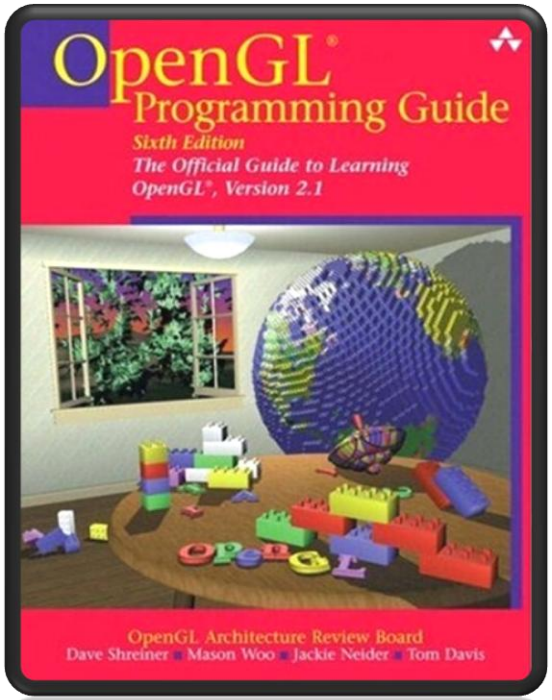
Referências Recomendadas

- Não muito mais que a Física do Ensino Médio...



Referências Técnicas

- OpenGL

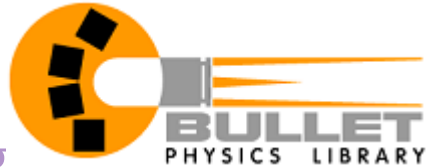


- COHEN, Marcelo, MANSSOUR, Isabel. OpenGL : uma abordagem prática e objetiva. São Paulo : Novatec, 2006. 478 p.
- SHREINER, Dave et al. OpenGL Programming Guide: The Official Guide to Learning OpenGL . Reading, MA: Addison-Wesley, 5 edition, 2005. 896 p.

Referências técnicas (*Engines*)

- Box 2D
  - http://box2d.org/manual.pdf
  - Prefere C#?
    - https://code.google.com/p/box2dx/
  - Prefere em Flash??
    - http://www.box2dflash.org/

- NVIDIA PhysX
  - http://www.geforce.com/hardware/technology/physx
- Prefere a Bullet?
  - http://bulletphysics.org



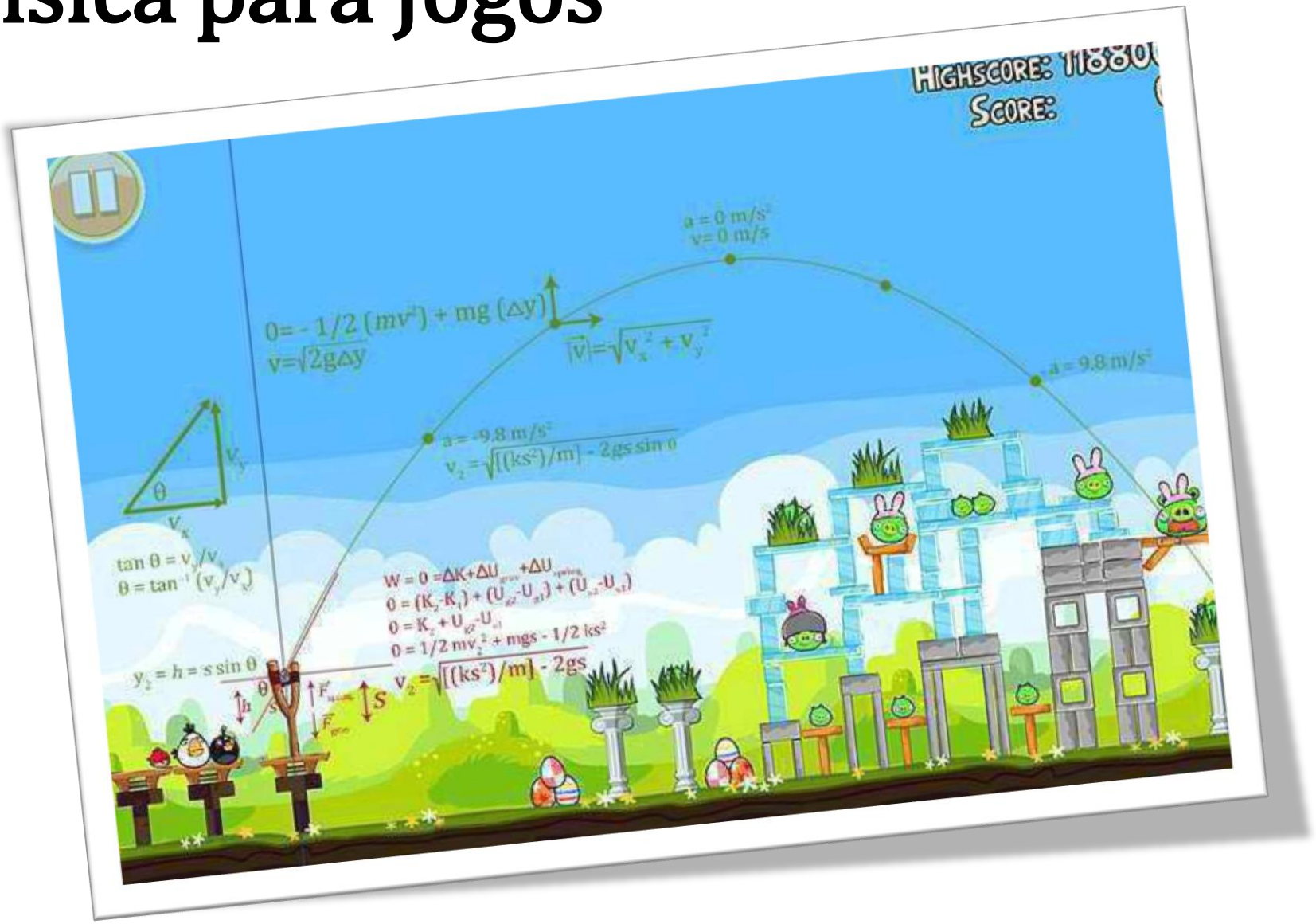
Avaliação

- **Grau A**
  - Mini-seminário sobre *engines* : 2,0
  - Exercícios diversos (teóricos e práticos): 3,0
  - Mini-projeto de jogo 2D: peso 5,0
- **Grau B**
  - Exercício *engine 3D*: 5,0
  - Projeto Final 6,0
- **Grau C**
  - Prova substitutiva do Grau A OU do Grau B

Cronograma e Faltas

- No máximo 4,5 faltas!
- **Grau A**
  - Mini-seminário: 01/09
  - Entrega do trabalho: 29/09
- **Grau B**
  - Exercício 3D: 03/11
  - Entrega do trabalho: 24/11
- **Grau C**
  - Prova: 08/12

Física para Jogos



Motivação

- Métodos de interpolação
  - Linear
  - Curvas paramétricas cúbicas
    - Hermite
    - Bèzier
    - B-Spline
    - Catmull-Rom

- Definem uma trajetória ao longo do tempo, dado um conjunto de pontos

Mas...

- Como definir a trajetória de um tiro de canhão de forma convincente?

Mas...

Mas...

???

Mas... Física?!?

- É necessário buscarmos uma forma de reproduzir a realidade.

- Através da Física podemos simular a reação dos objetos quando em contato com *forças*
  - Gravidade, viscosidade, atrito, resultantes de colisões, etc...

- Variabilidade de ações em contraponto a movimentos pré-gravados

Pode ser mais específica?

- Cálculo de trajetórias
- Movimento dirigido de corpos rígidos
- Corpos articulados
- Tratamento de Colisões
- Efeitos Especiais
- Simulação de de partículas
- Fluidos
- Tecidos

Física

- Composta por várias áreas:
  - Mecânica
  - Termologia
  - Ondulatória
  - Acústica
  - Óptica
  - Eletromagnetismo
  - Física Moderna
  - Teoria da relatividade
  - Física de Partículas
  - Física Atômica
  - Física Molecular
  - Física Nuclear
  - Mecânica Quântica
  - Mecânica Estatística

Física

- Áreas de interesse
  - Ótica
  - Mecânica
    - Estática
    - Cinemática
    - Dinâmica

Física

- Áreas de interesse
  - Ótica: **iluminação, renderização**
  - Mecânica: **animação**
    - Estática
    - Cinemática
    - Dinâmica

Física

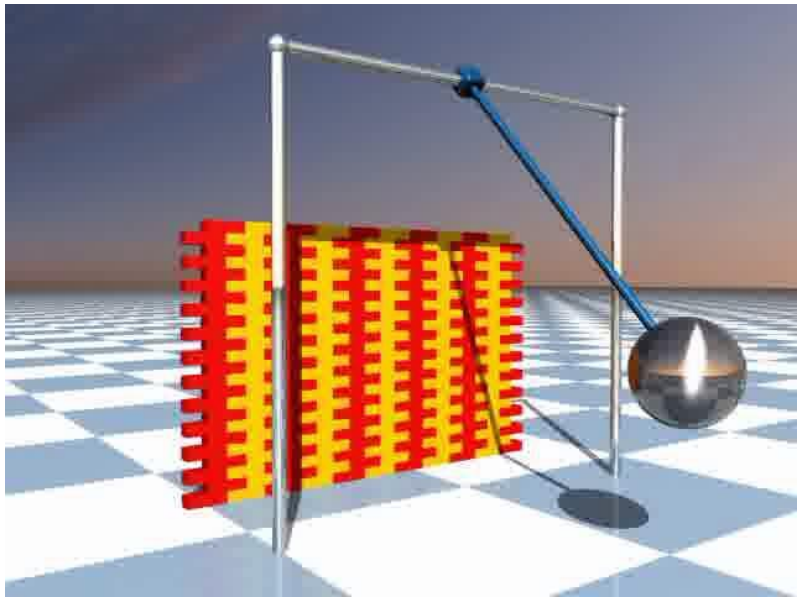
- **Mecânica**
  - **Estática:** estuda as forças atuantes em um corpo que está em <u>equilíbrio estático</u>
    - Equilíbrio estático é o caso especial de equilíbrio mecânico observado em um objeto em repouso

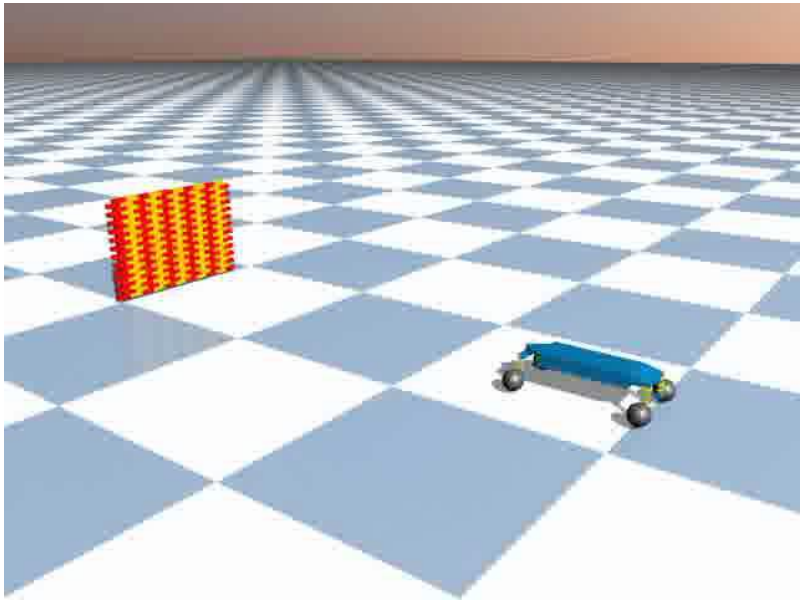
  - **Cinemática:** estuda os movimentos dos corpos, sem se preocupar com a análise de suas causas.

  - **Dinâmica:** estuda as relações entre as forças e os movimentos que são produzidos por estas.

Aplicações da Física em CG

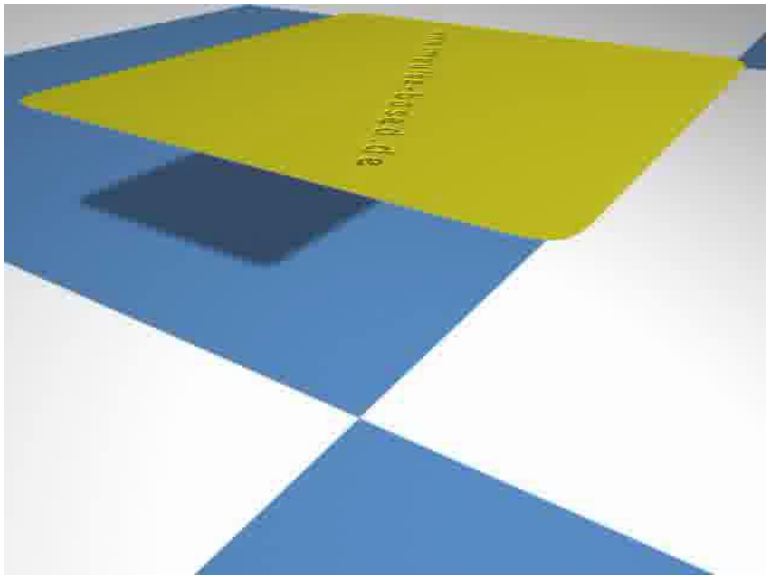
- Tratamento de colisão

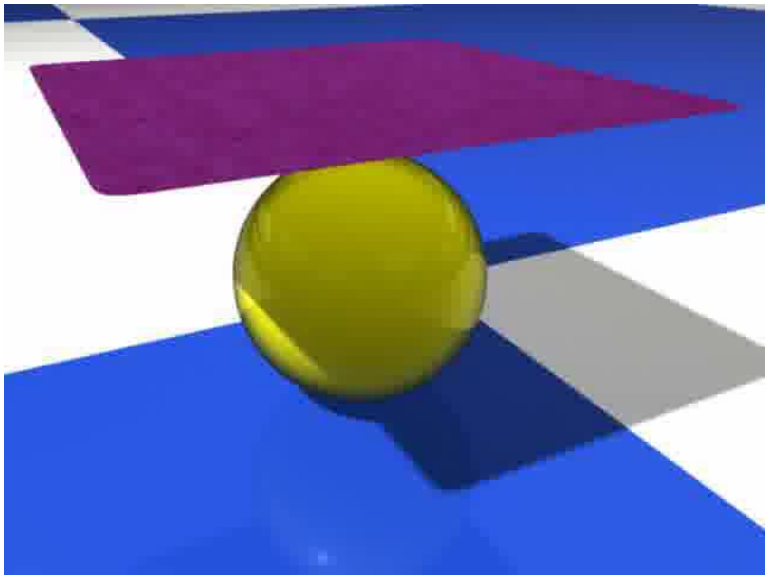




Aplicações da Física em CG

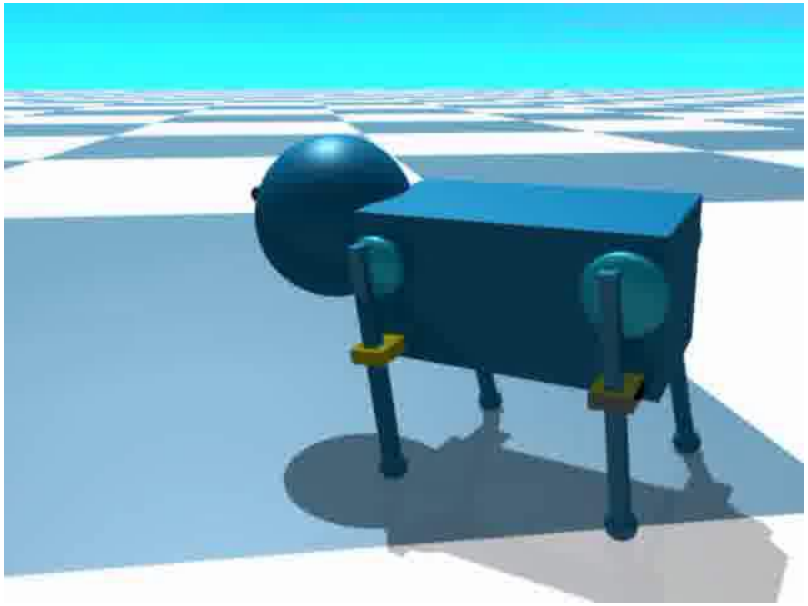
- Simulação de tecidos

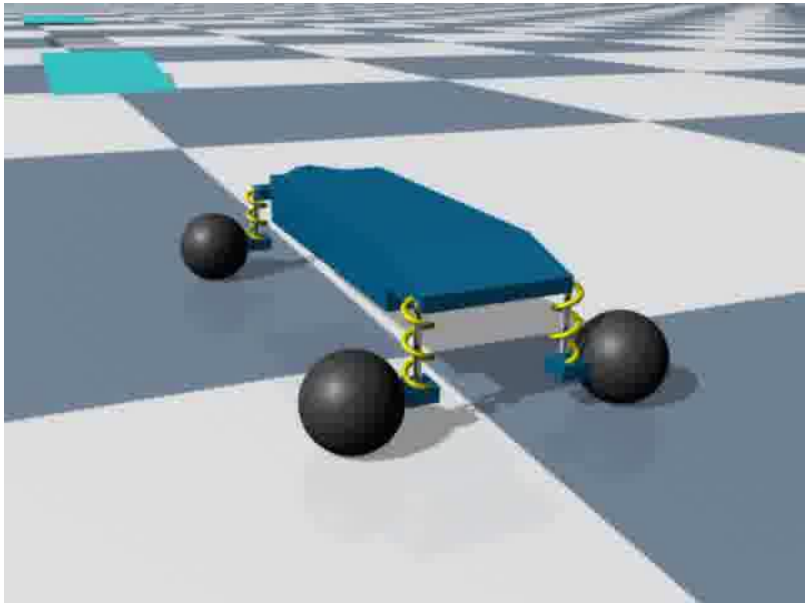




Aplicações da Física em CG

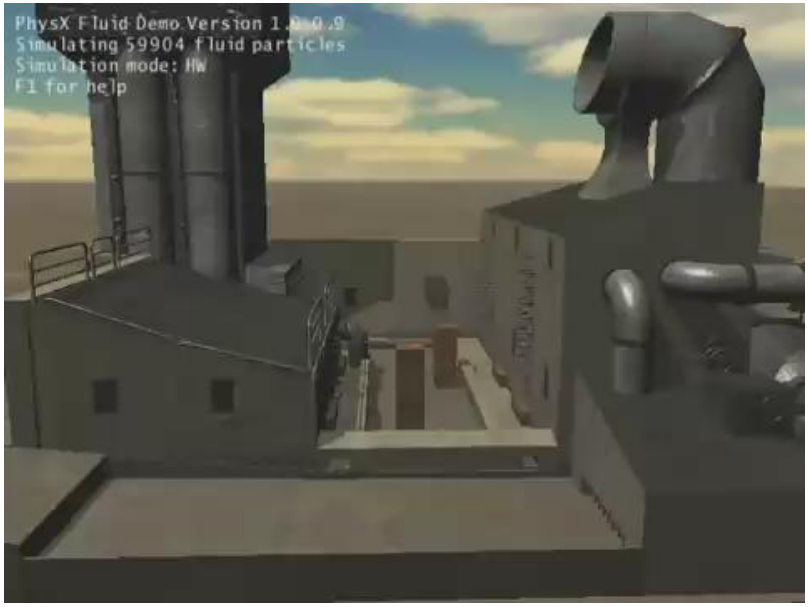
- Animação de corpos rígidos

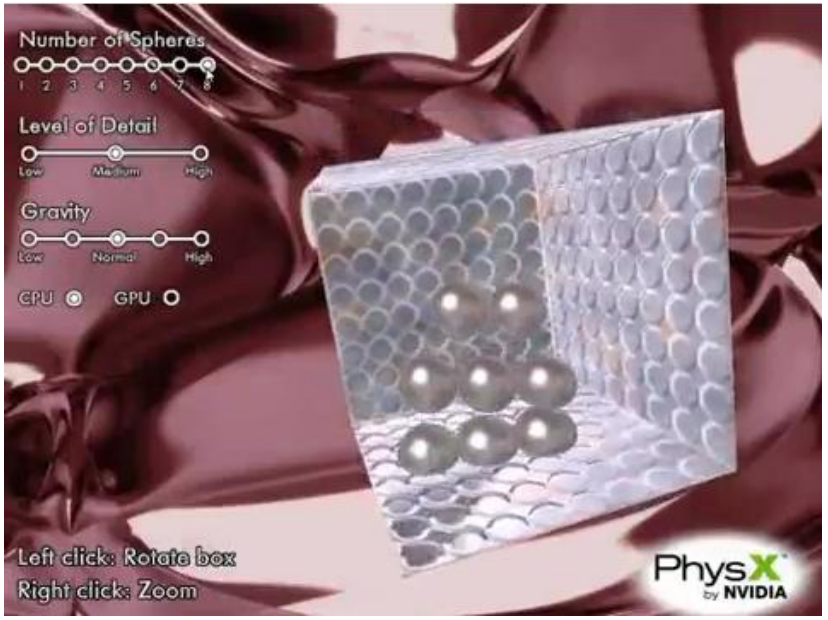




Aplicações da Física em CG

- Fluídos e Corpos deformáveis





Aplicações da Física em Jogos

Física em Jogos

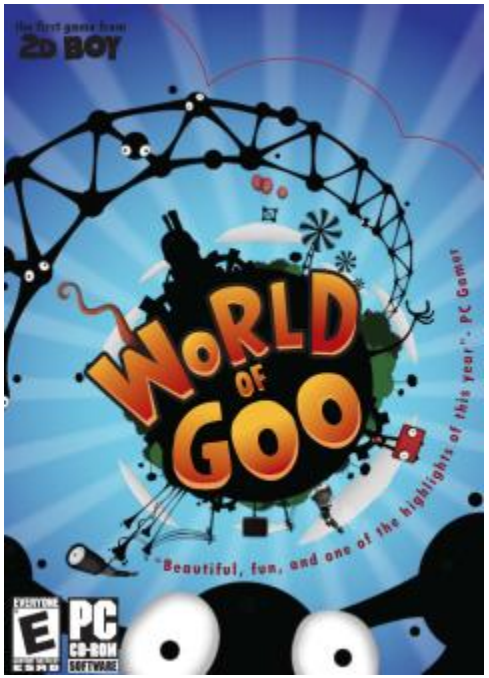
- Física de Produção (cinema) vs. Física para jogos
- Física de efeitos vs. Física de *gameplay*
  - Discussão interessante em http://enthusiast.hardocp.com/article/2006/07/10/effects_physics_gameplay_explored
  - Exemplos (efeitos)
    - Ondas na água
    - Cabelos e roupas balançando
    - Pedaços que voam de uma explosão
    - Objetos que colidem e deformam
  - Exemplos (gameplay)
    - Peso do personagem no ambiente
    - Intensidades e movimentos capturados por dispositivos (kinect, wii mote, ps move, acelerômetros do iPhone) usados para estimar forças aplicadas aos objetos do jogo
- Precisão vs. velocidade

Física em Jogos

- Hoje é uma realidade! ☺

Frozenbyte, 2009

PhysX



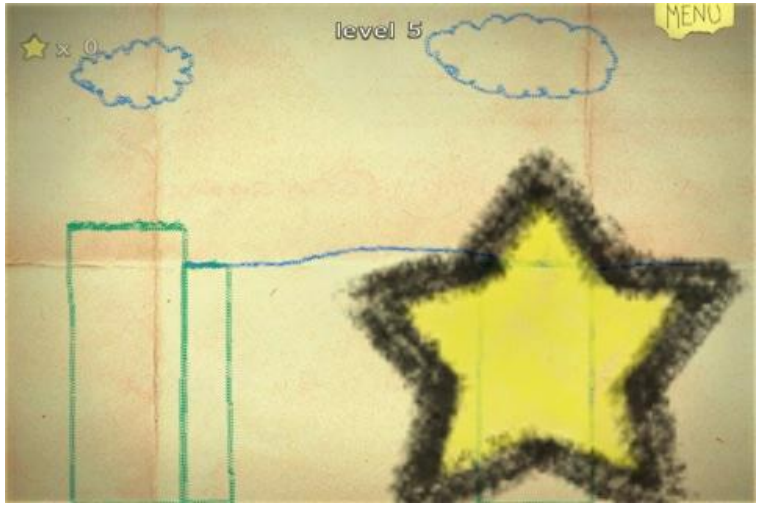
2D Boy, 2008

ODE

Phun, uma dissertação de mestrado...
2008

Algodoo

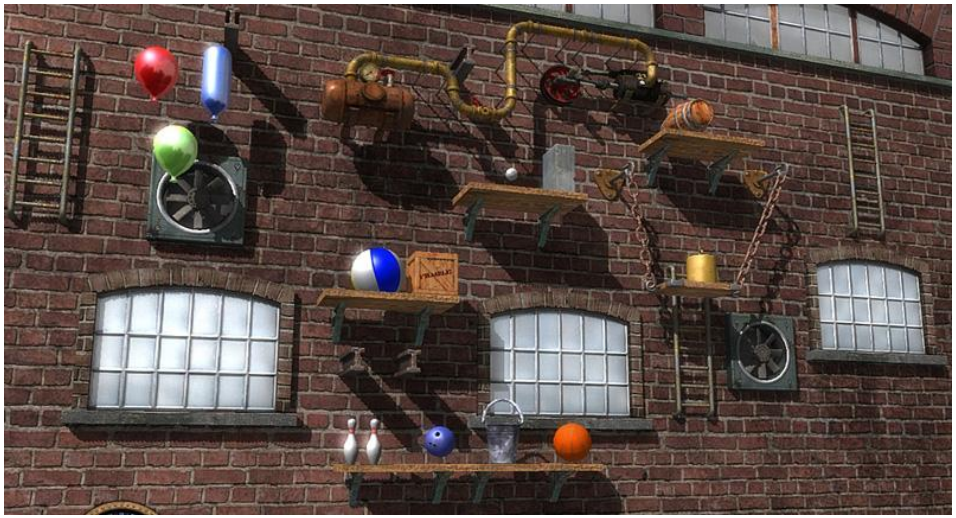
Física em Jogos



Crayon Physics Deluxe, 2009
Box 2D

Angry Birds
Box 2D

Física em Jogos



Crazy Machines 2, 2008
PhysX

Mafia 2, 2010
PhysX

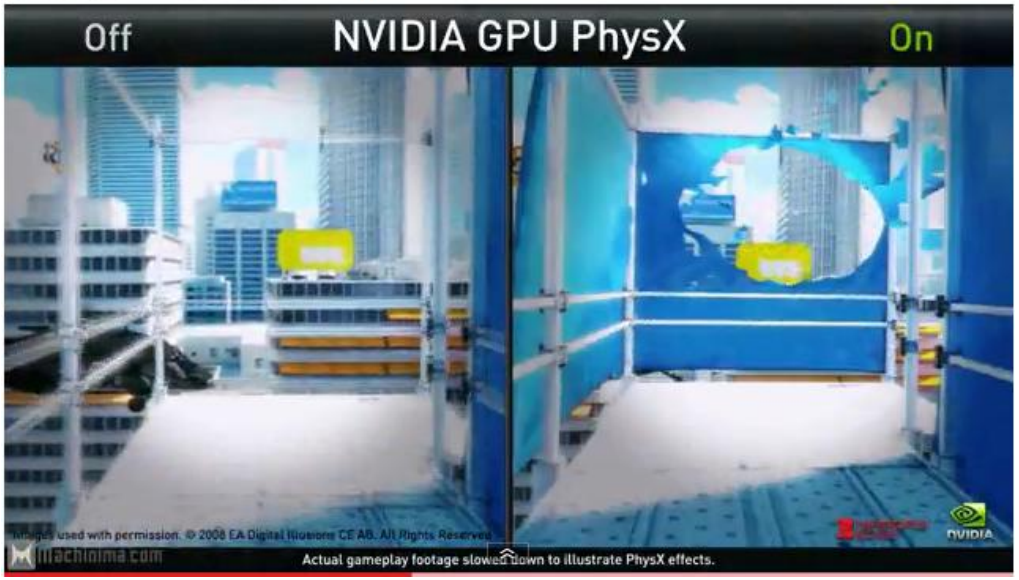
Sacred 2 Physx on/off

PCGamesHardware 572 vídeos Inscrever-se

PCGamesHardware 572 vídeos Inscrever-se

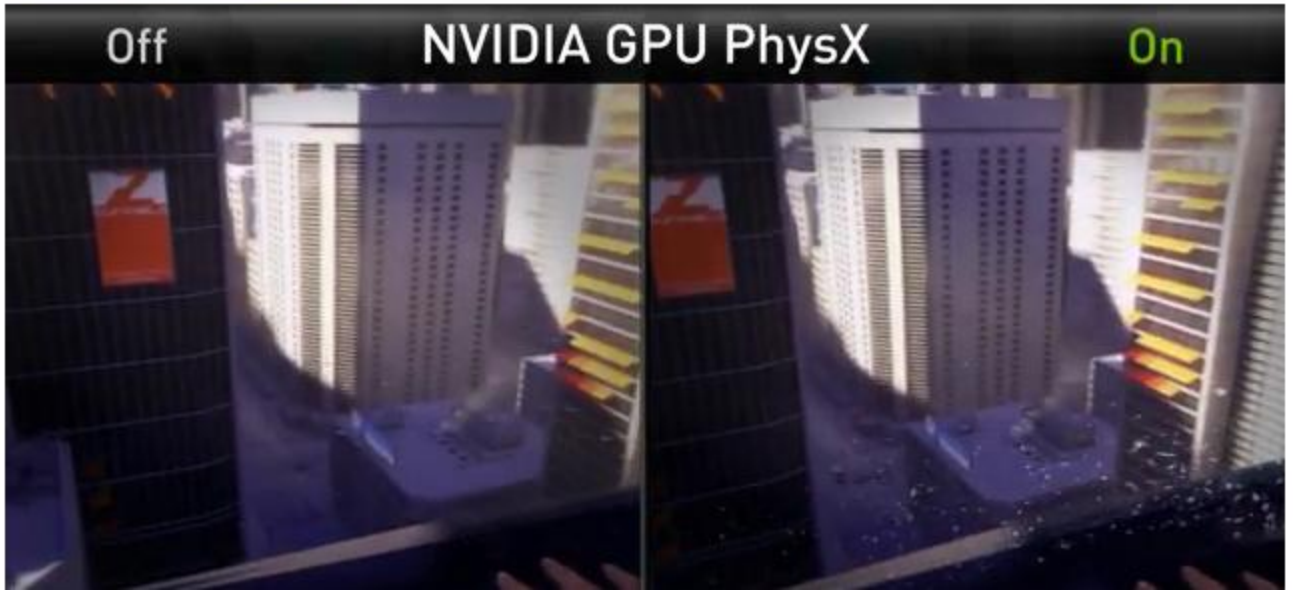
Mirror's Edge - PhysX Comparison (Game Trailer HD)

RIFT PRE-ORDER NOW! 13743 vídeos Inscrever-se



Mirror's Edge - PhysX Comparison (Game Trailer HD)

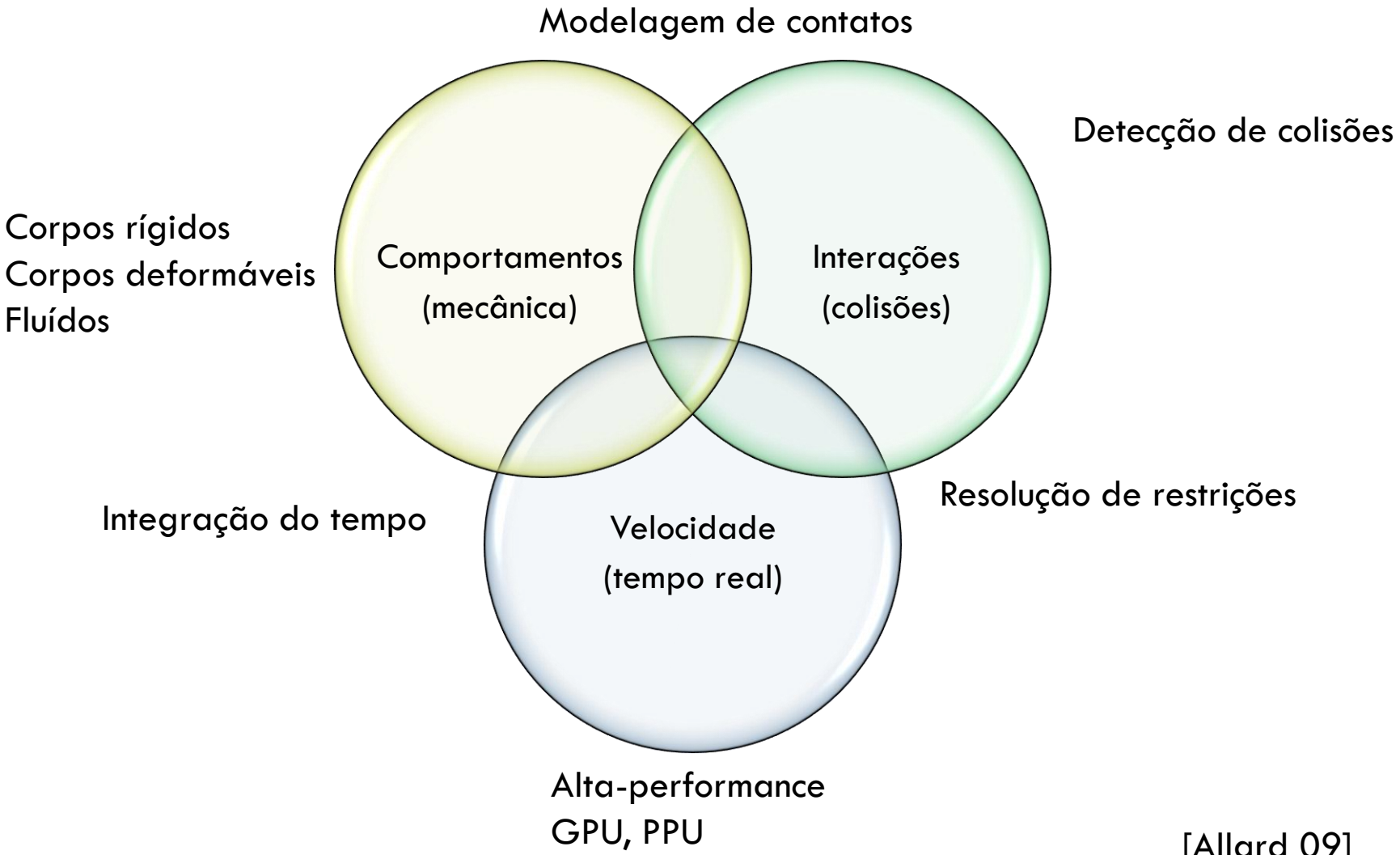
RIFT PRE-ORDER NOW! 13743 vídeos Inscrever-se



Simulação Física

- Etapas
  - Observação da realidade
  - Criação de um modelo físico/matemático
  - Simulação

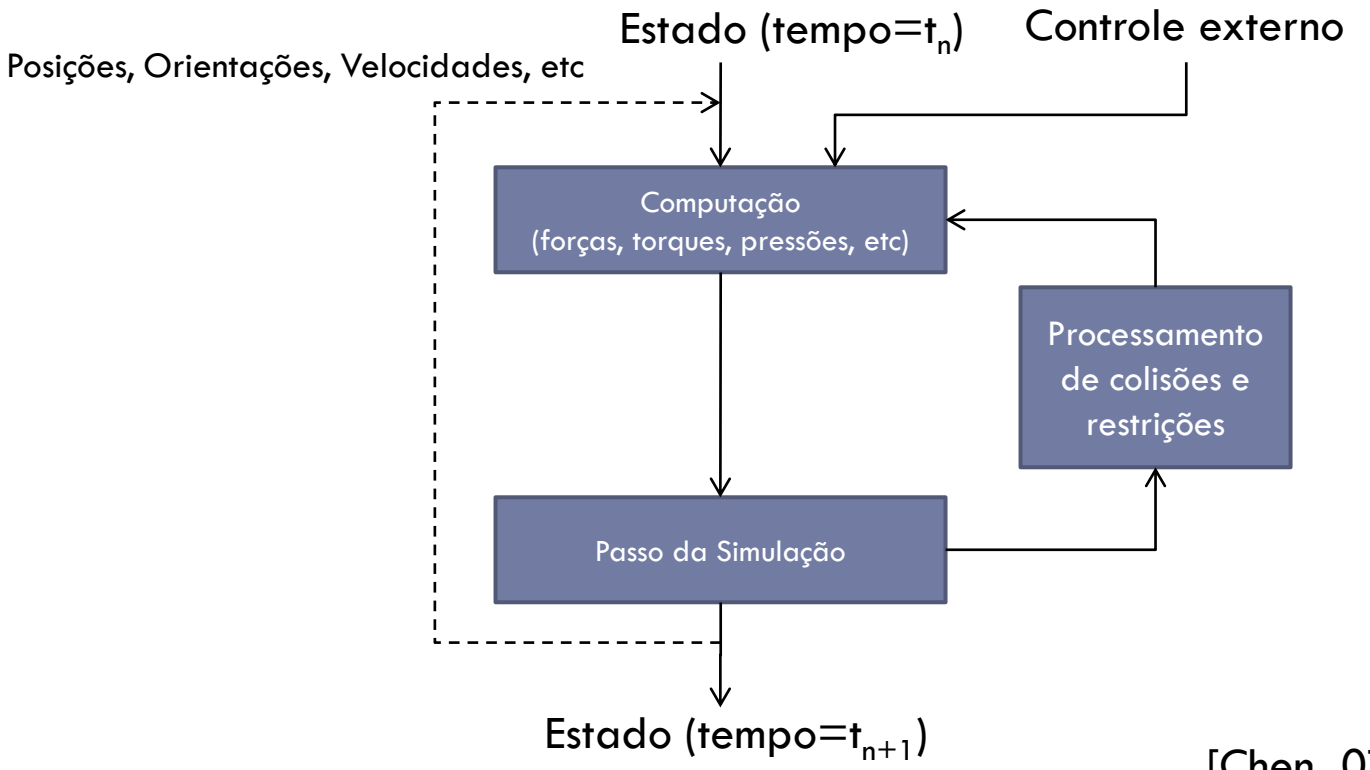
Simulação Física



[Allard 09]

Simulação Física

- Pipeline Geral



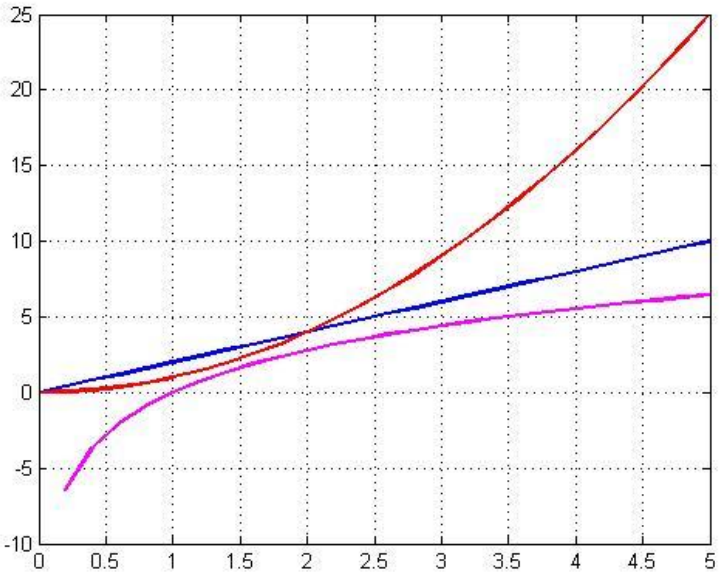
[Chen 07]

Matemática: a linguagem da Física

$x = x_0 + vt$

$x = x_0 + v_0 t + at^2/2$

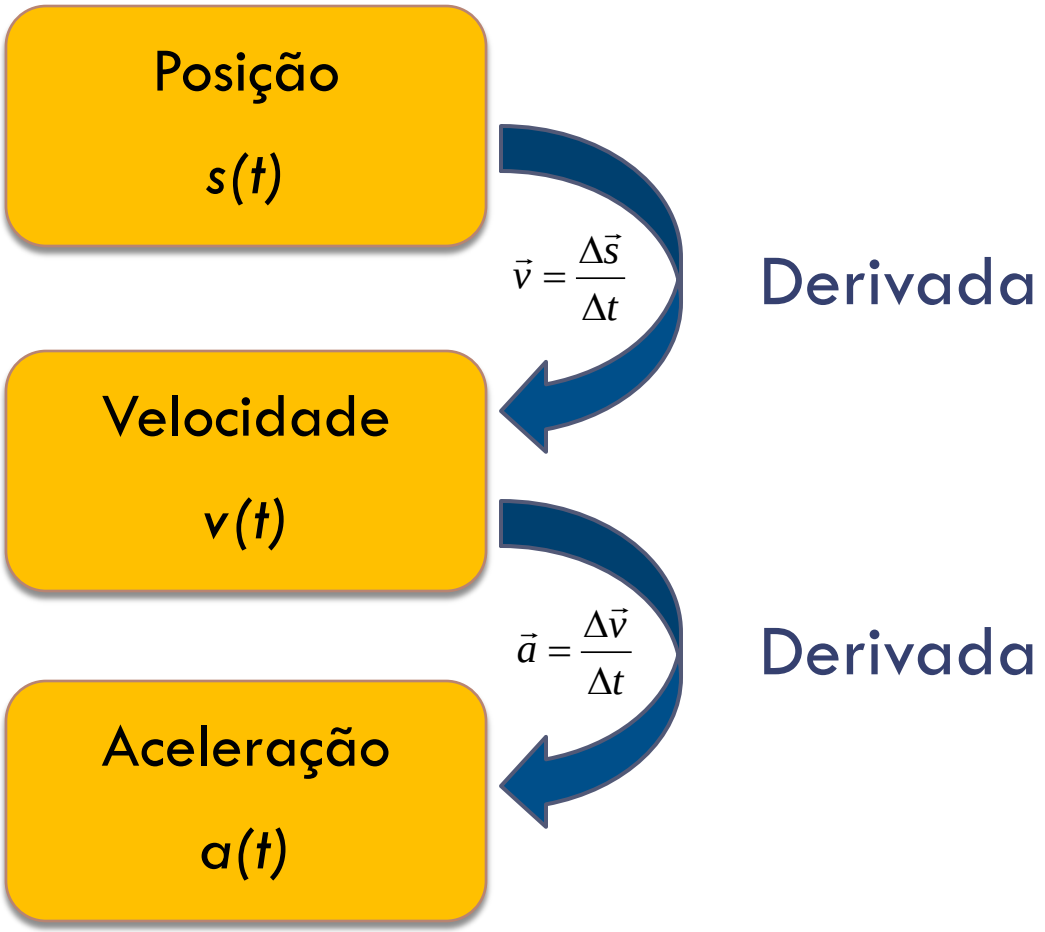
$\beta = log\ (I/I_0)$



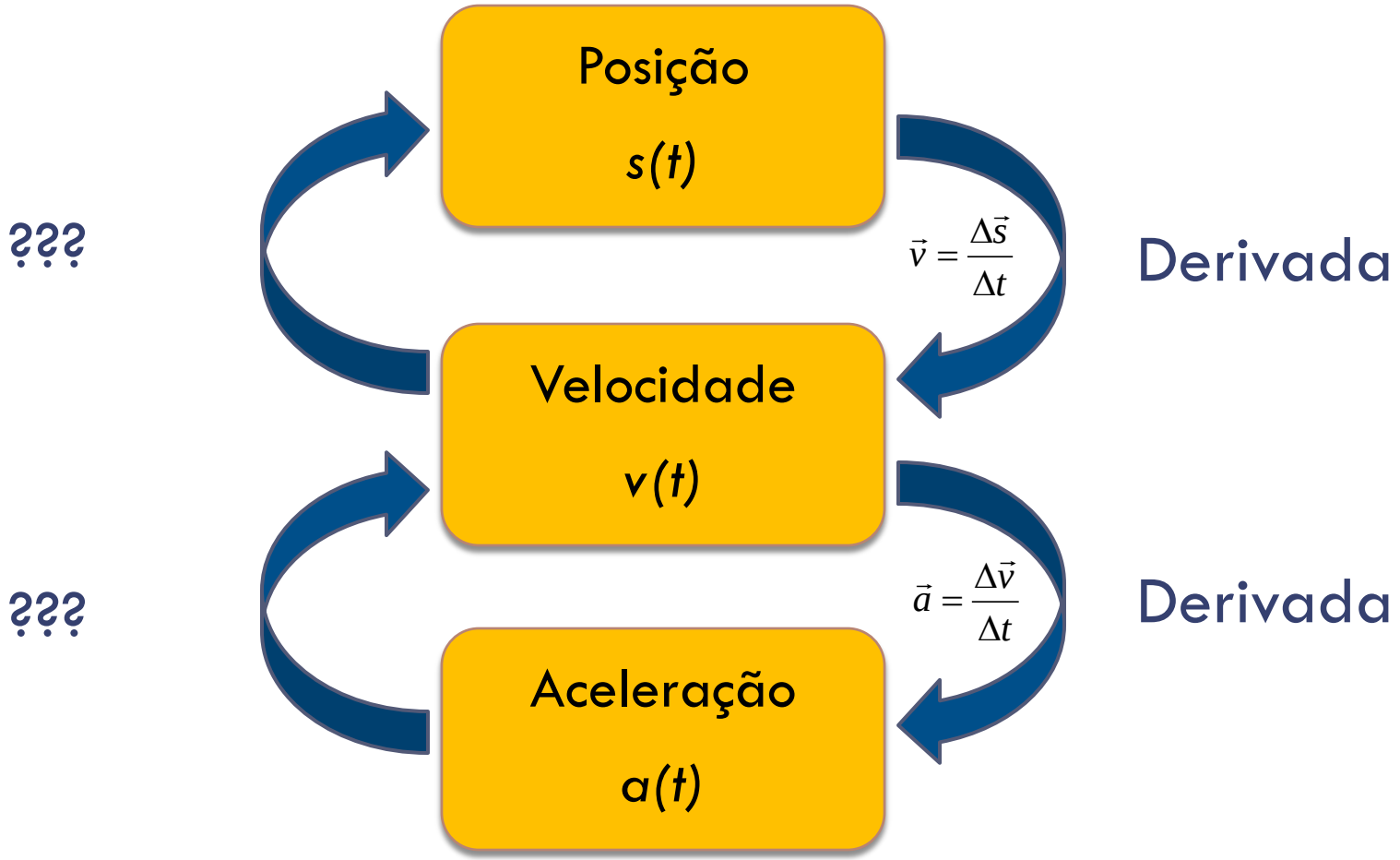
Matemática

- Derivadas
  - A derivada nos informa o quanto uma coisa varia em relação a outra.
  - Quanto que a posição de um corpo varia em relação ao tempo?
    - Velocidade é a derivada da posição em relação ao tempo.
  - Quanto que a velocidade de um corpo varia em relação ao tempo?
    - Aceleração é a derivada da velocidade em relação ao tempo.

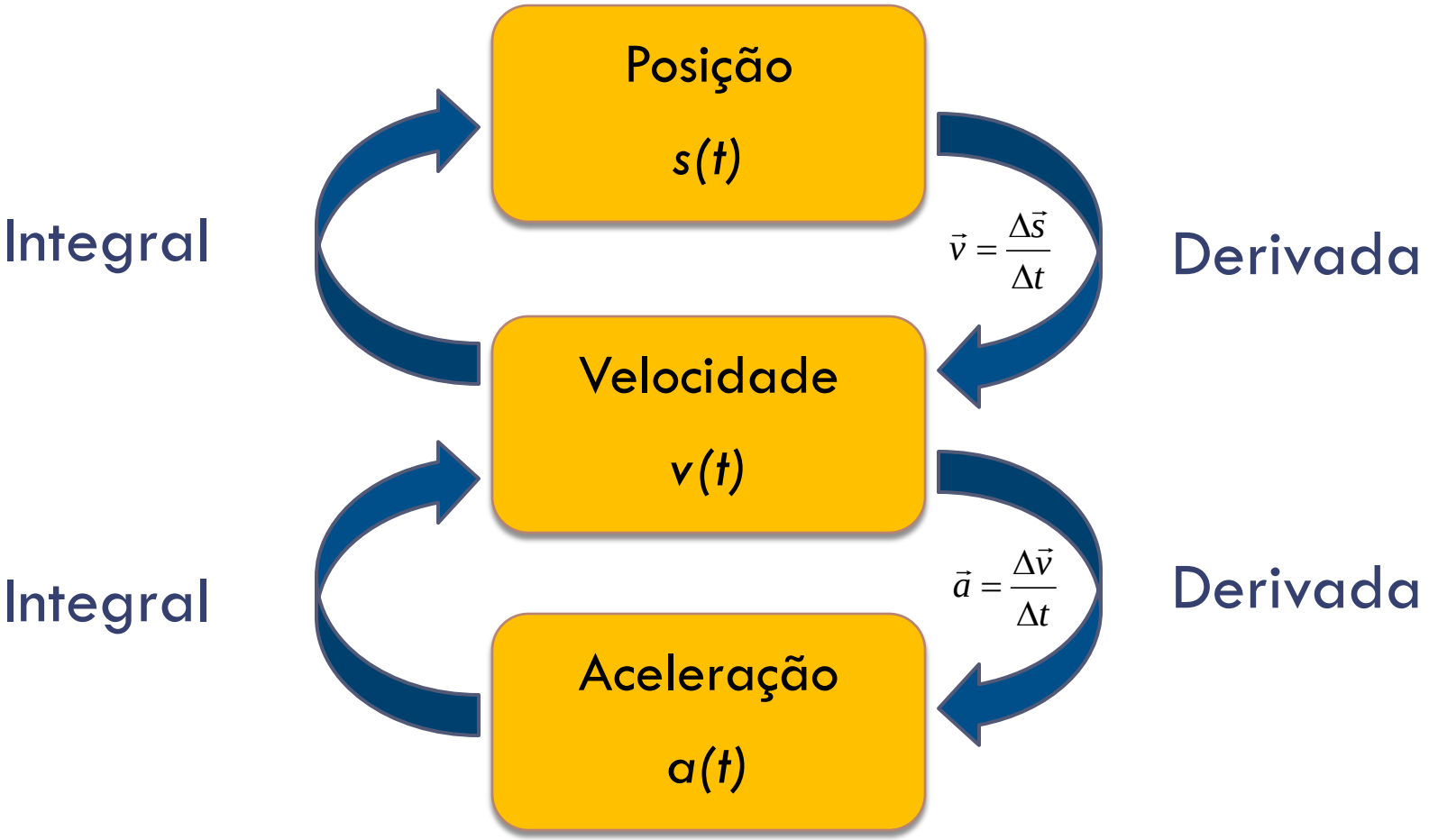
Matemática


Posição
s(t)
$\vec{v} = \frac{\Delta \vec{s}}{\Delta t}$
Derivada
Velocidade
v(t)
$\vec{a} = \frac{\Delta \vec{v}}{\Delta t}$
Derivada
Aceleração
a(t)

Matemática

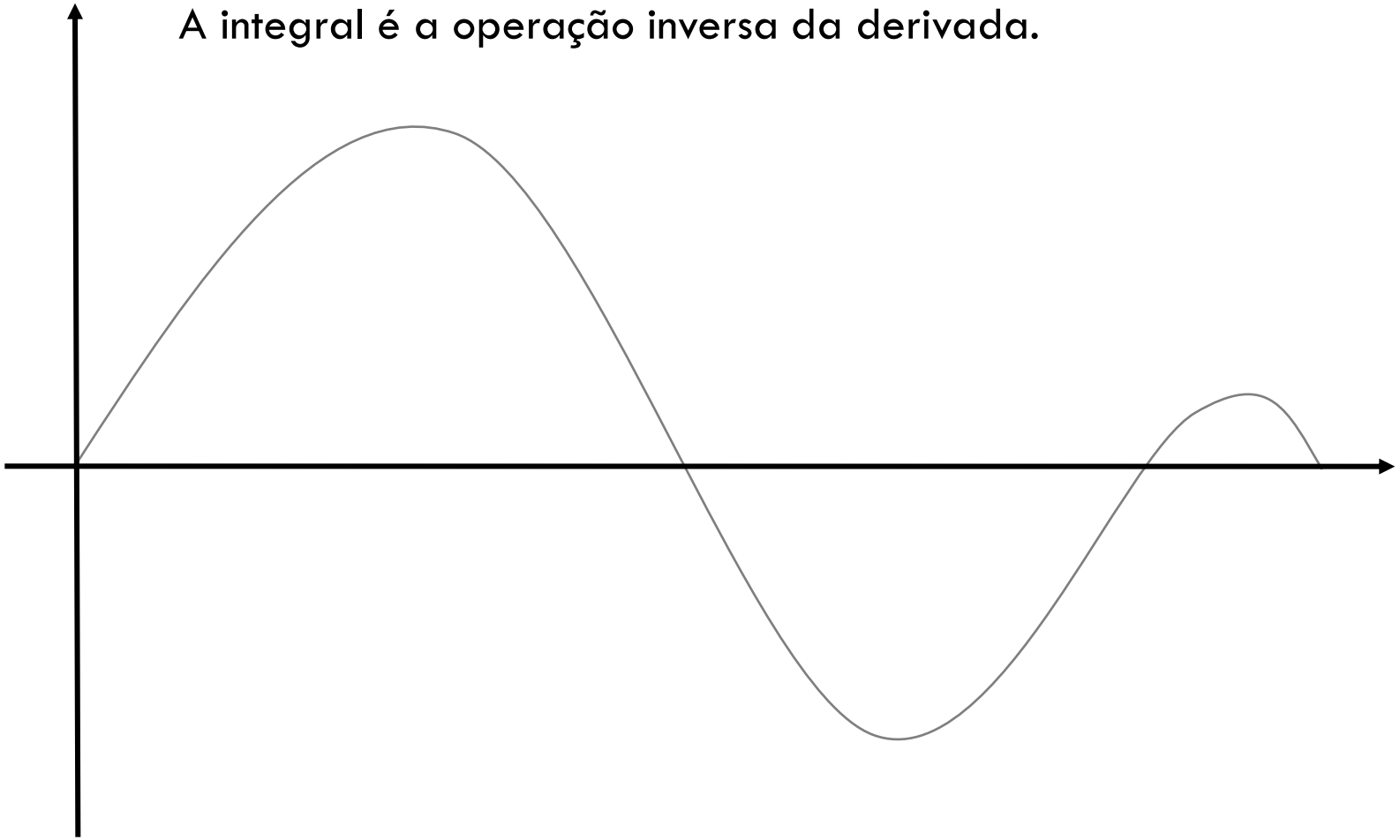

Posição
s(t)
Velocidade
v(t)
Aceleração
a(t)
???
???
$\vec{v} = \frac{\Delta \vec{s}}{\Delta t}$
$\vec{a} = \frac{\Delta \vec{v}}{\Delta t}$
Derivada
Derivada

Matemática


Posição
s(t)
Velocidade
v(t)
Aceleração
a(t)
Integral
Integral
Derivada
Derivada
$\vec{v} = \frac{\Delta \vec{s}}{\Delta t}$
$\vec{a} = \frac{\Delta \vec{v}}{\Delta t}$

Integral

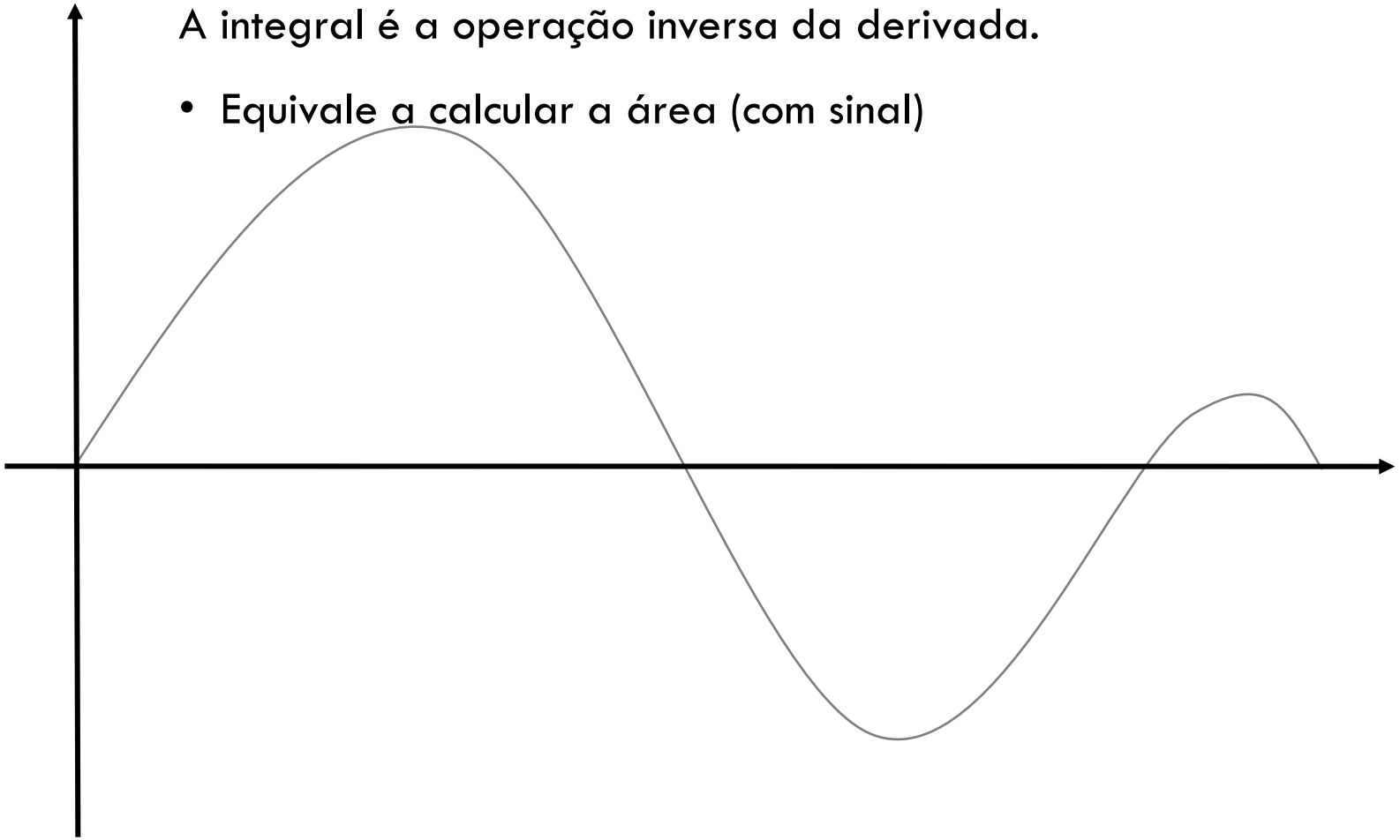
A integral é a operação inversa da derivada.



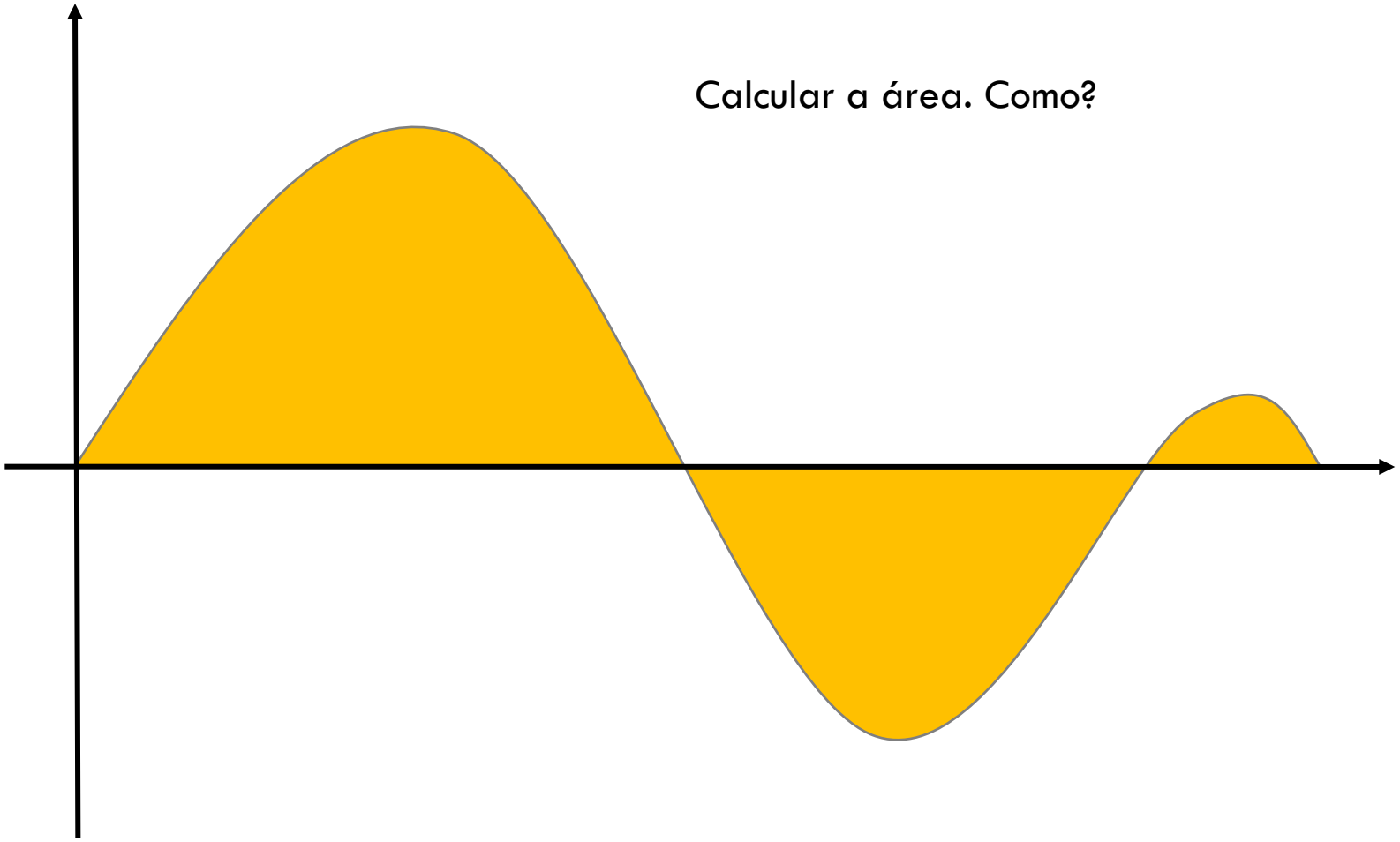
Integral

A integral é a operação inversa da derivada.

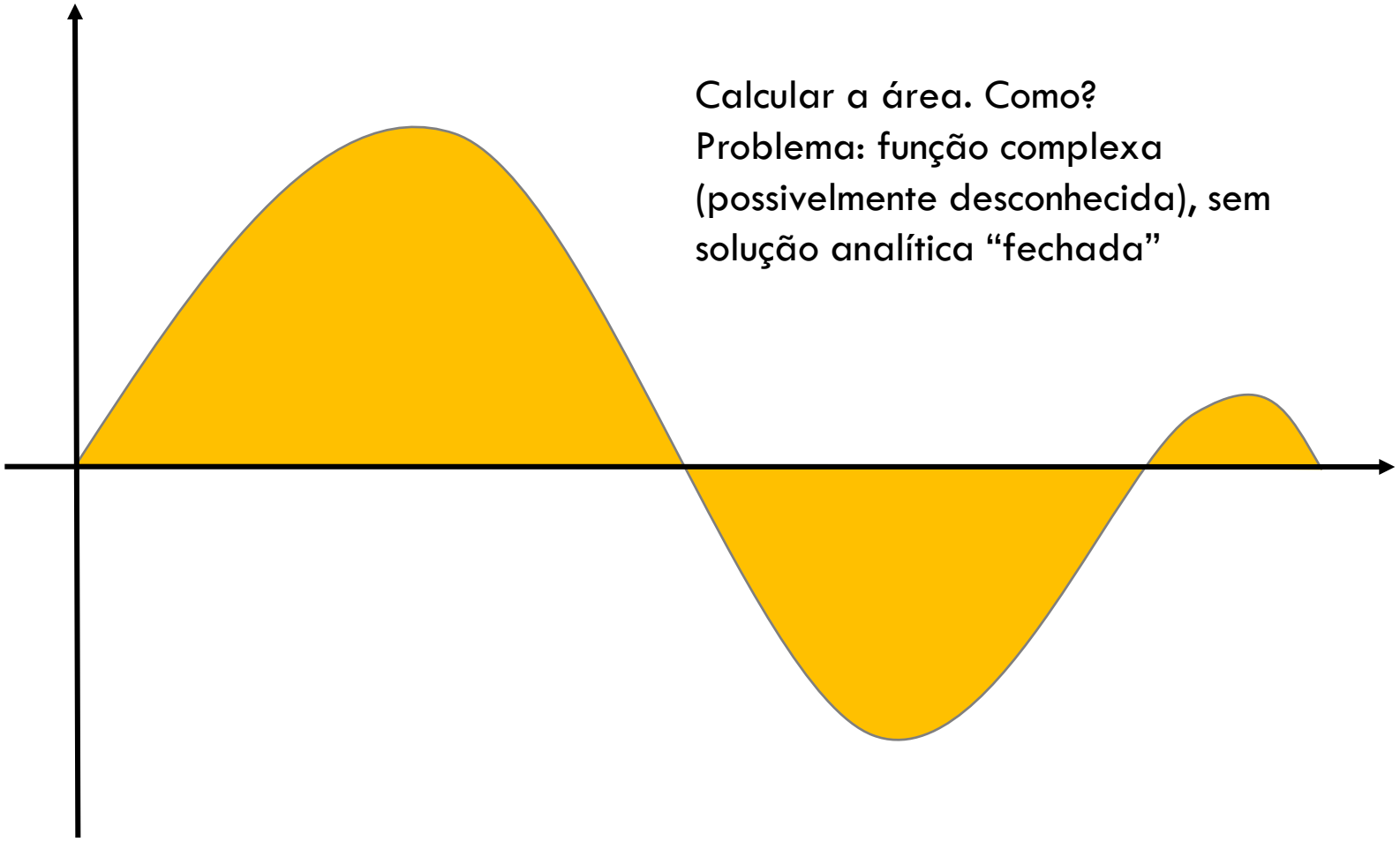
- Equivale a calcular a área (com sinal)



Integral

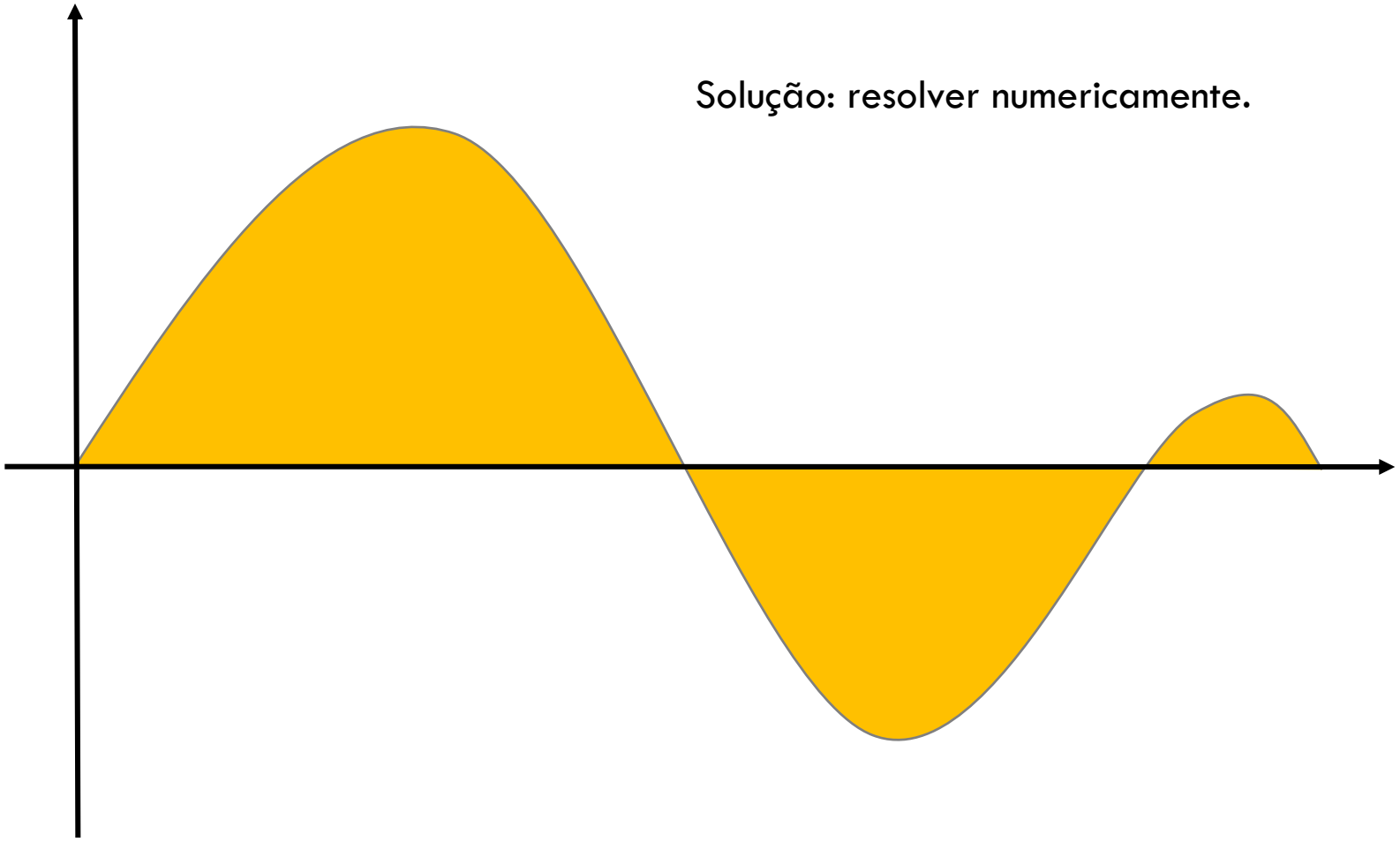


Integral



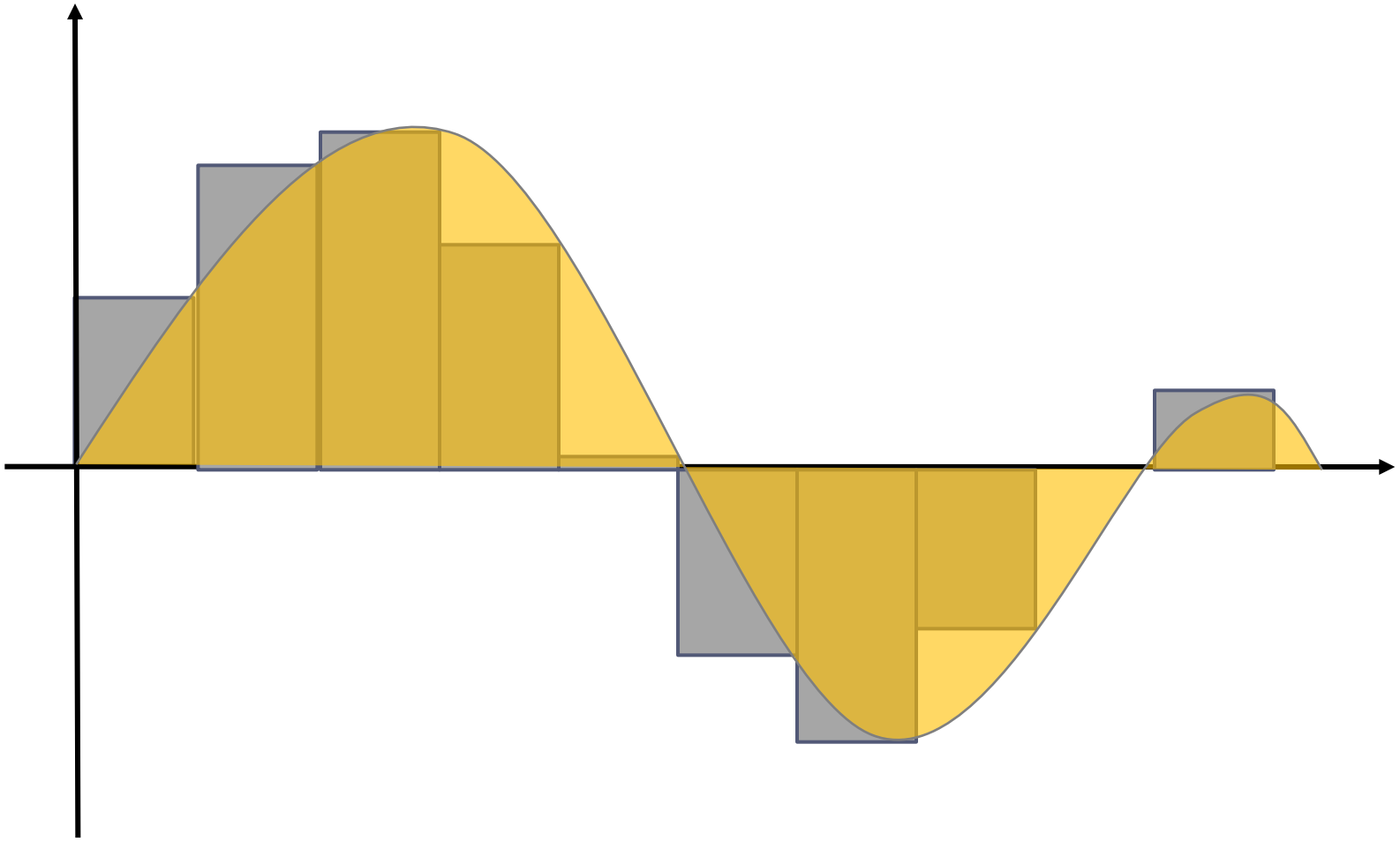
Calcular a área. Como?
Problema: função complexa (possivelmente desconhecida), sem solução analítica “fechada”

Integral

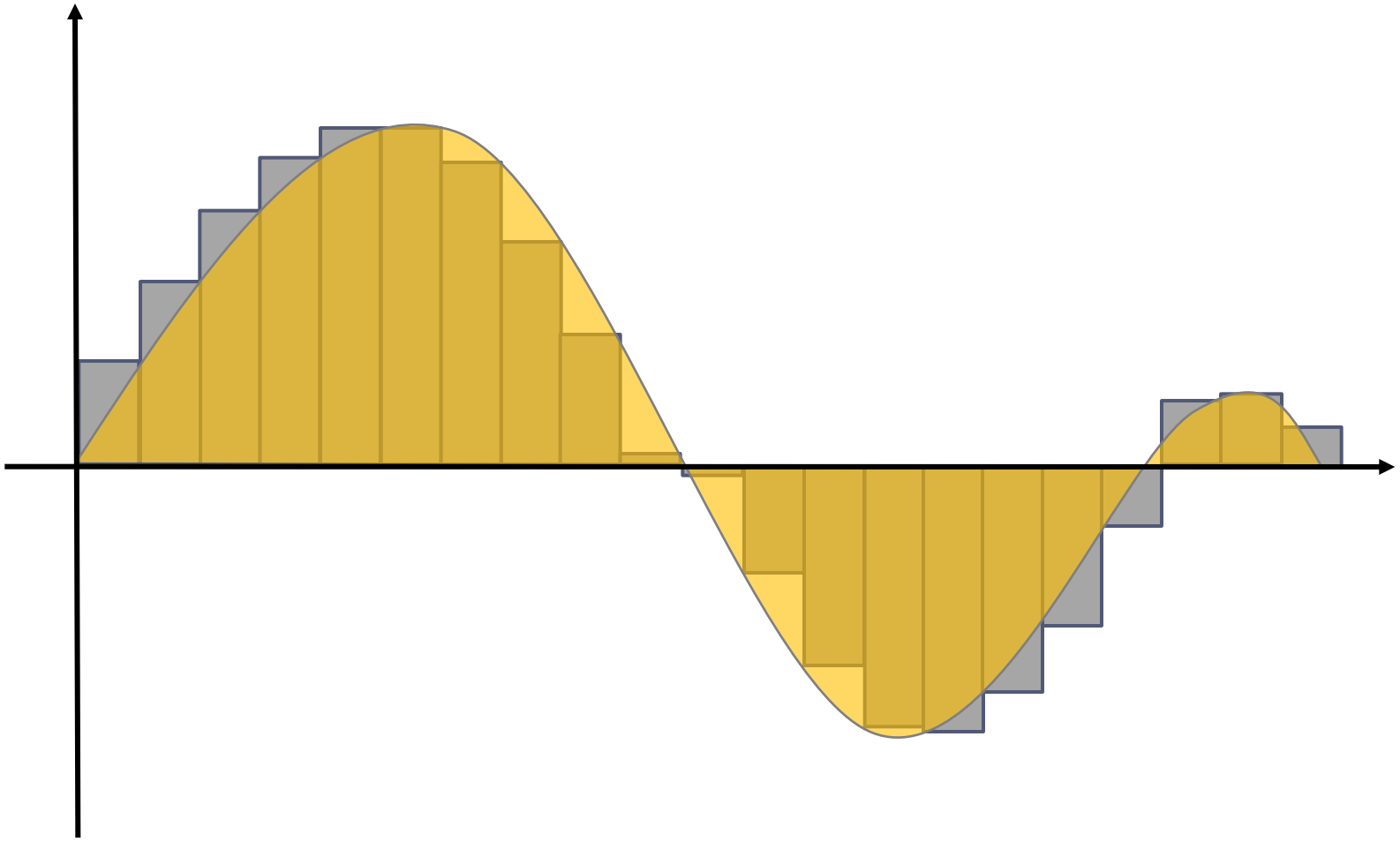


Solução: resolver numericamente.

Integral



Integral

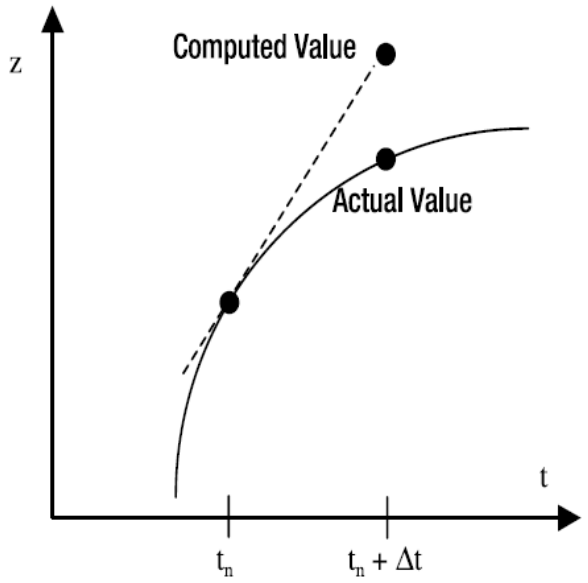


Integração Numérica

- Método de Euler

$$v_{t+\Delta t} \approx v_t + a\Delta t$$

- Simples, porém impreciso: erros acumulam, simulações explodem!
  - Passo da simulação (Δt)
  - Se for muito grande: ???
  - Se for muito pequeno: ???



Integração Numérica

- Método de Euler

$$v_{t+\Delta t} \approx v_t + a\Delta t$$

- Simples, porém impreciso: erros acumulam, simulações explodem!
  - Passo da simulação (Δt)
  - Se for muito grande: **imprecisão**
  - Se for muito pequeno: **custo computacional**

Integração Numérica

- Outras possibilidades
  - Runge-Kutta
    - Equações diferenciais
  - Leapfrog
    - Posição
    - Velocidade

$$r_{n+1} = r_n + v_n \times dt + a_n \times \frac{(dt)^2}{2}$$

$$v_{n+1} = v_n + (a_n + a_{n+1}) \times \frac{dt}{2}$$

Integração Numérica

- Exemplo

x	Y_{Euler}	Y_{RK2}	Y_{RK4}	Y_{EXATA}
0	1,0	1,0	1,0	1,0
0,1	1,1	1,11	1,11034167	1,11034183
0,2	1,22	1,24205	1,24280515	1,24280552

Como fazer então?

- Temos objetos de massas conhecidas
- Sobre eles, vamos aplicar forças conhecidas
  - A Física nos diz como obter <u>acelerações</u>
    - F=ma, então a(t)=F/m
  - Integrando (numericamente) as acelerações, <u>obtemos velocidades</u>
    - v(t+dt) = v(t) + dt * v'(t) = v(t) + dt * a(t)
  - Integrando (numericamente) as velocidades, <u>obtemos as posições</u>
    - p(t+dt) = p(t) + dt * p'(t) = p(t) + dt * v(t)

Posição atual (armazenada)

Passo da simulação

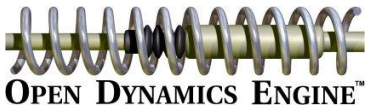
Muito complicado?!?!?

- Para isso, hoje em dia temos bibliotecas bastante completas!
- Motores de Física (*Engines*)

Engines de Física

Engines 3D

http://www.newtondynamics.com



http://www.ode.org

http://www.bulletphysics.com

http://www.impulse-based.de

http://www.nvidia.com/object/physx_new.html

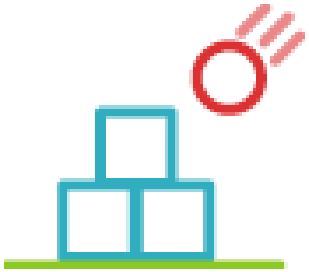
http://www.tokamakphysics.com

http://www.trueaxis.com

http://www.havok.com

Engines 2D

http://code.google.com/p/chipmunk-physics/



http://www.box2d.org/

http://farseerphysics.codeplex.com/

Engines Físicas

- Dinâmica de corpos rígidos
- Dinâmica de corpos flexíveis
- Dinâmica de fluídos

- Tempo real ou não
- Suporte a diferentes linguagens e plataformas
- Tipos de licença (open-source, free, pagas..)

- Provêem toda a parte matemática que comentamos
  - Integração do tempo
  - Cálculos físicos

Engines de Física – Principais passos

1. Inicializar a *engine*
   - Criar uma instância do gerenciador da simulação
2. Setar os parâmetros da *engine*
   - Parâmetros globais como vetor da gravidade, tamanho do passo da simulação
3. Criar a cena
   - Criação dos objetos da cena com seus parâmetros físicos (massa, propriedades dos materiais...)
4. Aplicar força aos objetos
   - Forças iniciais aplicadas aos objetos
5. Iniciar o *loop* da simulação
   - Passo da simulação, que determinará as novas posições dos objetos
6. Finalizar a *engine*
   - Liberar os recursos

Engines Físicas

- E a visualização??
  - *Engines de Física* não são responsáveis pelo desenho dos objetos...

- O que fazer então???
  - Integrar com API's gráficas
    - OpenGL
    - DirectX
    - Ou outras *engines*/bibliotecas gráficas
      - Irrlicht
      - OSG
      - Unity

Uma leitura interessante

- http://www.wildbunny.co.uk/blog/2011/04/06/physics-engines-for-dummies/

- *Post* recente de um desenvolvedor (Paul Firth, Wildbunny)

Tema de Casa

- Para semana que vem
  - Pesquisar 3 títulos de jogos que usem Física e descobrir qual a *engine* que os desenvolvedores utilizaram
    - Não vale os exemplos dos slides :P
    - Enviar pelo moodle um txt com os títulos dos jogos, engine e referência(s) de onde foram tiradas as informações
    - É claro que vale nota!

Conceitos Básicos

Sistemas de Unidades
Notação científica
Notação de somatório
Letras gregas
Sistemas de referência
Grandezas Físicas
Vetores

Sistemas de Unidades

- Para descrever o mundo físico, é necessário **medir** as coisas
- Para medir coisas, é necessário **unidades de medida**
- No Brasil, usamos o Sistema Internacional de Unidades (conhecido como SI) como padrão
  - Quase todo mundo usa, menos EUA e Inglaterra...
- Como boa parte da literatura para desenvolvedores de *games* está em língua inglesa, pode ser que nos deparemos com unidades no sistema de unidades inglês

Sistemas de Unidades

- Unidades e fatores de conversão

Quantity	English Units	SI Units	Conversion Factor
Length	foot (*ft*)	meter (*m*)	0.3048
	mile	kilometer (*km*)	1.609
Mass	pound-mass (*lbm*)	kilogram (*kg*)	0.4536
	slug	kilogram (*kg*)	14.593
Force	pound (*lb*)	Newton (*N*)	4.448
Pressure	lb/in^2	N/m^2	6894.7
Density	$slug/ft^3$	kg/m^3	515.379
	lbm/ft^3	kg/m^3	16.018
Temperature	Fahrenheit (^oF)	Graus Celsius(°C)	°C = (°F − 32) / 1,8
	Rankine (*R*)		°C = K − 273,15

Notação Científica

- No mundo da Física, às vezes precisamos trabalhar com números muito grandes ou muito pequenos
  - Ex.: constante gravitacional G (uma quantidade que relaciona a força que dois objetos exercem um sobre o outro)

$$G = 0.0000000000667 \quad \frac{N - m^2}{kg^2}$$

  - Notação científica serve para expressar esses números de forma mais compacta, utilizando números entre 0 e 10, a letra "e" e um número que representa em qual potência de 10 o primeiro valor está

$$G = 6.67e-11 \quad \frac{N - m^2}{kg^2}$$

Notação Científica

- Você vai precisar usar notação científica quando for incorporar Física em seus programas
- Linguagens como C/C++, C# e java reconhecem esse tipo de notação

```
double G = 6.67e-11;
```

Notação de Somatório

- Muitas vezes precisamos somar uma seqüência de números
  - Ex.: somar a massa total de 5 objetos
    - $m_{total} = m_1 + m_2 + m_3 + m_4 + m_5$

    Uma maneira mais fácil de se expressar é utilizando a notação de somatório:

    - $$m_{total} = \sum_{j=1}^{5} m_j$$

Notação de Somatório

- E como se programa isso???
  - Assumindo que temos um *array* **mass**[] com as massas dos objetos que queremos somar:

```
massTotal = 0.0;
for(j=0; j<5; ++j) {
  massTotal += mass[j];
}
```

Letras Gregas

- Matemáticos ADORAM usar letras gregas em suas equações
- E bom, Matemática é a linguagem da Física...
- Pense que são símbolos, que normalmente irão denotar uma variável, uma constante ou uma operação matemática

Ex.: $d = at^2 + bt + c$

pode ser escrita como

$$\delta = \alpha t^2 + \beta t + \chi$$

Aliás, a notação de somatório $\sum$ é a letra grega sigma maiúscula! ☺

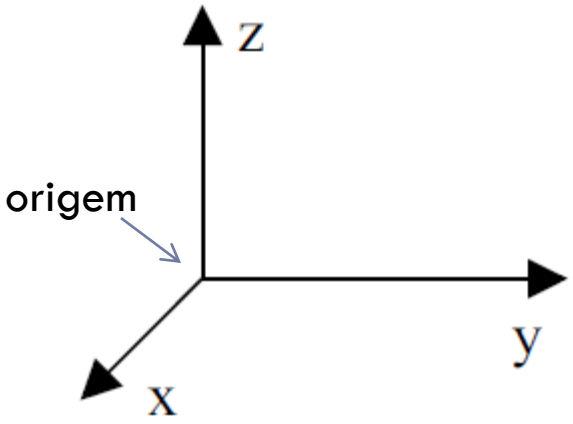
http://pt.wikipedia.org/wiki/Alfabeto_grego

Sistemas de Coordenadas

- A maneira utilizada para especificar a localização de objetos/pontos no espaço 2D ou 3D
- Define o referencial para um objeto
- Os dois sistemas de coordenadas mais utilizados são
  - Sistema de coordenadas cartesianas
  - Sistema Esférico de coordenadas

Sistema de Coordenadas Cartesiano

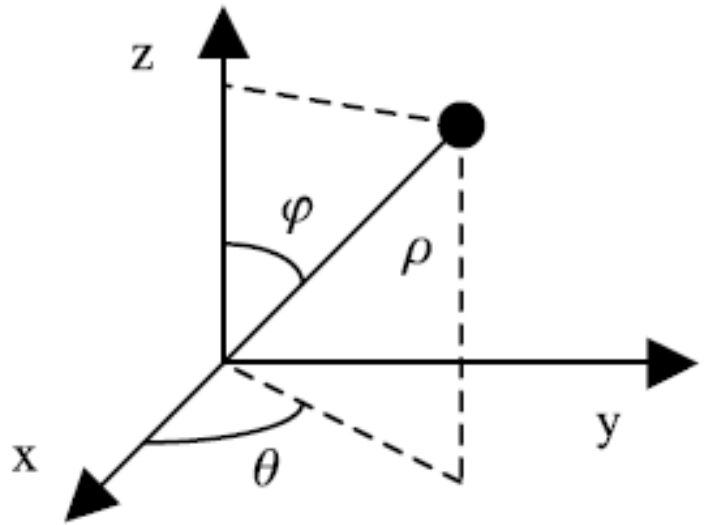
- Define a localização de um objeto especificando-se as localizações nos eixos perpendiculares x, y e z
- Normalmente, z é para cima (convenção)
- É o mais utilizado na programação de jogos
- É o mais fácil de se compreender



- Porém, ele não é muito adequado para se descrever movimentos circulares (rotações...)

Sistema Esférico de Coordenadas

- Define a posição de um objeto pela distância ρ (rho) de uma origem e dois ângulos θ (theta) e φ (fi)



- Este sistema é útil para descrever movimentos circulares ou rotações

Sistema Esférico de Coordenadas

- Como visto até agora, cada sistema de coordenadas provêem uma maneira de se representar a posição de um objeto no espaço
- Dependendo as operações, é mais útil usar um ou outro sistema
- É possível fazer a conversão de um sistema para outro
- Para converter um ponto em coordenadas esféricas (ρ, θ, φ) em coordenadas cartesianas (x,y,z), utilizamos as seguintes equações:

$$x = \rho \sin\varphi \cos\theta$$

$$y = \rho \sin\varphi \sin\theta$$

$$z = \rho \cos\varphi$$

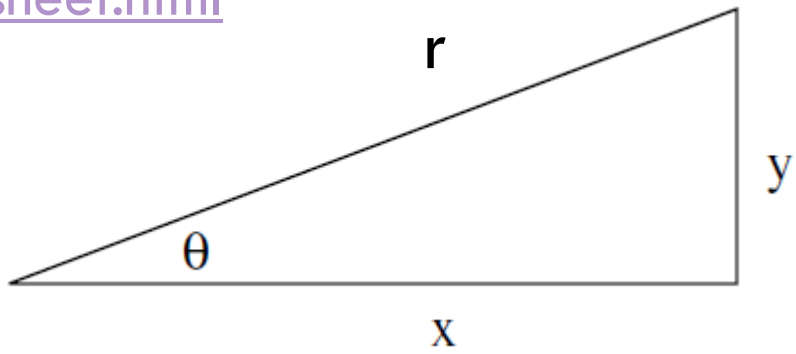
$$\rho = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = arctan\frac{y}{x}$$

$$\varphi = arctan\frac{\sqrt{x^2 + y^2}}{z}$$

Sistema Esférico de Coordenadas

- Daonde isso?!?
  - Trigonometria, teorema de Pitágoras!
    - http://www.dummies.com/how-to/content/trigonometry-for-dummies-cheat-sheet.html



$\sin \theta = y/r$ $\quad x = r \cos \theta = y/\tan \theta$ $\quad \sin^{-1}(y/r) = \theta$

$\cos \theta = x/r$ $\quad y = r \sin \theta = x \tan \theta$ $\quad \cos^{-1}(x/r) = \theta$

$\tan \theta = y/x$ $\quad r = y/\sin \theta = x/\cos \theta$ $\quad \tan^{-1}(y/x) = \theta$

Sistema Esférico de Coordenadas

- Quando lidamos apenas com 2 dimensões (2D), usamos as coordenadas polares (ρ, θ)

$$x = r\cos\theta \qquad \theta = arctan\frac{y}{x}$$

$$y = r\sin\theta \qquad \rho = \sqrt{x^2 + y^2}$$

  - Rotinas de matemática do C++ fornecem ângulos em radianos
  - Conversão de graus em radianos e radianos em graus??
    - Regrinha de 3 ☺

$$180\ graus = 3{,}1416\ radianos$$

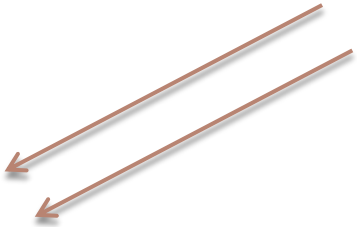
Grandezas Físicas

- Grandezas escalares: são bem definidas por um número e uma unidade de medida.
  - Ex: temperatura, altura, distância percorrida, pressão

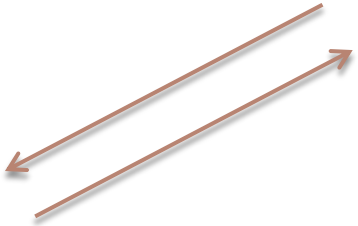
- Grandezas vetoriais: além de um número e uma unidade de medida necessitam de uma direção e um sentido para serem bem definidas.
  - Ex: velocidade, aceleração, força, campo elétrico.

Grandezas vetoriais:

- Representadas por uma seta
  - Tamanho: módulo ou intensidade
  - Direção: reta suporte
  - Sentido: seta
- Mesma direção e sentido:



- Mesma direção e sentidos contrários:

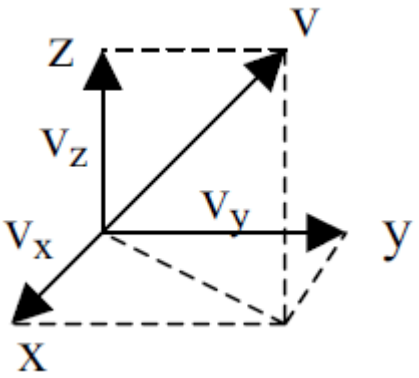


Grandezas vetoriais

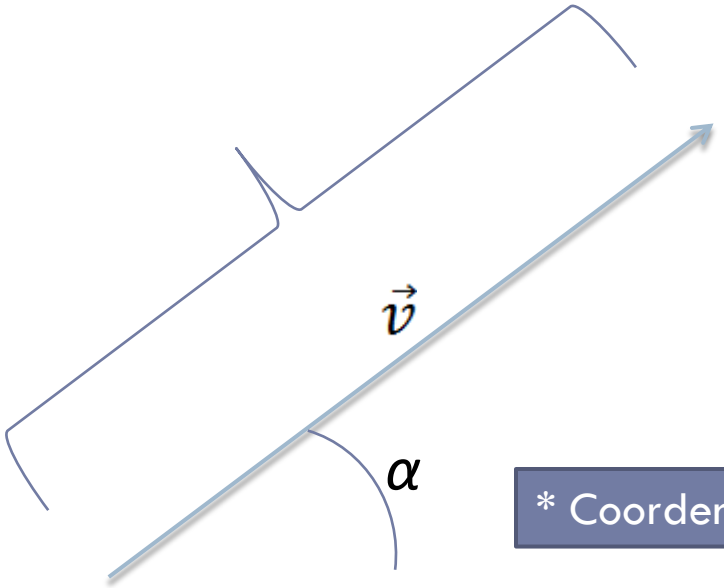
- Por exemplo, sob o sistema de coordenadas cartesianas, a velocidade de um objeto pode ser dividida nos componentes x, y e z

$$\vec{v} = v_x\vec{x} + v_y\vec{y} + v_z\vec{z}$$

- Os termos v_x, v_y e v_z são os componentes da velocidade nas direções x, y e z



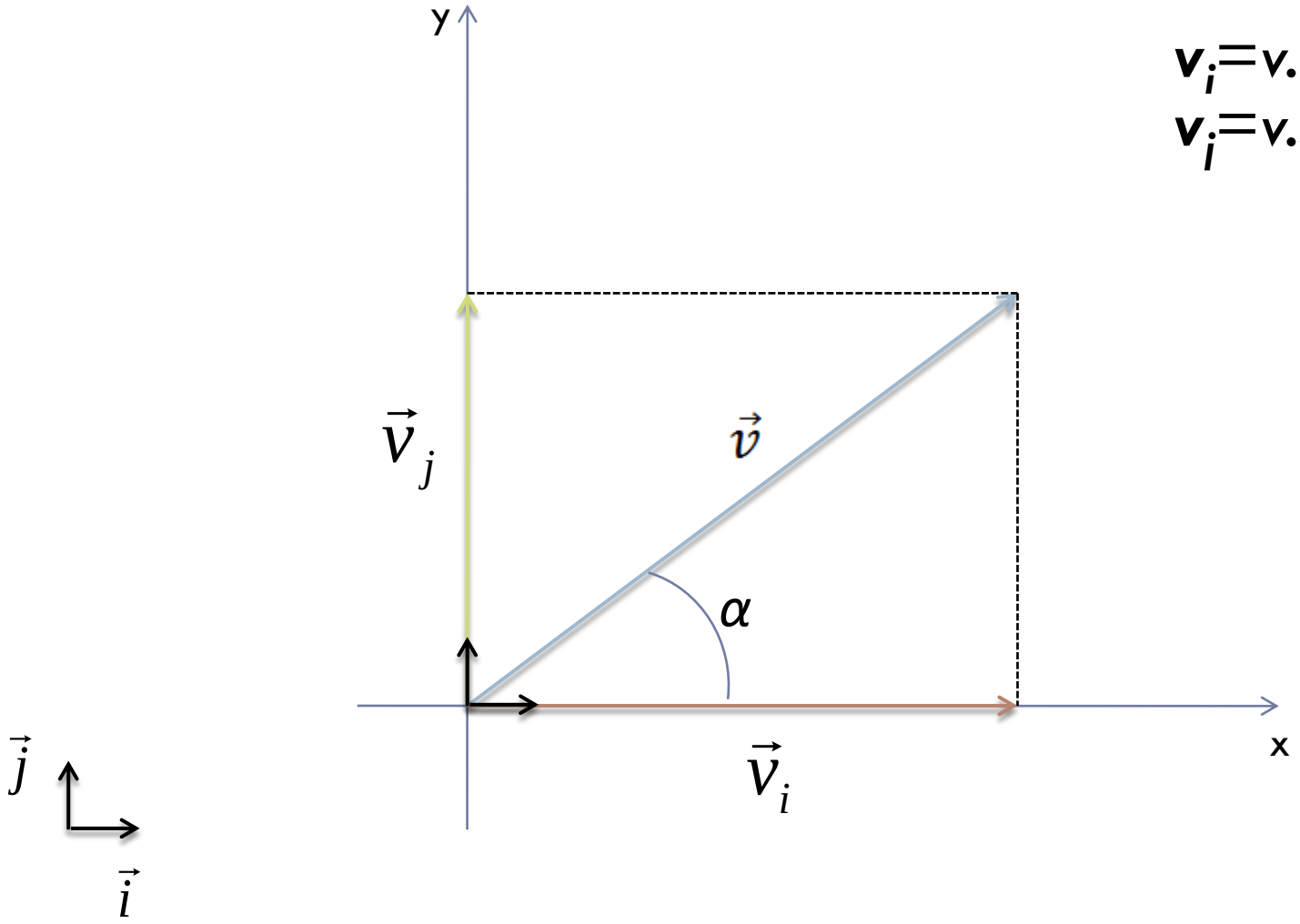
Como trabalhar com vetores?



* Coordenadas polares

Coordenadas retangulares

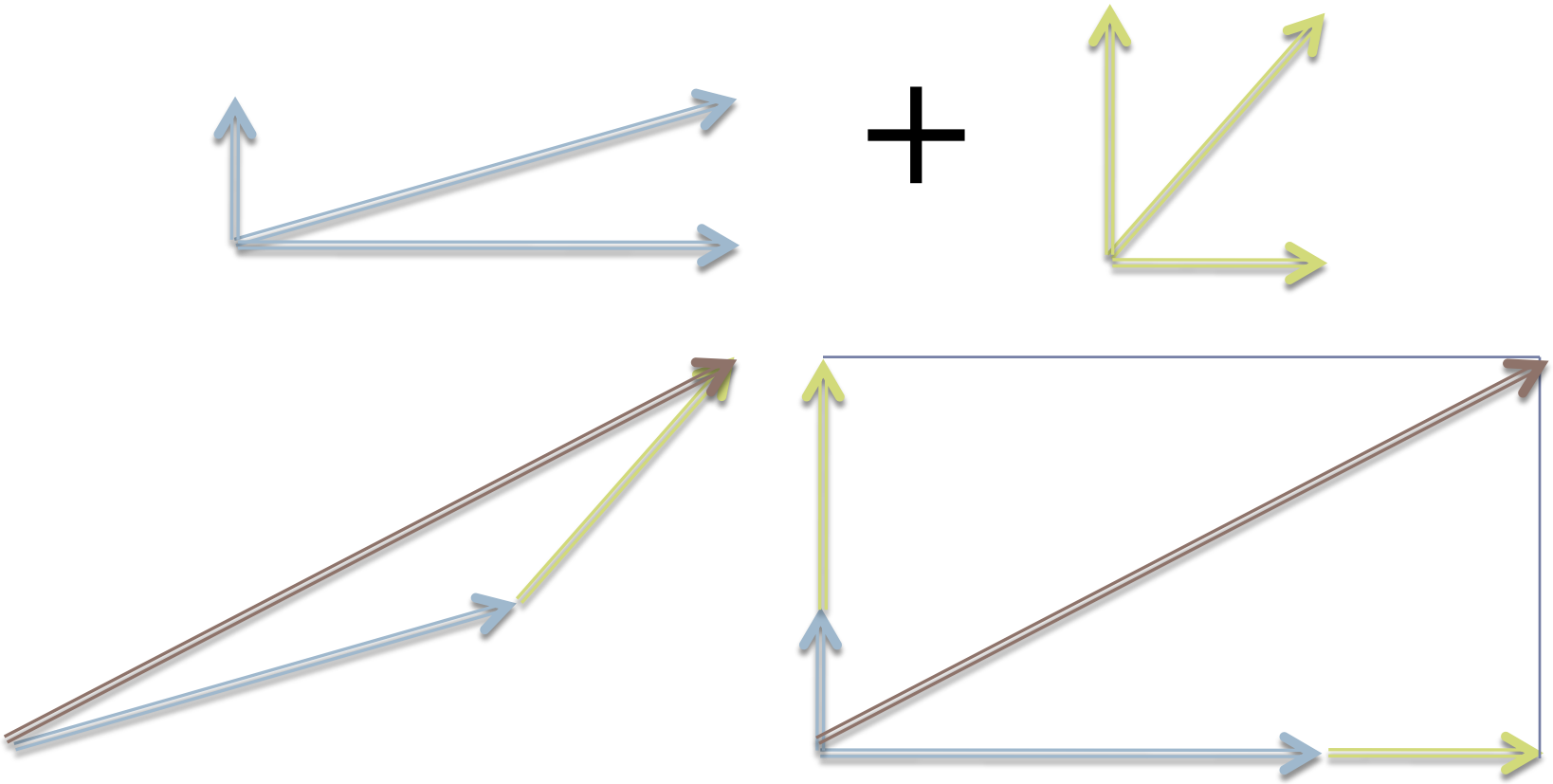
Decomposição de vetores em componentes ortogonais



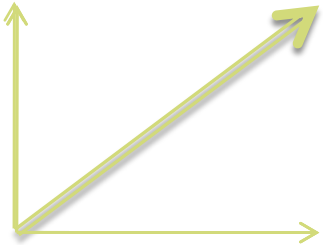
$\mathbf{v_i} = v.\cos\alpha\ \mathbf{i}$

$\mathbf{v_i} = v.\mathrm{sen}\alpha\ \mathbf{j}$

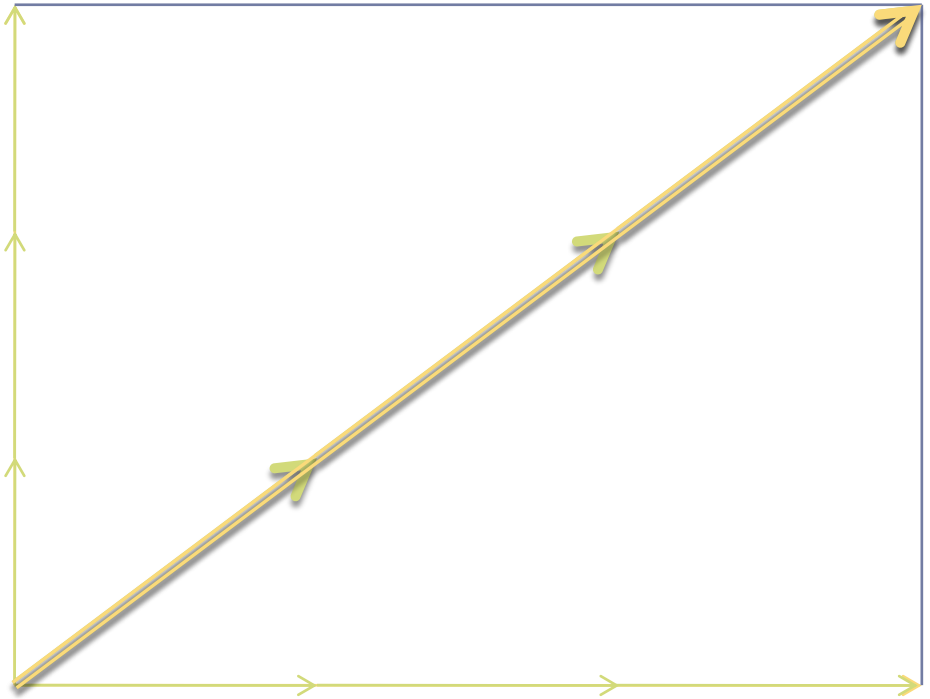
Soma de Vetores



Multiplicação por escalar



X 3



Módulo e normalização de um vetor

- O módulo (comprimento) do vetor $\mathbf{v} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$ é:

$$\left|\vec{v}\right| = \sqrt{a^2 + b^2 + c^2}$$

- Para encontrar o vetor unitário (de módulo 1) na direção de **v**, faz-se:

$$\vec{u}_v = \frac{a}{\left|\vec{v}\right|}\vec{i} + \frac{b}{\left|\vec{v}\right|}\vec{j} + \frac{c}{\left|\vec{v}\right|}\vec{k}$$

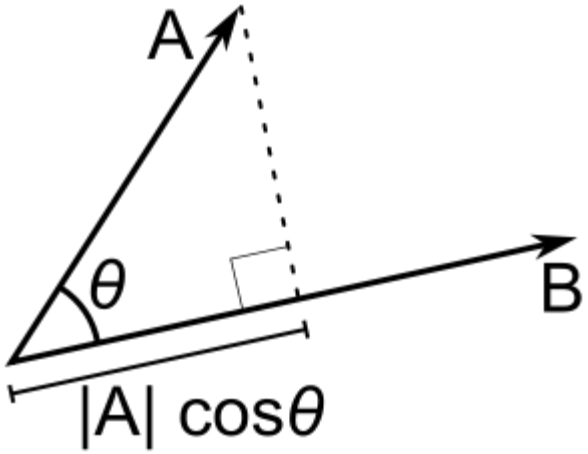
- Este processo chama-se *normalização* de um vetor (encontrar o vetor unitário que o representa)

Produto escalar

- $\boldsymbol{A} = a_1\boldsymbol{i} + a_2\boldsymbol{j}$ $\qquad\qquad$ $\boldsymbol{B} = b_1\boldsymbol{i} + b_2\boldsymbol{j}$

$$\mathbf{A} \bullet \mathbf{B} = a_1 b_1 + a_2 b_2$$

$$\mathrm{A} \bullet \mathrm{B} = \|A\|\|B\| \cos\theta$$



O produto escalar de dois vetores A e B é o resultado do produto do comprimento de Apela projeção escalar de B em A.

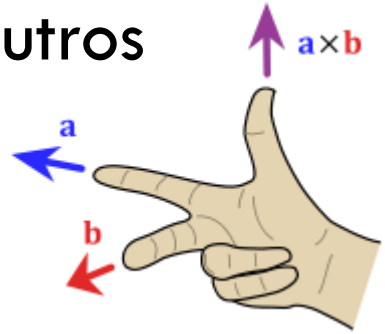
Uma aplicação interessante

- Ângulo entre 2 vetores:
  - Mais importante aplicação do produto escalar
- O cosseno do ângulo entre dois vetores é o produto escalar destes vetores normalizados

$$\theta = \arccos \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|}$$

Produto vetorial

- A partir do produto vetorial, encontramos o vetor que é *normal* ou perpendicular a dois outros vetores:



$$\vec{a} = a_x\vec{x} + a_y\vec{y} + a_z\vec{z}$$

$$\vec{b} = b_x\vec{x} + b_y\vec{y} + b_z\vec{z}$$

$$\vec{c} = \left(a_y b_z - b_y a_z\right)\vec{x} + \left(a_z b_x - b_z a_x\right)\vec{y} + \left(a_x b_y - b_x a_y\right)\vec{z}$$