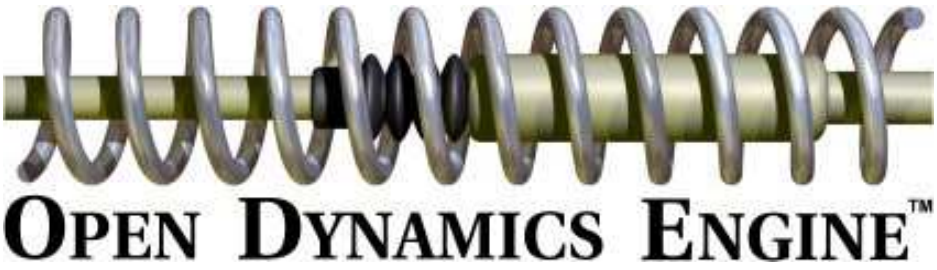


A biblioteca de simulação física ODE

Denis Fernando Wolf	USP - Universidade de São Paulo - ICMC
Eduardo do Valle Simões	LRM – Laboratório de Robótica Móvel
Fernando Santos Osório	INCT – SEC: Sistemas Embarcados Críticos
Gustavo Pessin	SENA – Sistema Embarcado de Navegação Autônoma
Kalinka R.L.J. Castelo Branco	*FOG – The Fellowship Of the Game*

Laboratório de Robótica Móvel – LRM
Instituto de Ciências Matemáticas e de Computação – ICMC
Universidade de São Paulo – USP

Cronograma

- Importância da simulação realística
- Funcionamento conceitual da ODE
- Exemplos: código fonte C com ODE
  - Morfologia (ambiente, agentes)
  - Navegação e colisão
- Exemplos de ferramentas que utilizam a ODE
  - Juice
  - Simulator Bob

FramesPerSecond: 111

Importância da simulação realística

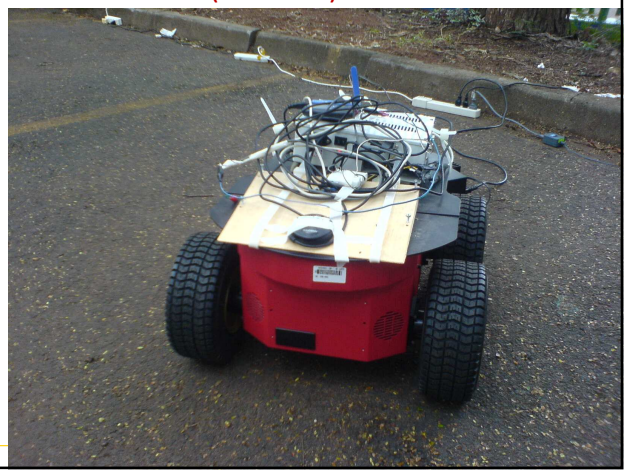
- Experimentos em robótica móvel podem ser realizados de duas formas: diretos em um robô real ou em um robô simulado em um ambiente virtual (realista?).

Robô real = {tempo++, $$++}

A simulação é especialmente útil para robôs caros, grandes ou frágeis. (Eliminando desperdícios)

Porém, para que uma simulação seja realmente útil, ela deve capturar o máximo possível de características reais do sistema desenvolvido.

Como capturar estas características?

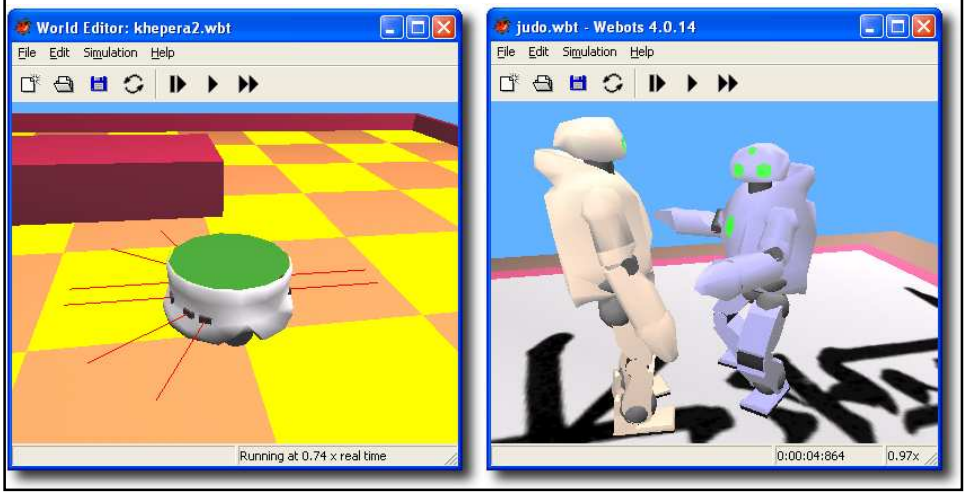


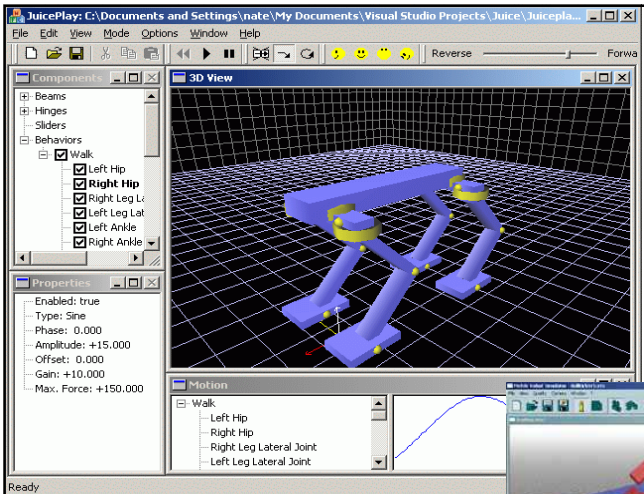
Importância da simulação...

- Quando usar uma ferramenta pronta? Quando usar um conjunto de bibliotecas?
  - **Depende das necessidades da sua aplicação!**
    - Realismo na modelagem robótica...
    - Realismo na interação de robôs com o ambiente (outros robôs, árvores, terrenos, etc...)
    - Comunicação entre os robôs?
    - Controle nos robôs? (Regras, Redes Neurais Artificiais)
    - Organização de sistemas multirrobóticos? (Regras, Algoritmos Genéticos)
    - Características ambientais... (Incêndio, Exploração)
    - Plataforma (S.O.)?

Ferramentas...

- Webots... proprietário $$... Simulação ambiental?... Programação de sistemas multirrobóticos (colaboração/cooperação)? [**Usa ODE!**]

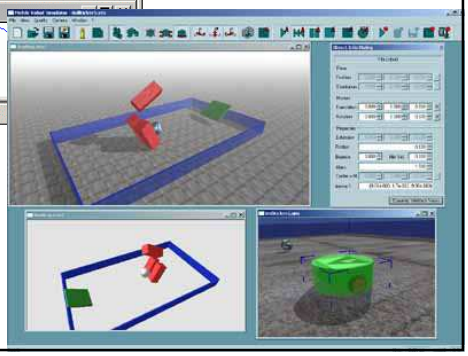




Juice

- Livres (meio abandonados)...
- Só para Win...
- Simulação ambiental?...
- Programação de sistemas multirrobóticos? (colaboração/cooperação)?
- **Usam ODE!**

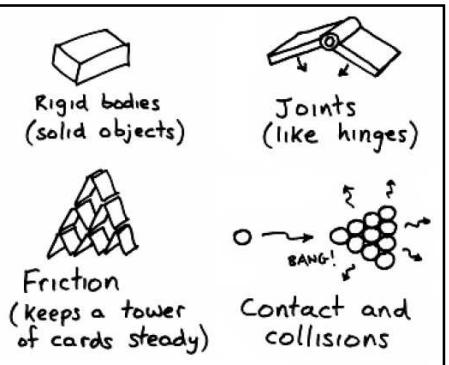
Simulator Bob



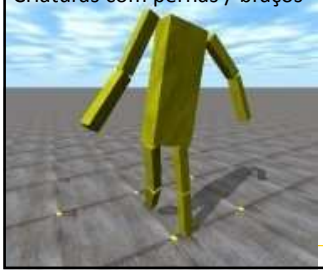
- Arquivos XML...
- Aproveitamento de Código?

Open Dynamics Engine (ODE)

- Biblioteca para simulação de dinâmica de corpos rígidos articulados.
- Uma estrutura articulada é criada quando corpos rígidos são conectados por algum tipo de articulação (junção)



Criaturas com pernas / braços

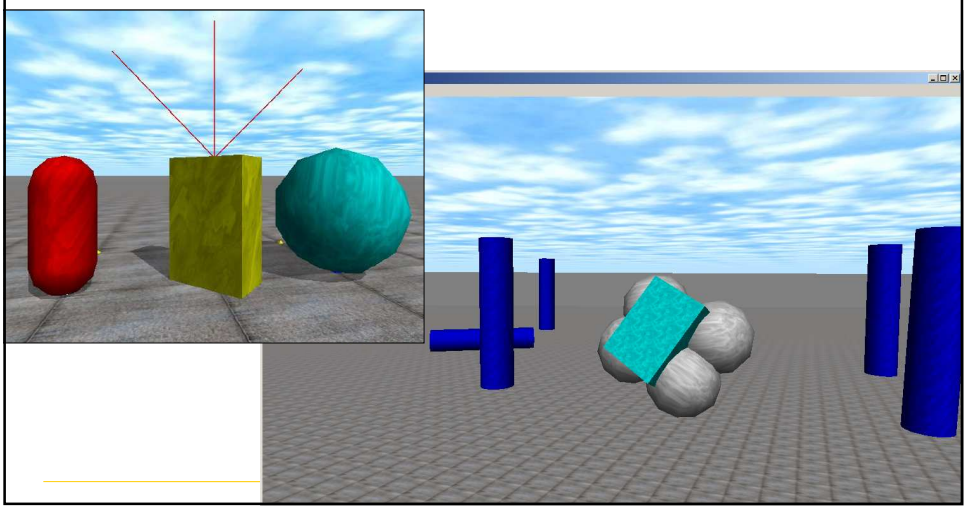


Open Dynamics Engine (ODE) :: Características

- Desenvolvida para simulação de tempo real
- Rápida, robusta e estável (acuracidade--).
- Licença BSD ou GPL
- Has a native C interface (even though ODE is mostly written in C++).

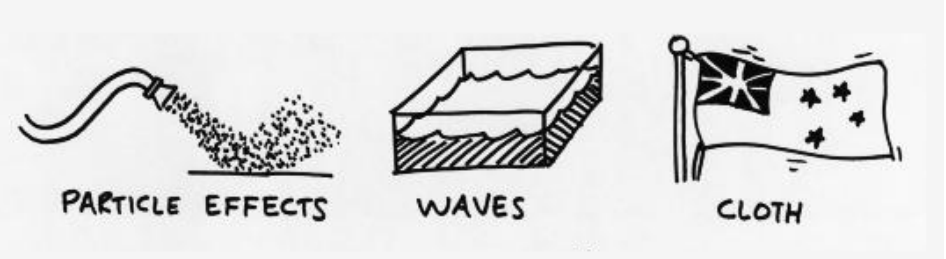
Open Dynamics Engine (ODE) :: Características

- The current collision primitives are sphere, box, capped cylinder, plane, ray, and triangular mesh (more collision objects will come later)



Open Dynamics Engine (ODE) :: Características

- A ODE não tem como objetivo realizar simulação de outras dinâmicas além da de corpos rígidos, como partículas, roupas, ondas, fluídos, corpos flexíveis ou fraturas.



Instalação e Uso

- Compile para a sua plataforma ☺ [Linux, Win, MacOS]

- Segundo o autor da biblioteca:
  - “The best way to understand how to use ODE is to look at the test/example programs that come with it.”

```
#include <ode/ode.h>
```

Open Dynamics Engine (ODE)

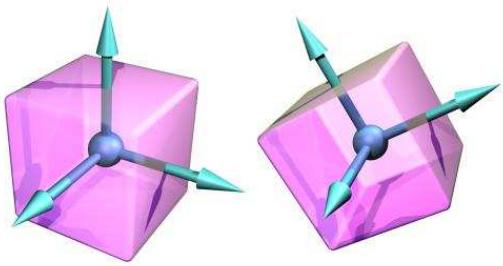
- Do ponto de vista físico, um robô pode ser considerado simplesmente como um conjunto de **corpos rígidos conectados**. Cada um destes corpos pode interagir com os demais, assim como um motor faz com que um veículo se movimente.

- Além disso, a atuação da gravidade e da inércia devem estar presentes.

- Na ODE, estas propriedades são tratadas em dois conceitos:
  - **Corpos rígidos** e **articulações**.

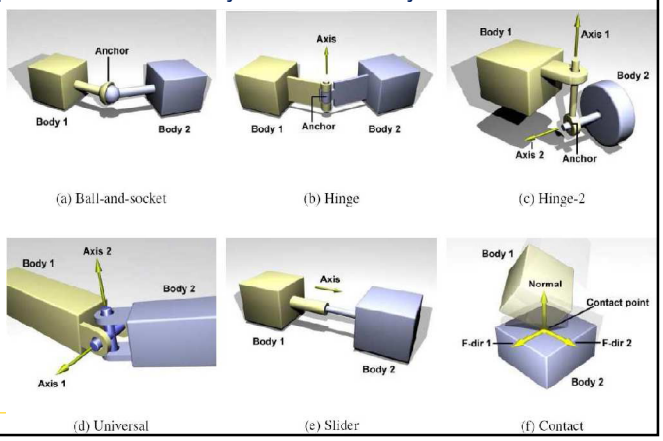
Corpos Rígidos

- Suas propriedades, do ponto de vista da simulação:
  - Vetor de posição (x,y,z)
  - Velocidade linear, representada como um vetor (vx,vy,vz)
  - Orientação do corpo, representada como um quaternion (qs;qx;qy;qz) ou uma matriz de rotação
  - Velocidade angular (mudança de orientação com o passar do tempo)
  - Massa do corpo;
  - Posição centro de massa
  - Matriz de inércia (descreve como a massa do corpo está distribuída ao redor do centro de massa)



Articulações

- As articulações são relacionamentos entre dois corpos de modo que possam existir posições e orientações que sejam relativas a ambos os corpos.
- Este relacionamento realiza certos tipos de restrições/obrigações de movimento.
- A cada passo de simulação, o integrador aplica num corpo uma força de atuação, mas move este corpo preservando as restrições das articulações.



Cada tipo de junta tem um conjunto específico de parâmetros!

Código típico de uma simulação

1. Criação do mundo;
2. Criação dos corpos, no mundo;
3. Ajuste da posição e orientação dos corpos;
4. Criação de articulações, no mundo;
5. Conexão das articulações nos corpos;
6. Ajuste dos parâmetros das articulações;
7. Repetir, para cada passo de simulação:
    (a) Aplicar forças nos corpos como necessário;
    (b) Ajustar os parâmetros das articulações;
    (c) Chamar rotina de teste de colisão;

DrawStuff

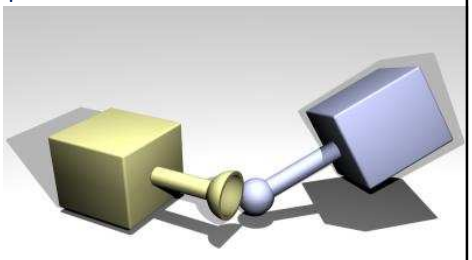
- **A ODE é completamente independente de visualizador!**
  - Iniciar a criação dos corpos e das simulações pode ser uma tarefa bastante árdua caso não tenhamos uma forma simples e fácil de visualizar os objetos.
  - Por este motivo, a biblioteca DrawStuff é disponibilizada em conjunto com a ODE.
  - Basicamente, o DrawStuff é um ambiente de visualização de objetos 3D que tem o propósito de permitir a demonstração visual da ODE.
  - Muito, muito, muuuuuuuuito amigável.

Integração

- Na ODE, o **processo de simulação** dos corpos rígidos **através do tempo** é chamado de integração.
- Cada passo de integração avança no tempo um dado step size, ajustando o estado dos corpos rígidos.
- **Questões**
  - **Quão próxima da realidade é a simulação (acuracidade)?**
  - **Quão estável é (irá calcular erros que causem comportamento completamente não físico)?**
- **ODE's current integrator is very stable, but not particularly accurate unless the step size is small.** For most uses of ODE this is not a problem -- ODE's behavior still looks perfectly physical in almost all cases.
- However ODE should not be used for quantitative engineering!

Error Reduction Parameter (ERP)

- **When a joint attaches two bodies, those bodies are required to have certain positions and orientations relative to each other.**
- **Joint error** can happen in two ways:
  - User sets the position/orientation of one body without correctly setting the position/orientation of the other body.
  - During the simulation, errors can creep in that result in the bodies drifting away from their required positions.
- **There is a mechanism to correct joint error: during each simulation step each joint applies a special force to bring its bodies back into correct alignment. This force is controlled by the error reduction parameter (ERP), which has a value between 0 and 1.**



- Example of error in a ball and socket joint (where the ball and socket do not line up).

Constraint Force Mixing (CFM)

- **A maioria das juntas/restrições são "hard".**
  - Não existe penetração de objetos! ["metal duro"]
  - Condições que nunca são violadas!
  - *The ball must always be in the socket*.
- How simulate softer materials? [**allowing some natural penetration of the two objects when they are forced**]
- If CFM is set to zero, the constraint will be hard.
- If CFM is set to a positive value, constraint will be soft.

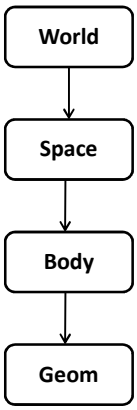
Collision Handling

- **Collisions between bodies are handled as follows:**
  - **First: A collision is detected by the bodies geoms.**
  - **Next: Forces are applied to both bodies.**
- Antes do passo da simulação, a função de detecção de colisão cria uma lista de pontos de contato.
- Cada ponto de contato tem uma posição no espaço.
- Uma "junção especial" é criada em cada ponto de contato, reunindo informação extra, como fricção, quão hard/soft é, e várias outras propriedades.
- Considerando que a velocidade da simulação cai de acordo com o número de pontos de contato, algumas estratégias são utilizadas para limitar o número de pontos de contato.
- Um passo de simulação é dado.
- As junções especiais são removidas.

Contact points
- When two boxes collide many contact points may be needed to properly represent the geometry of the situation, but we may choose to keep only three.

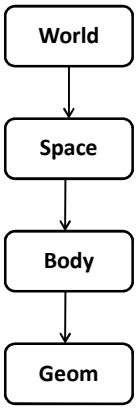
Objects

- dWorld
  - Is a container for rigid bodies and joints. Objects in different worlds can not collide.
- dSpace
  - É uma geometria que contém outras geometrias, na prática, serve para otimizar a detecção de colisão.



Objects

- dBody
  - Rigid body
  - Size, Mass, Position, Velocidades
- dGeom
  - Geometry (for collision)
  - Are the fundamental objects in the collision system.
- dJoint
  - joint
- dJointGroup
  - Na prática, serve para criação e destruição de grandes conjuntos de junções (função de detecção de colisão)



“Motor”

```
for (int i=0;i<QTD_RODAS;i++)
{

dJointSetHinge2Param(veiculo.roda[i],
                     dParamVel,
                     veiculo.dGiro);

dJointSetHinge2Param(veiculo.roda[i],
                     dParamVel2,
                     veiculo.dTorque);

}
```

Criando um cilindro

```
void Obstaculo::GenesisCilindro(dWorldID World, dSpaceID
Space)
{
   dReal x = float(rand()%50)+10;
   dReal y = float(rand()%50)+10;
   dMass m;

   Raio = 1.0;
   Massa = 20.0;
   Altura = 9.0;

   Body = dBodyCreate (World);
   dMassSetSphere (&m,1,Raio);
   dMassAdjust (&m,Massa);
   dBodySetMass (Body,&m);
   Geom = dCreateCylinder(Space, Raio, Altura);
   dGeomSetBody (Geom,Body);
   dBodySetPosition (Body,x,y,Altura);
}
```

Exemplos de usuários [+ de 100 listados]


Elite Heli Squad


BloodRayne2


Quest 3D


Dreampainters


Flight Simulator


Rally World


Motor Sport


Shattera


Ragdoll Matrix


Stalker


Amsterdam Taxi Madness


Monster 4x4

Biliografia

- Página do projeto: http://www.ode.org
- *OPEN DYNAMICS ENGINE USER GUIDE*
  http://www.ode.org/ode-latest-userguide.pdf
- Muitos outros docs em
  http://www.ode.org/ode-docs.html

Vamos ver a ODE funcionando...

- Em “C+-”
  - Navegação e colisão... 1 veículo... 40 veículos...
  - Morfologia... rodinhas... rodões...
  - Navegação sem colisão...
  - Variação do step size...
  - Variação do CFM...

- Exemplo Juice
- Exemplo Simulator BOB