Realidade Virtual: Aumentando ainda mais o realismo

Mestrado em Computação Aplicada
PIPCA

Dr. Fernando S. Osório
fosorio@unisinos.br

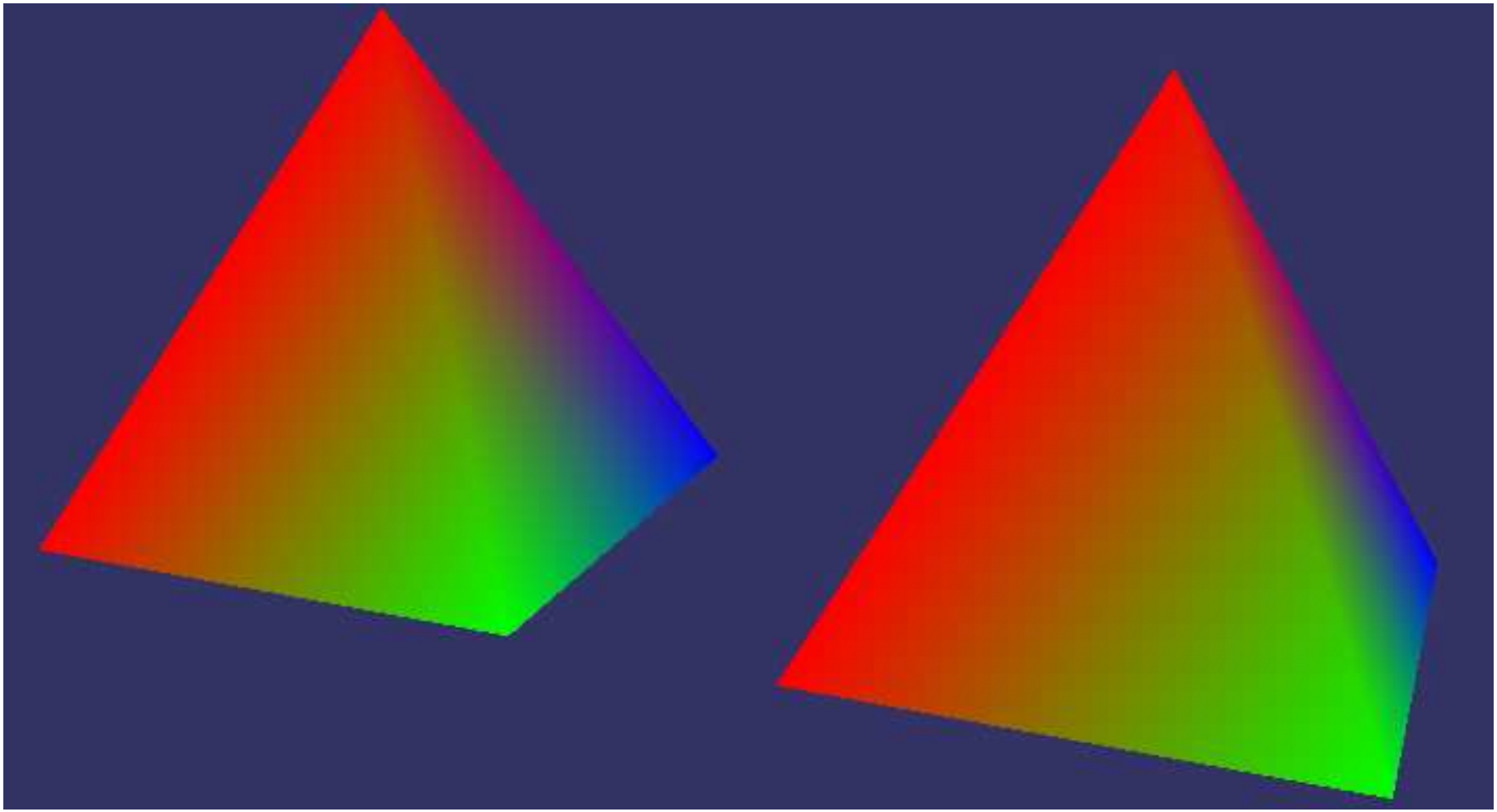
Milton Roberto Heinen
mheinen@turing.unisinos.br

Open Scene Graph

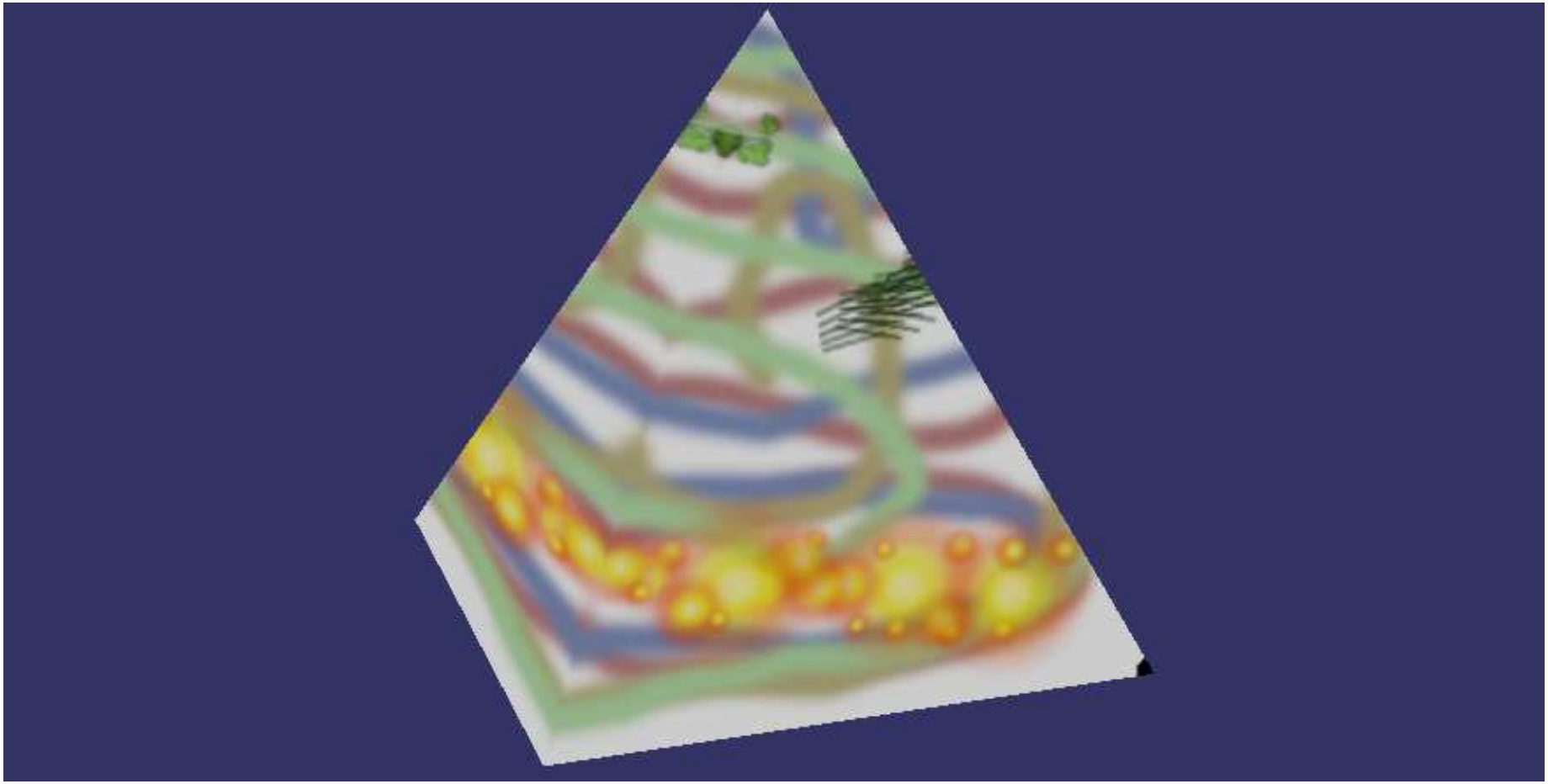
Open Scene Graph - OSG

- OSG é uma biblioteca de software aberta, gratuita e totalmente orientada a objetos em C++
- A biblioteca OSG é uma camada sobre a OpenGL, que permite a montagem de uma cena visual com poucos comandos
- Utilizando OSG, não é necessário utilizar bibliotecas para a criação de uma janela (como MFC, GTK ou glut)
- Também é possível ler arquivos de diversos formatos (vrml, osg, obj, etc) e utilizar texturas, luzes e outros efeitos
- A biblioteca OSG funciona tanto em Windows quanto em Linux – isto facilita o desenvolvimento de aplicações portáveis

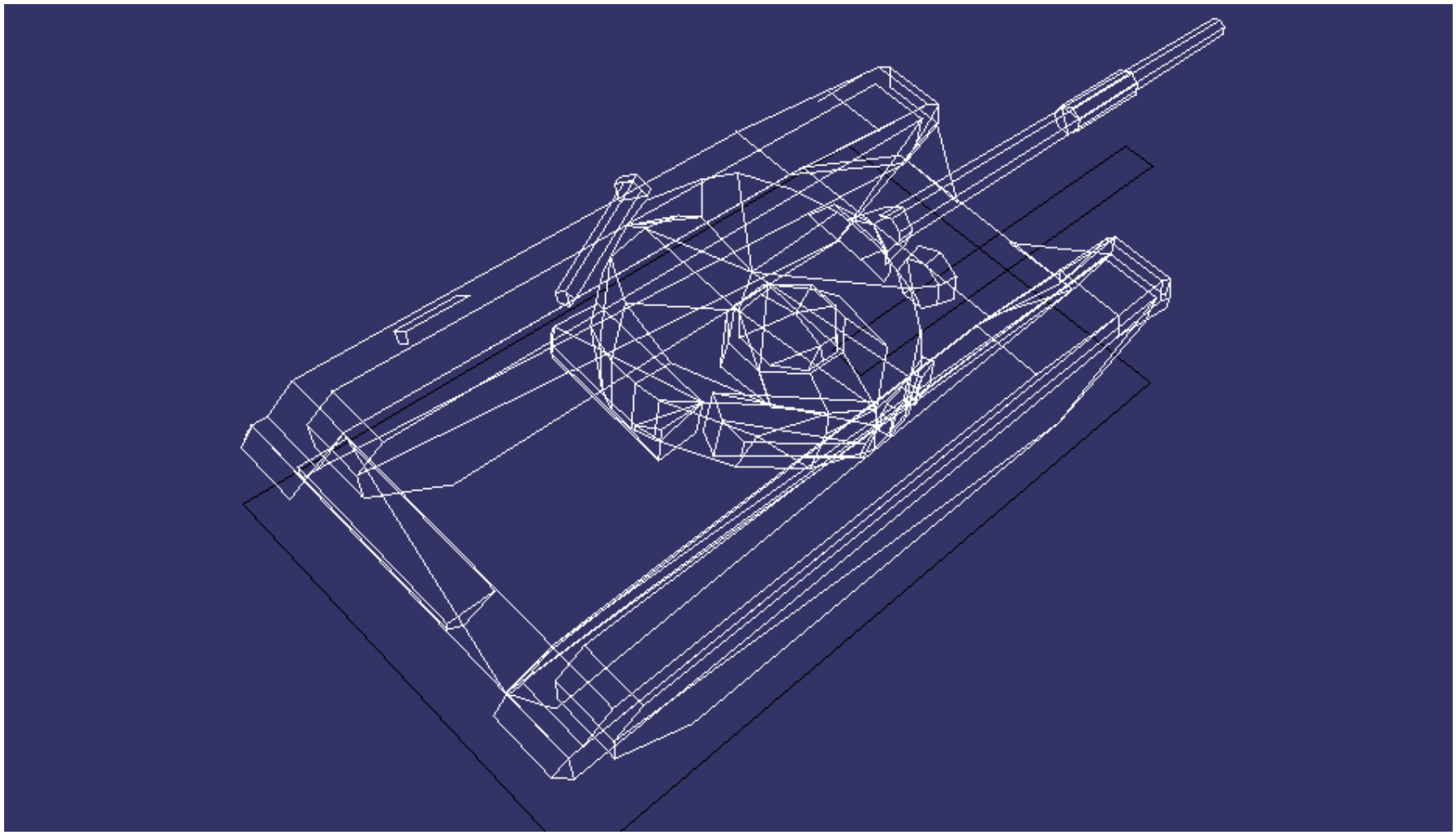
OSG – Objetos primitivos



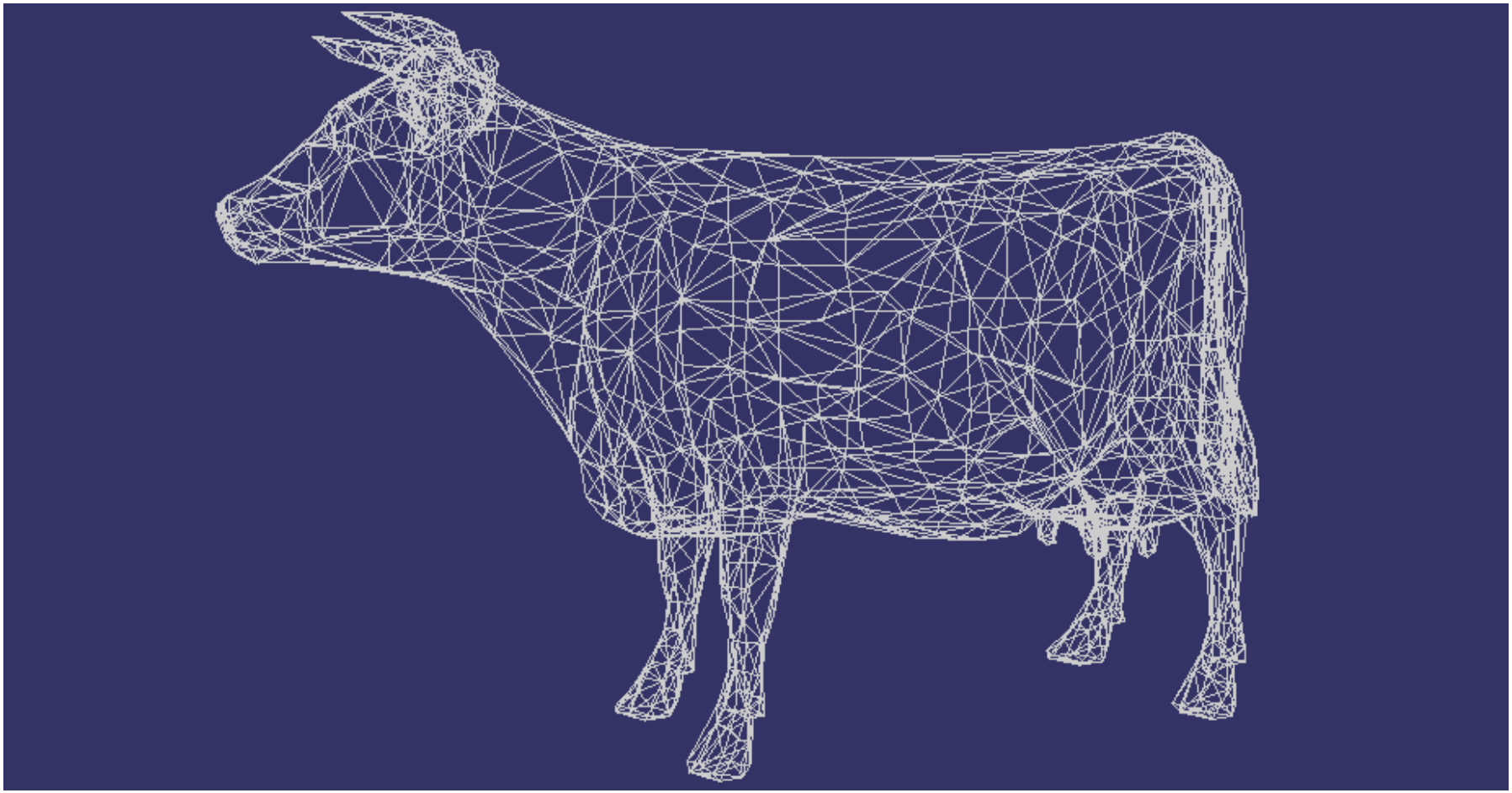
OSG – Texturas



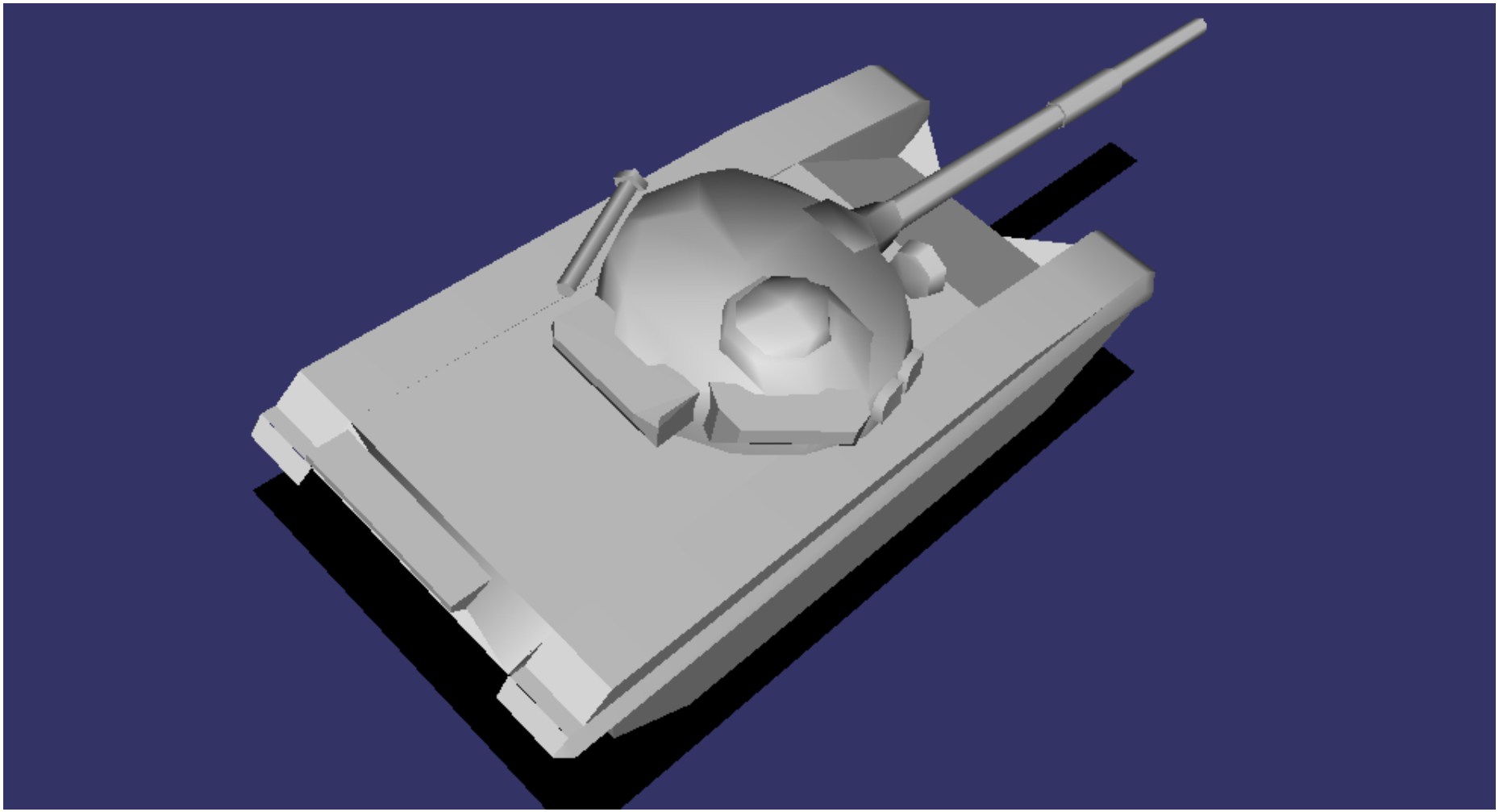
OSG – Wire frame



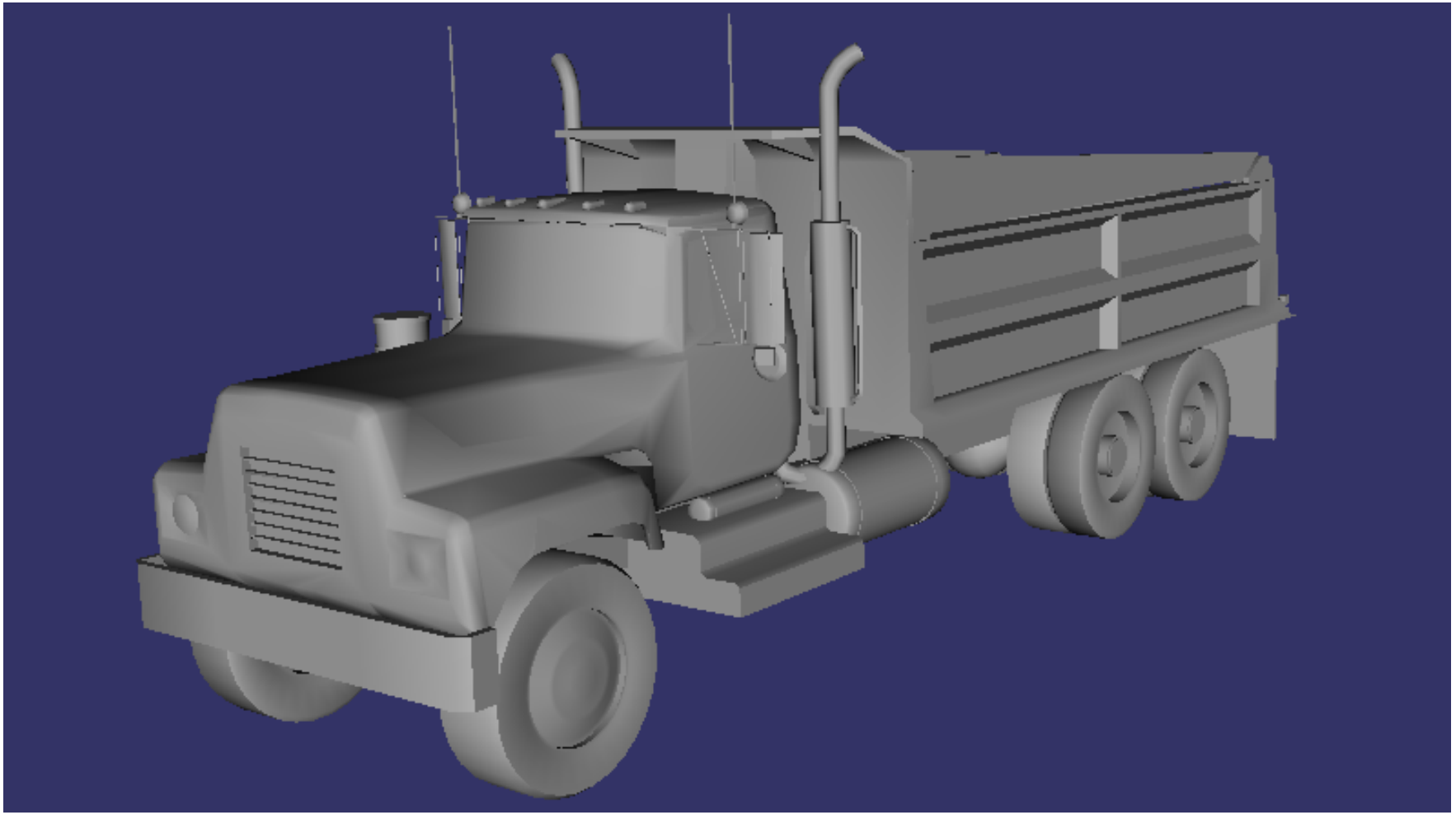
OSG – Wire frame



OSG – Faces



OSG – Faces



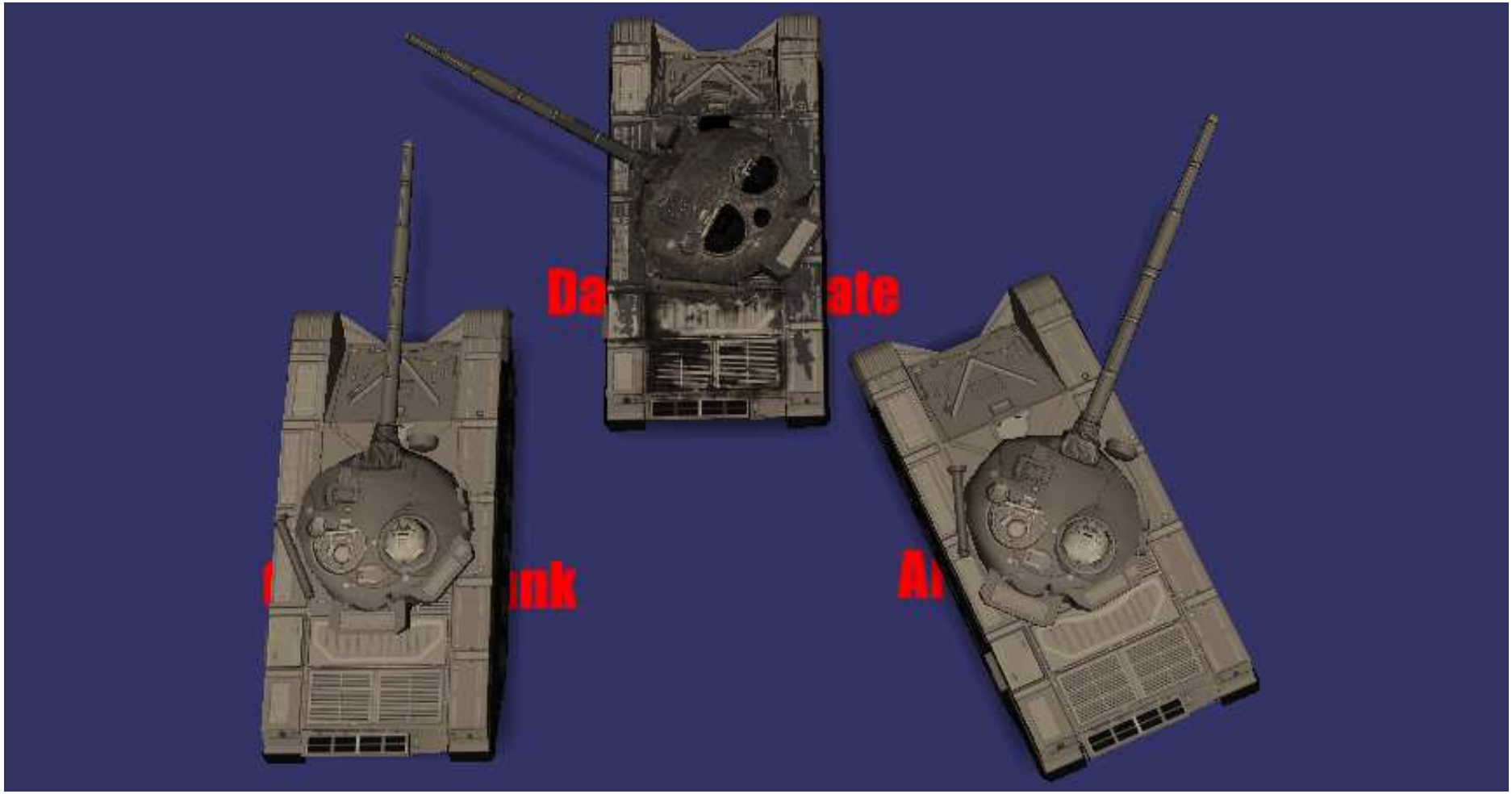
OSG – Texturas

OSG – Cenários

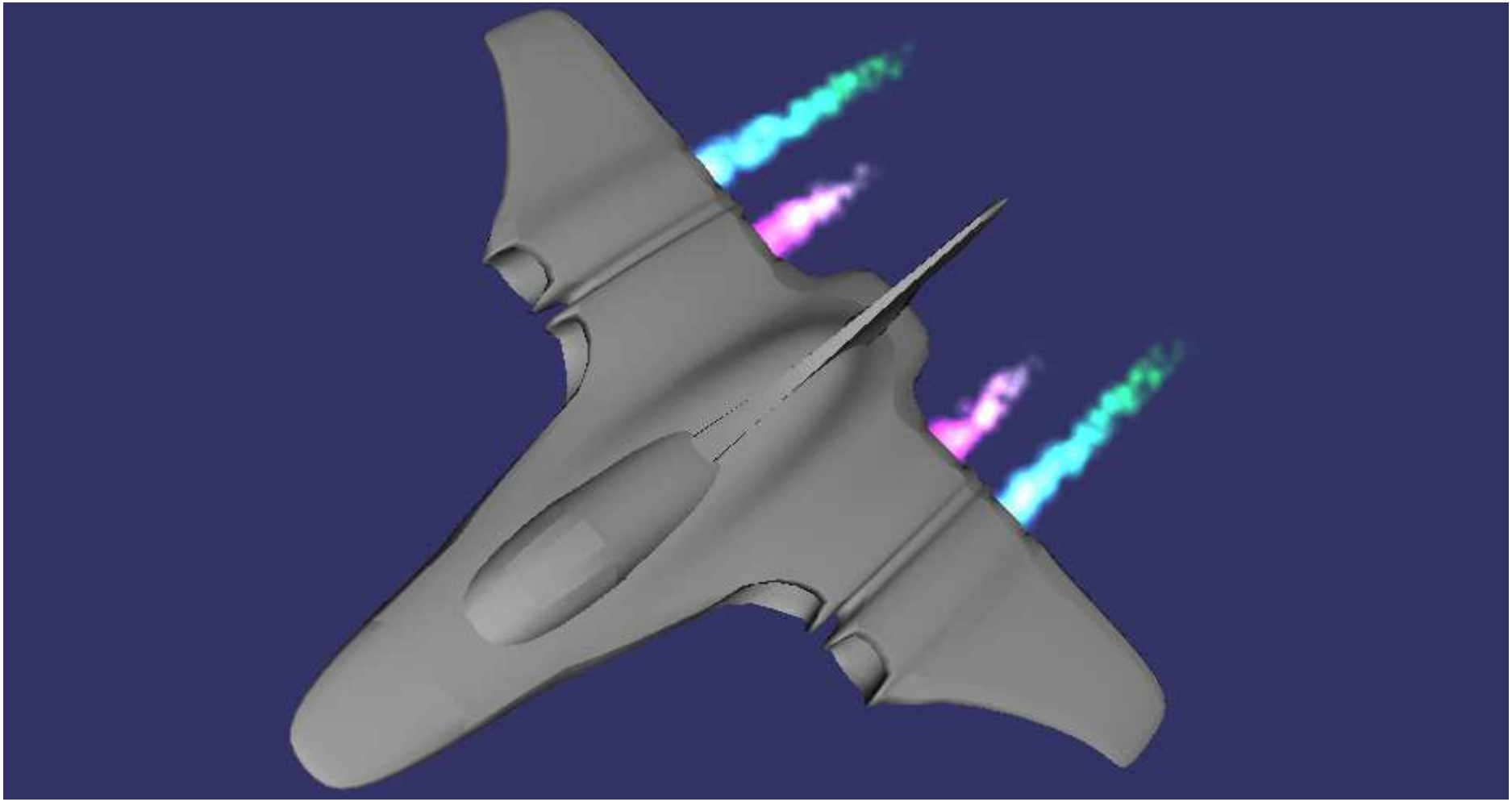
OSG – Escrevendo texto na tela

OSG – Escrevendo texto na tela

OSG – Texturas diferenciadas



OSG – Efeitos de partículas



OSG – Efeitos de partículas

OSG – Efeitos de partículas

Open Scene Graph - OSG

- Em suma, a OSG é uma biblioteca de computação gráfica que facilita o desenvolvimento de animações e jogos
- Ela permite que com poucos comandos se tenha uma cena com bastante realismo do ponto de vista visual
- Mas a OSG é uma biblioteca de visualização apenas – a movimentação dos corpos e o tratamento de colisões devem ser feitos pelo programador

Elementos de simulação em RV

- Percepção
- Ação
- Cinemática
- Dinâmica
- Colisão

Percepção

- Para que haja interação em RV, é necessário que se consiga perceber o ambiente
- Os seres humanos percebem o mundo através dos cinco sentidos
- Um robô precisa de diversos sensores para atuar no mundo real
- Um agente virtual precisa de sensores simulados:
  - ⇨ Em um jogo, o agente não pode ter uma visão universal
  - ⇨ O ideal é que os agentes virtuais percebam apenas o que está em seu raio de percepção (visão e audição)
  - ⇨ Um agente autônomo deve ser o mais próximo possível de um agente controlado por um humano (teste de Turing)

Percepção

- Robôs reais costumam possuir os seguintes sensores:
  - ⇨ Câmeras de vídeo / sensores visuais
  - ⇨ Microfones
  - ⇨ Bumbers
  - ⇨ Infravermelho, sonar e lazer
  - ⇨ Giroscópios e acelerômetros
  - ⇨ Odômetro, bússola e GPS
- Para um agente virtual se comportar de forma realista, ele precisa de sensores simulados, que emitam sinais similares aos dos sensores reais

Percepção

- Em um robô real, os dados sensoriais precisam ser tratados antes de serem utilizados na tomada de ação
- Os seres humanos utilizam cerca de 50% do cérebro apenas para o tratamento das imagens recebidas
  - ⇨ Os olhos captam radiação em uma certa faixa, chamada de luz visível
  - ⇨ O cérebro processa os sinais (detecção de contraste, bordas, profundidade, cores)
  - ⇨ O resultado do tratamento é aquilo que nós chamamos de visão
- Precisamos transformar os dados captados pelos sensores em informação útil para os agentes

Ação

- Ação é a forma como um agente responde aos estímulos do ambiente
  - ⇨ Ataque ou fuga
- No ser humano, a ação é realizada através de atuadores:
  - ⇨ Músculos
- Um robô real também possui atuadores:
  - ⇨ Juntas, motores, acionadores pneumáticos, rodas, garras
- Um agente virtual também precisa de atuadores simulados para interagir com o ambiente

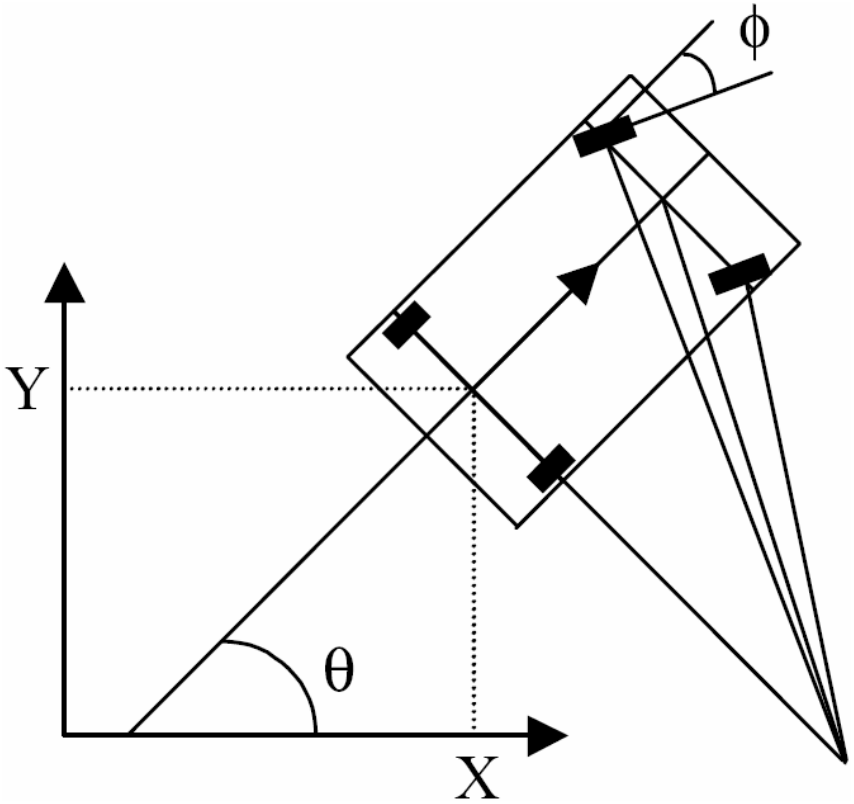
Ação

- Os atuadores de um agente virtual devem ser condizentes com a física:
  - ⇨ Se um agente precisa ir do ponto A ao ponto B, ele não pode simplesmente ser tele-transportado (a não ser em Star Trek)
  - ⇨ O agente não pode atravessar as paredes
  - ⇨ Obstáculos físicos (chão irregular) também devem ser respeitados
- Um agente deve se movimentar como se fosse as suas pernas que o conduzissem

Cinemática

- É o ramo da física que estuda os corpos em movimento
- Um corpo em movimento precisa ter sua posição atualizada a cada instante de tempo, de acordo com a sua velocidade, direção e sentido
- Na simulação de um veículo, o cálculo da posição é mais complexo, pois o ângulo das rodas pode alterar a trajetória
- O modelo cinemático mais utilizado em veículos do tipo carro é conhecido por cinemática Ackerman, onde o carro é descrito através de três equações diferenciais
- Integrando as equações é possível calcular a posição que o veículo deve se encontrar em determinado instante

Cinemática Ackerman



$$x(t) = \int_0^t V(t)\cos[\phi(t)]\cos[\theta(t)]\,dt$$

$$y(t) = \int_0^t V(t)\cos[\phi(t)]\sin[\theta(t)]\,dt$$

$$\theta(t) = \int_0^t \frac{V(t)}{L}\tan[\phi(t)]\,dt$$

V(t) representa a velocidade do veículo no instante *t*
Φ(t) representa o giro da direção no instante *t*
L representa o comprimento do eixo do veículo

Dinâmica

- Um corpo em movimento não pode estar parado no instante 0 e a 100km/h no instante seguinte
  - ⇨ Existe uma etapa de aceleração
- Da mesma forma, um corpo não pode simplesmente parar de um instante para o outro
  - ⇨ Existe uma etapa de desaceleração
- Uma parada brusca pode causar diversos efeitos:
  - ⇨ Derrapagem, deformação e calor
- Em suma, para uma simulação ser realista, ela precisa seguir as leis da física, em especial a dinâmica dos corpos rígidos

Dinâmica

- Uma simulação que respeita a dinâmica dos corpos rígidos precisa levar em conta os seguintes elementos:
  - ⇨ Gravidade
  - ⇨ Atrito e fricção
  - ⇨ Força e torque
  - ⇨ Impacto
- Os antigos jogos de corrida não respeitavam completamente a dinâmica- o impacto não danificava o carro
- A maioria dos personagens virtuais não caminha de forma realista – eles parecem "patinar".
  - ⇨ Isto ocorre porque não são seus atuadores que os fazem caminhar – a imagem é simplesmente deslocada

Colisão

- O tratamento de colisões é bastante complexo e caro de ser implementado computacionalmente
- Os jogos antigos simplesmente impediam que um personagem saísse fora das fronteiras do ambiente- quando dois personagens se aproximavam, eles se atravessavam
- Mas uma simulação realista precisa detectar uma colisão e tratá-la de forma adequada:
  - ⇨ Se um veículo colidir contra um prédio, este não pode simplesmente atravessá-lo
  - ⇨ Uma bola jogada contra uma parede precisa quicar
  - ⇨ Uma pedra jogada contra um vidro deve quebrá-lo
- O tratamento de colisões deve respeitar as leis da física

Elementos de simulação em RV

- Elementos:
  - ⇨ Percepção
  - ⇨ Ação
  - ⇨ Cinemática
  - ⇨ Dinâmica
  - ⇨ Colisão
- Em uma simulação realista, os personagens devem se comportar de forma similar aos elementos do mundo real
  - ⇨ As leis da física precisam ser implementadas, em especial a dinâmica e a cinemática

Open Dynamics Engine - ODE

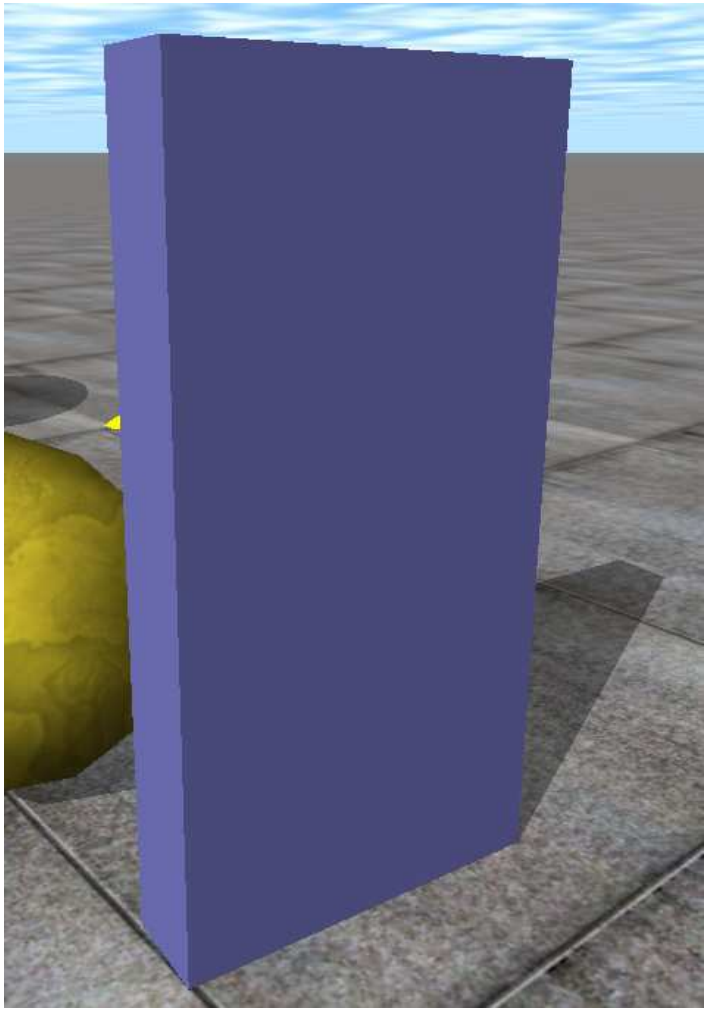
- ODE é um pacote de simulação baseada em física open source e free para C/C++
- Possui uma interface (API) C não orientada a objetos (procedural)
- Permite a construção de um mundo físico:
  - ⇨ Gravidade, aceleração, atrito, aplicação de forças aos corpos, torque
- Permite o tratamento de colisões:
  - ⇨ Choques contra o solo e entre os objetos
  - ⇨ Atrito, bounce (quicar)
- Permite a utilização de juntas (joints) e motores de vários tipos

Open Dynamics Engine - ODE

- Tipos de objetos suportados:
  - ⇨ Cubos, esferas, cilindros, capped cylinders e objetos compostos
- Objetos mais complexos são possíveis, mas dificultam a detecção de colisões
- A complexidade computacional do ODE é $O(n^2)$, onde n é o número de objetos
- A simulação avança através de um loop de passos físicos, onde o tamanho do passo pode ser informado em segundos
  - ⇨ Quanto maior o passo, mais rápida é a simulação, mas passos muito grandes geram erros

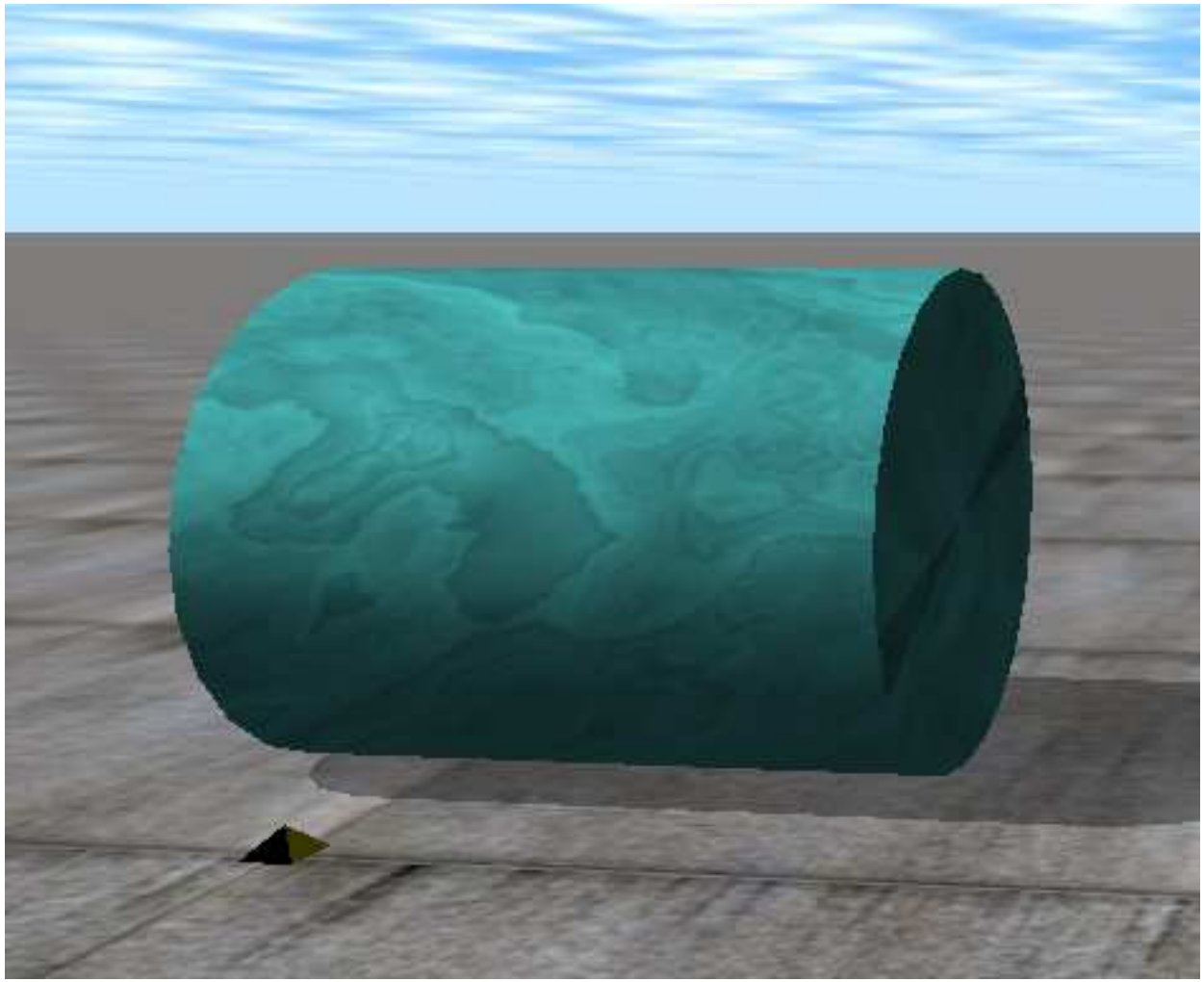
ODE - Objetos suportados

- Cubos

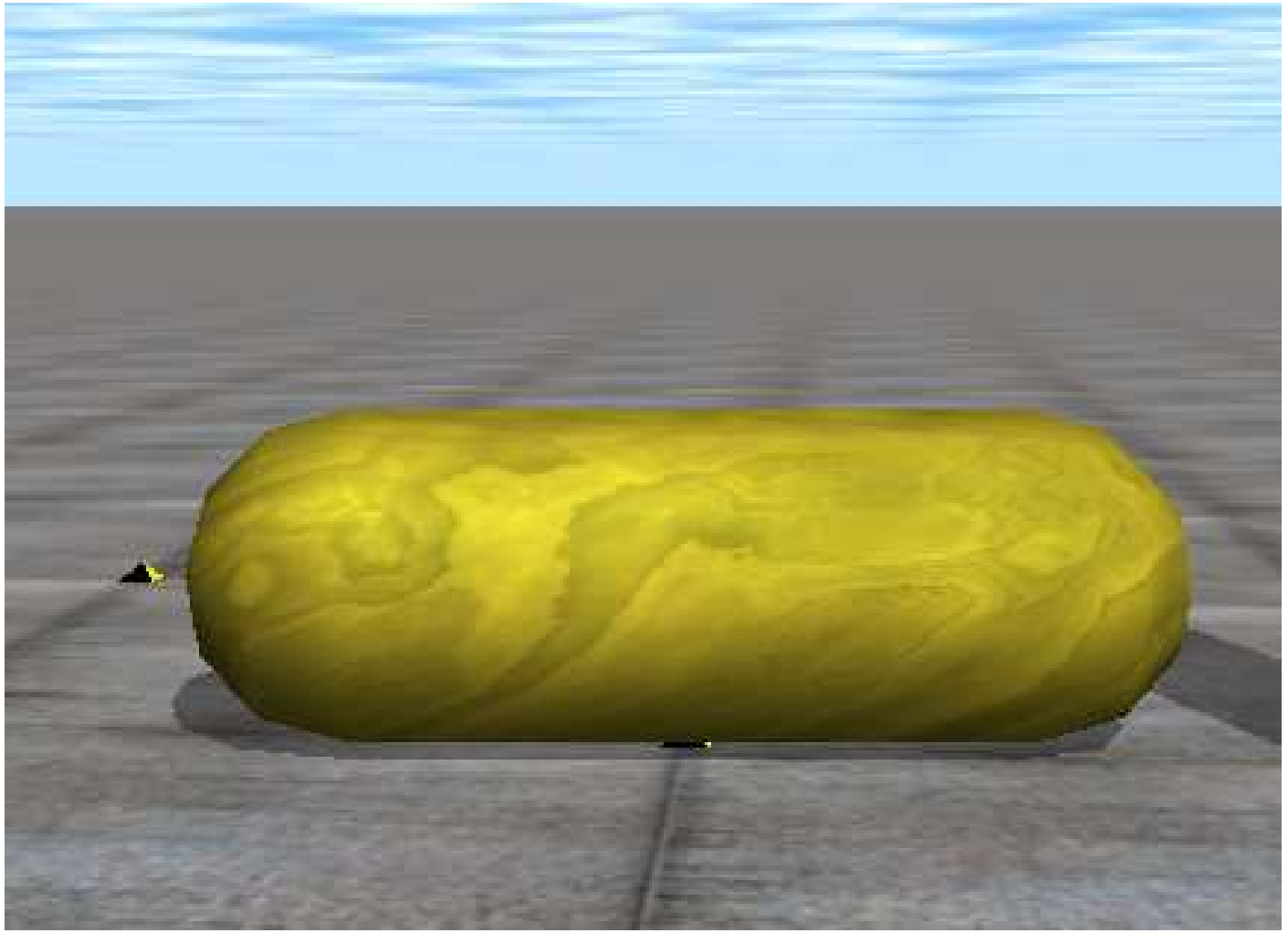


Objetos suportados - Esferas

Objetos suportados - Cilindros



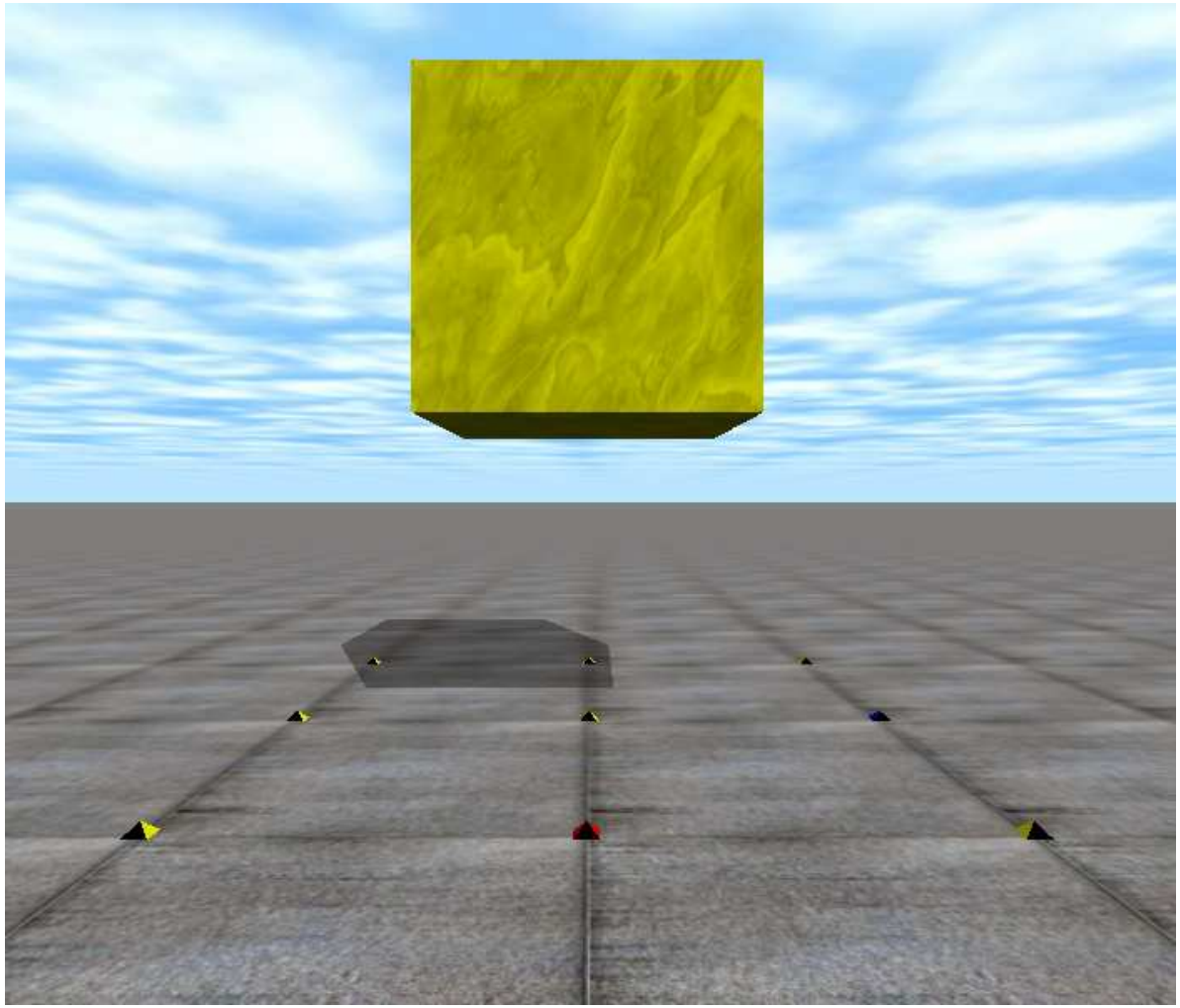
Objetos suportados - Capped Cylinders



Open Dynamics Engine - ODE

Na computação gráfica tradicional, os objetos não reagem às leis da física.

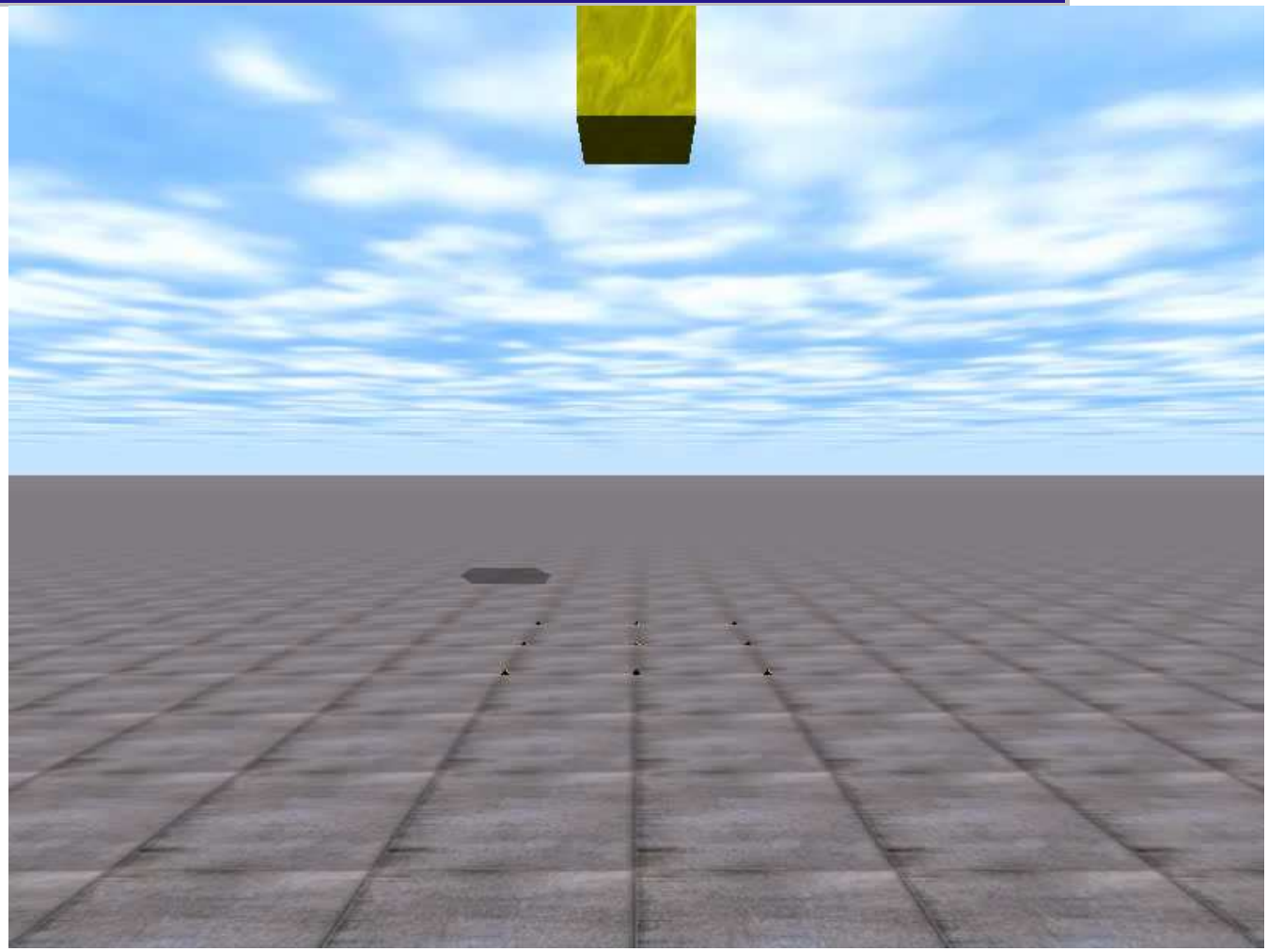
Este cubo, por exemplo, se não for deslocado manualmente via código, ficará para sempre flutuando.



Open Dynamics Engine - ODE

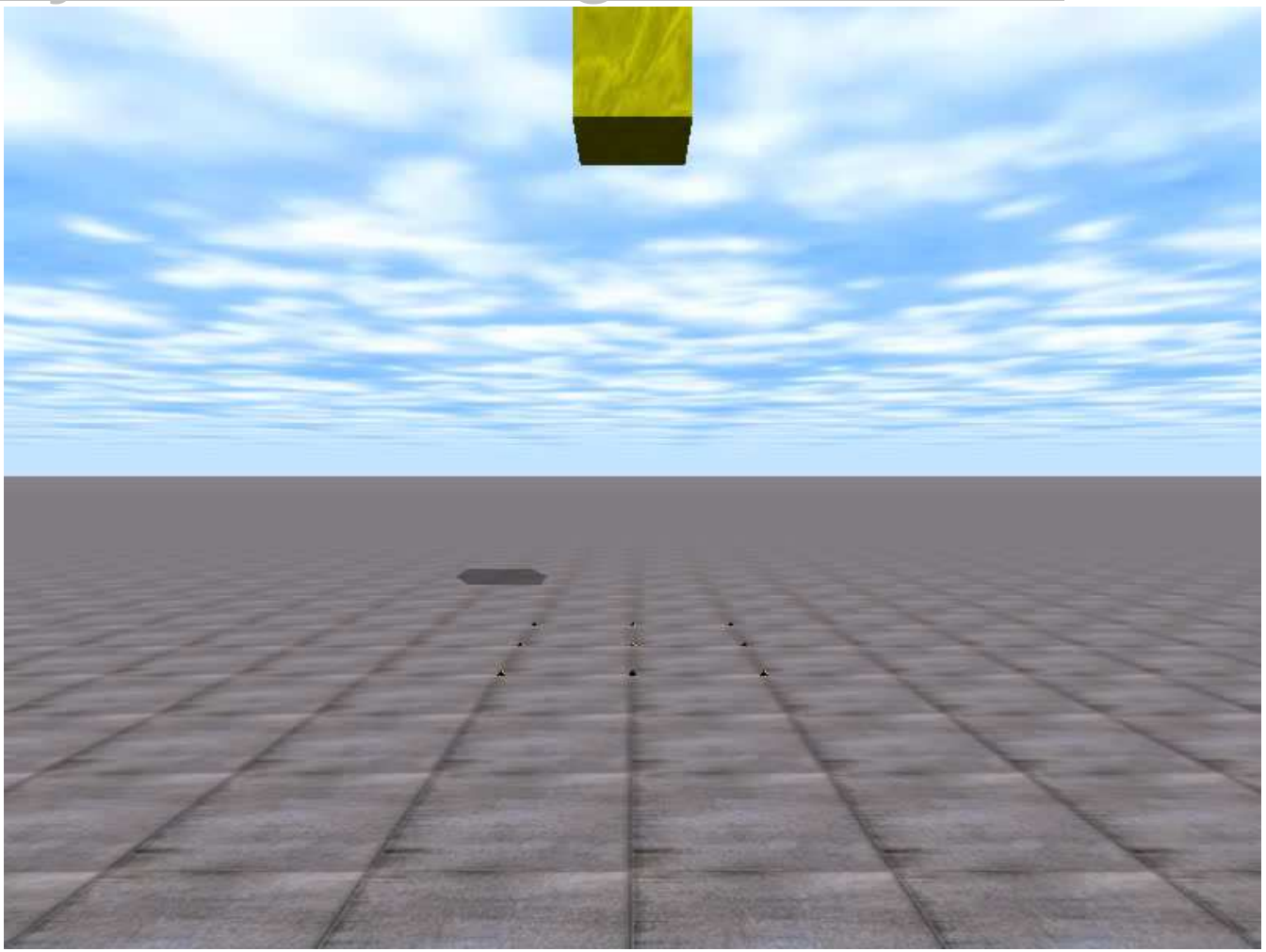
Com a ODE, podemos fazer com que o cubo sofra a ação da gravidade.

Podemos inclusive setar gravidades diferentes da nossa.



Open Dynamics Engine - ODE

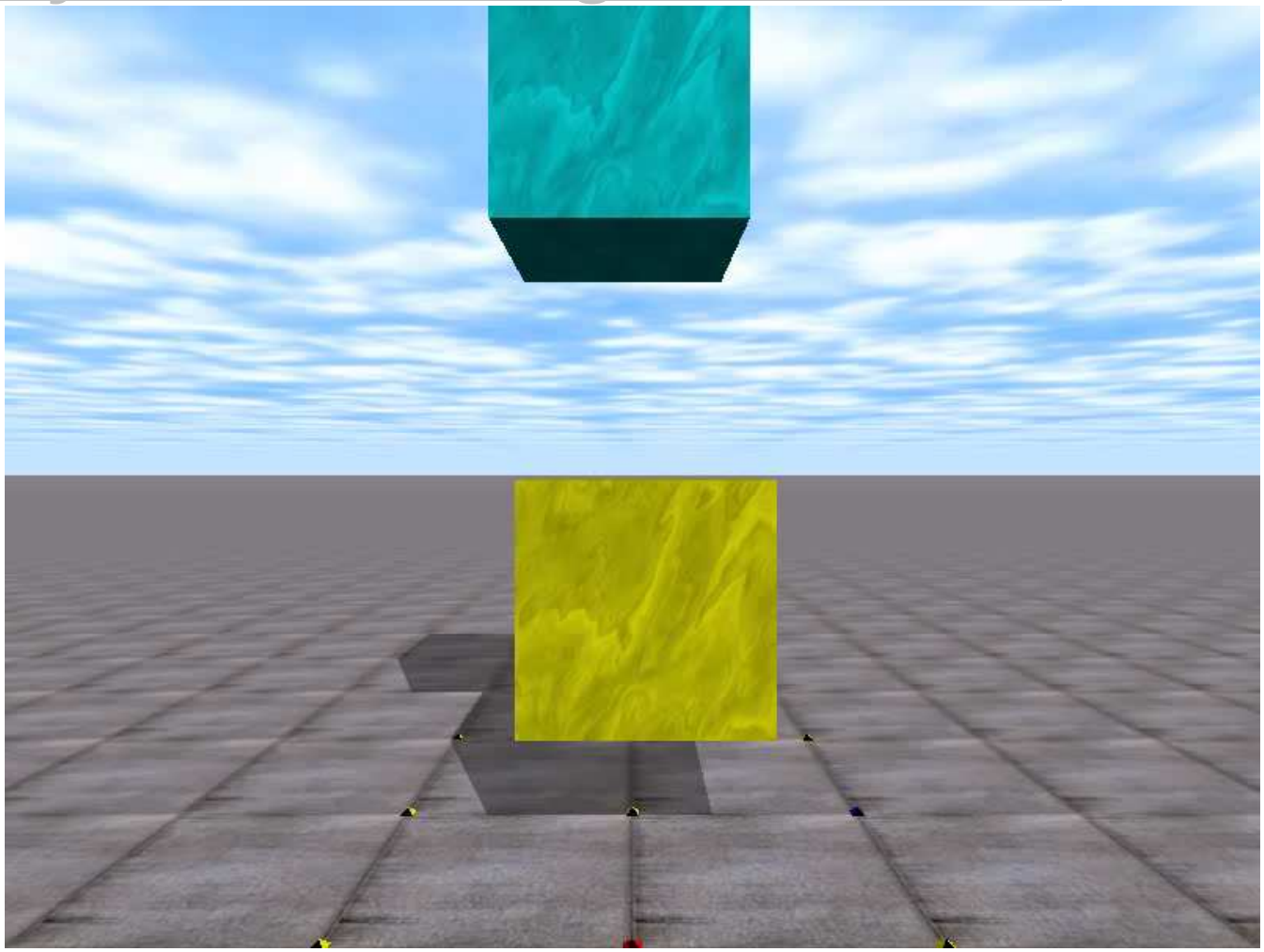
O tratamento de colisões é necessário para que o objeto não atravesse o chão, e o impacto deve se dar de forma realista.



Open Dynamics Engine - ODE

O tratamento de colisões entre objetos também precisa ser realizado.

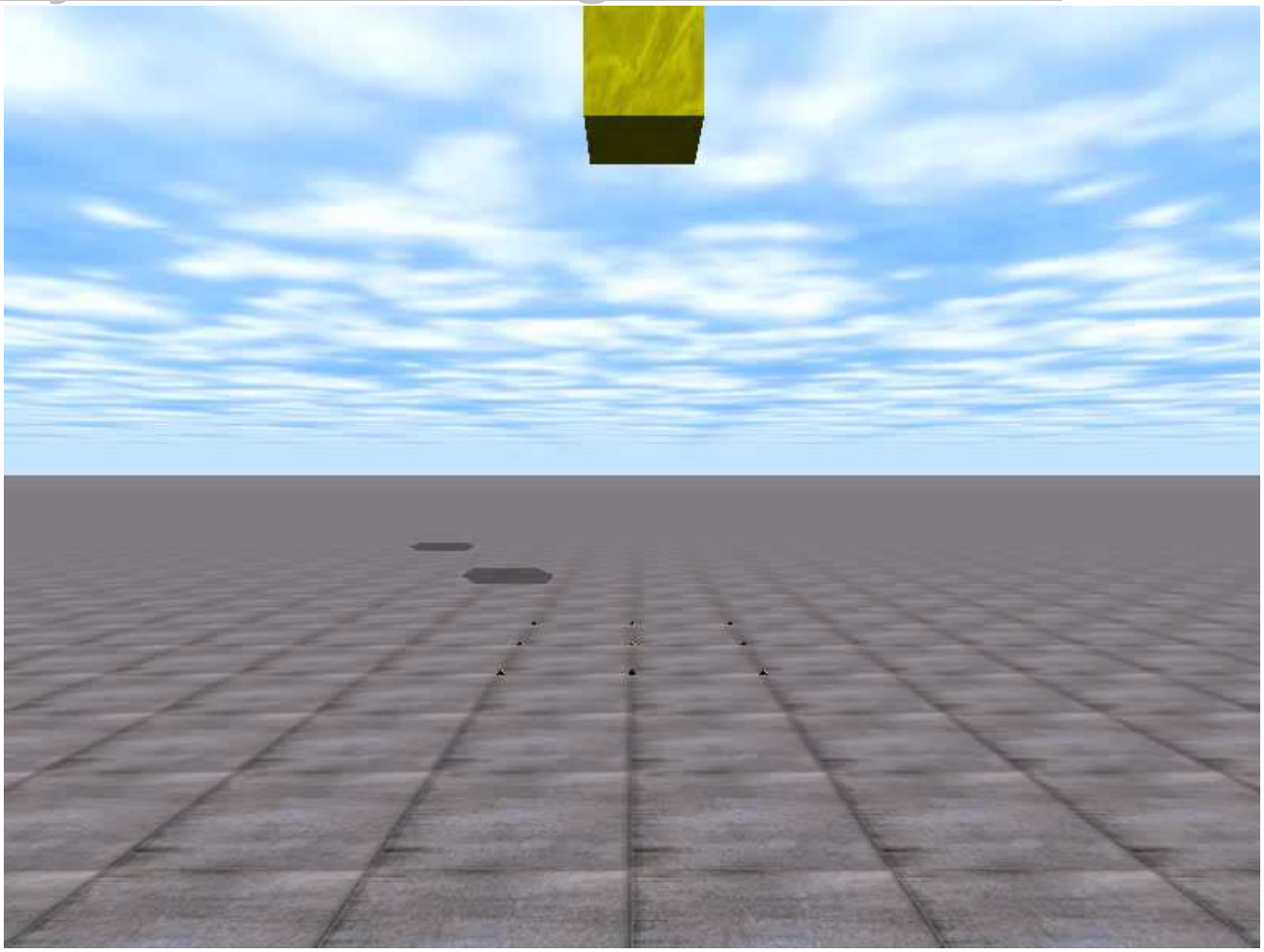
Em muitos jogos, os personagens simplesmente se atravessam.



Open Dynamics Engine - ODE

Mas com o tratamento correto, os efeitos se tornam realistas.

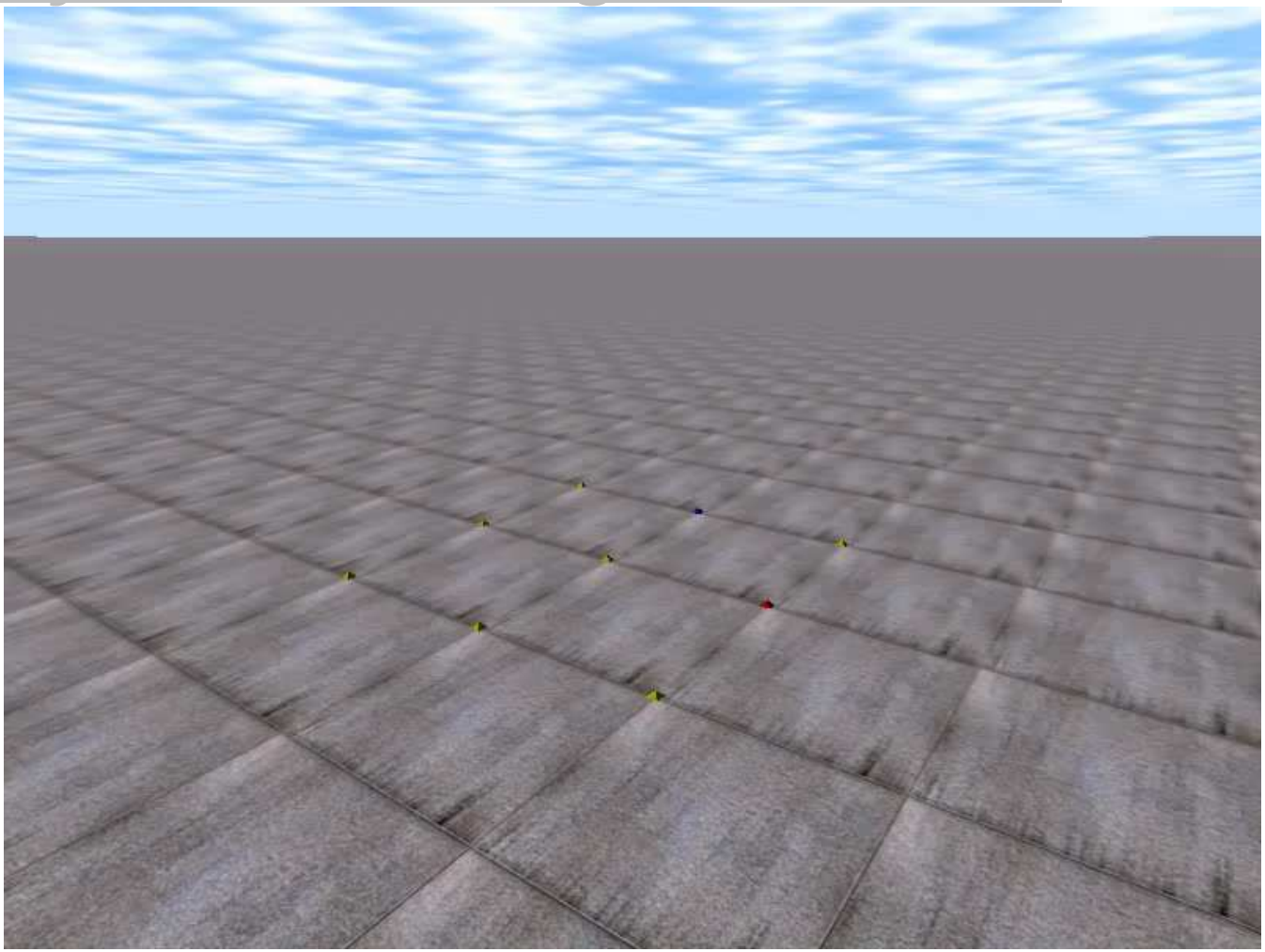
A ODE faz este tratamento de forma automática



Open Dynamics Engine - ODE

Neste vídeo, vários objetos são criados e caem no chão.

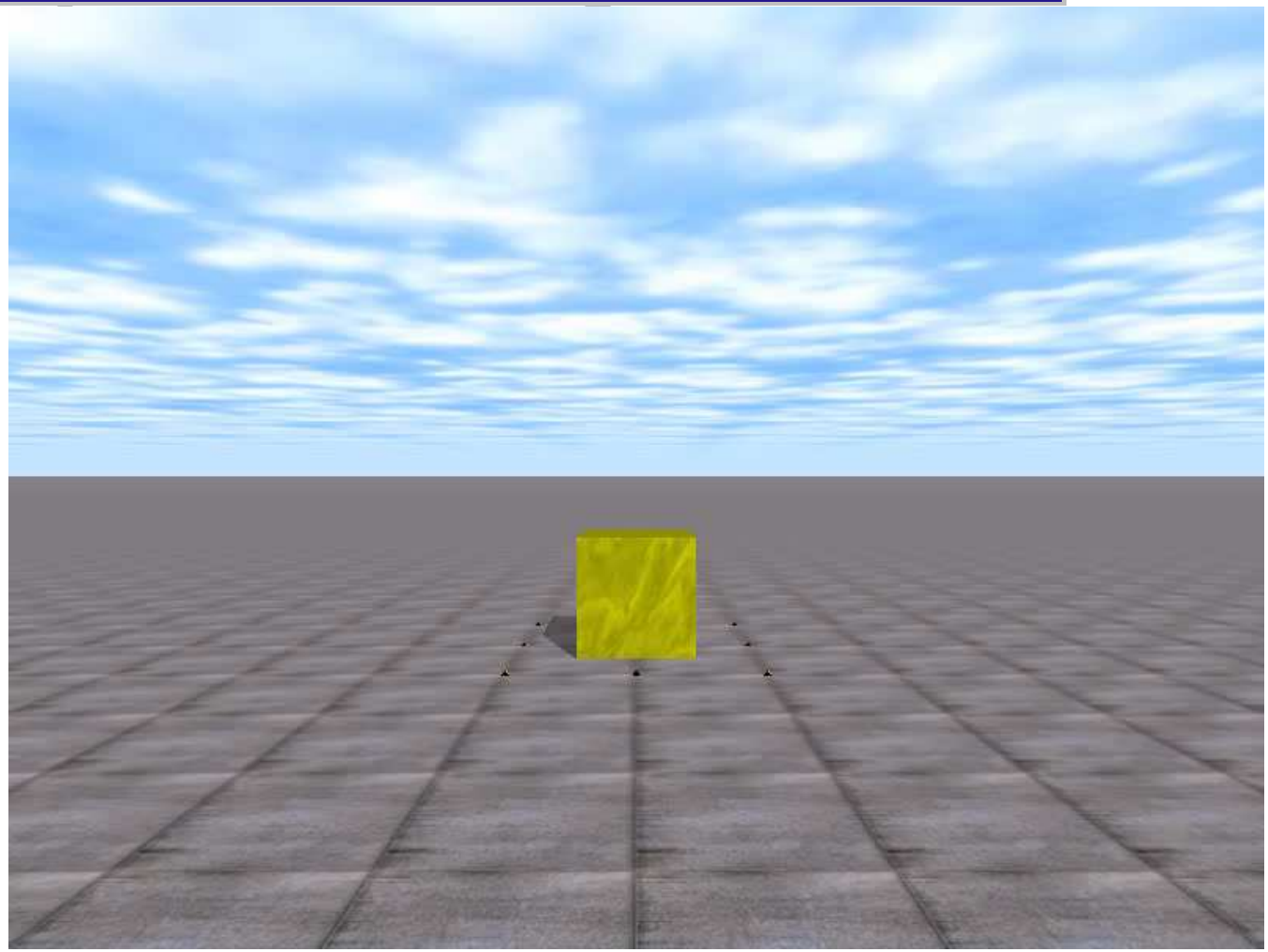
Repare que alguns dos objetos são compostos (formados por vários elementos)



Open Dynamics Engine - ODE

A ODE permite também a aplicação de forças aos objetos.

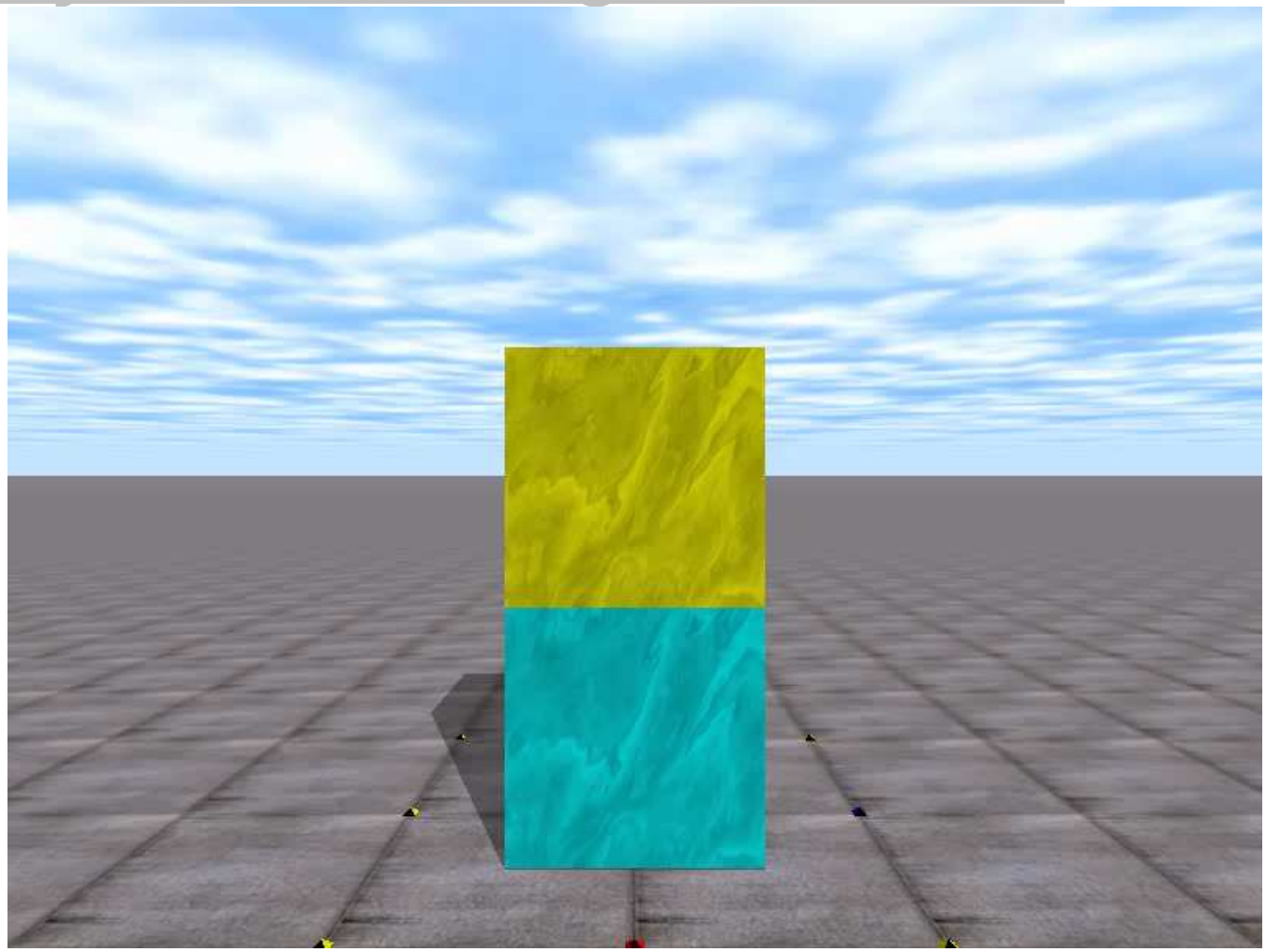
Uma força é um vetor que possui intensidade, direção e sentido



Open Dynamics Engine - ODE

A interação entre os objetos ocorre de forma natural.

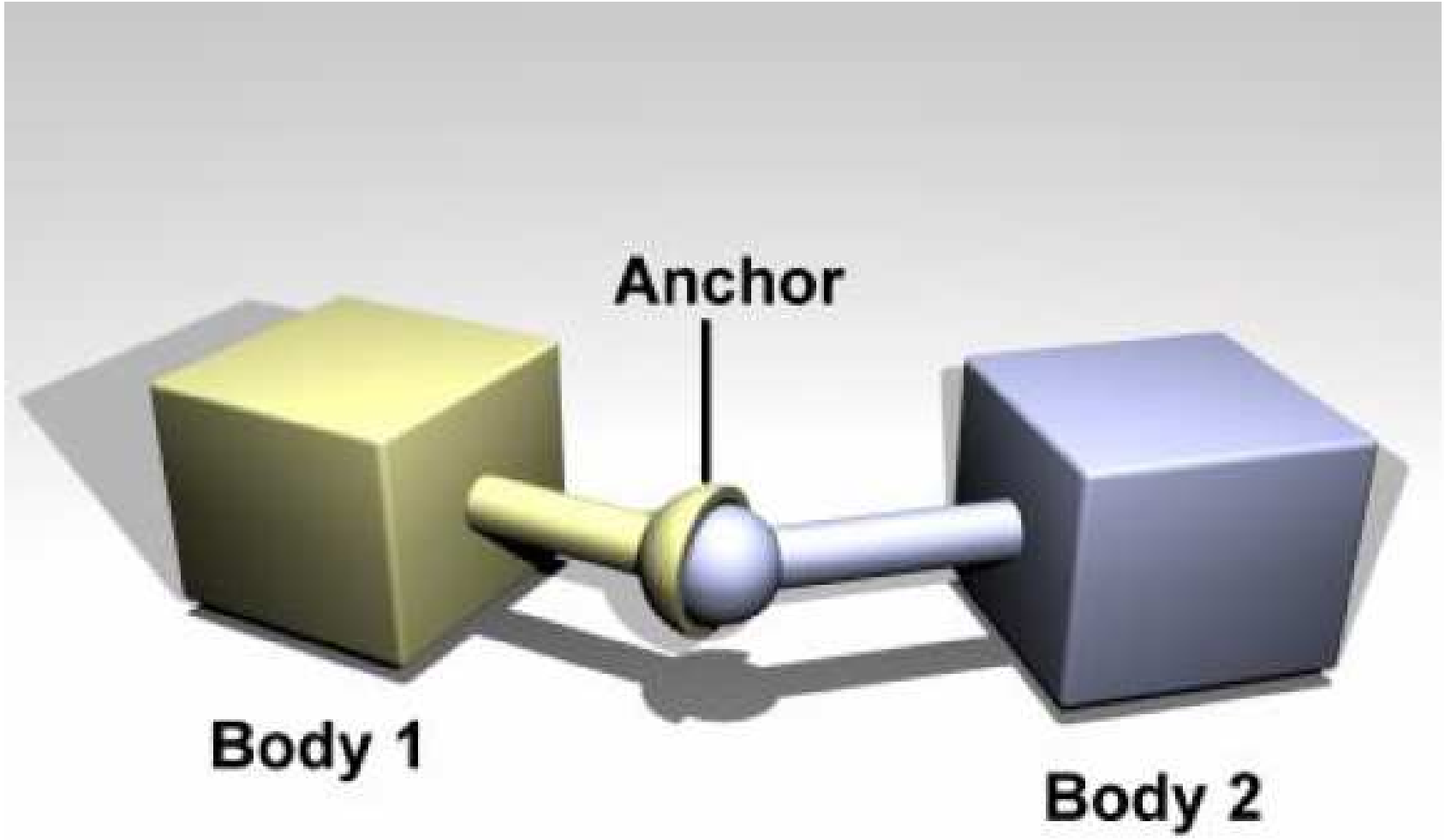
Um objeto pode ser atirado contra outro, causando uma reação.



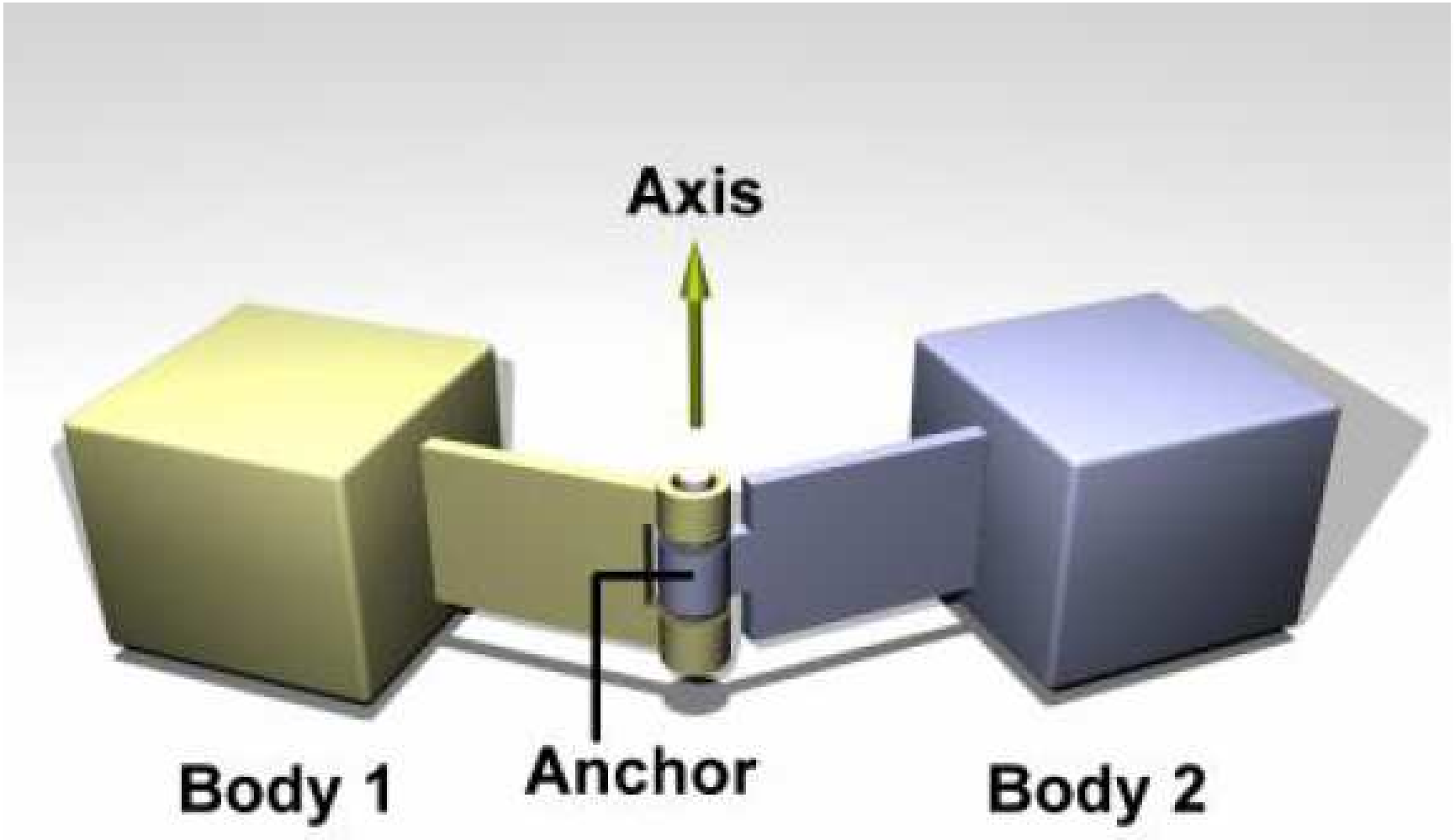
ODE Joints

- Tipos de juntas:
  - ball and socket, hinge, slider, universal, contact, etc
  - As juntas possuem um ou mais eixos (dependendo do tipo de junta) e limites máximos e mínimos
  - É possível obter o ângulo atual da juntas
  - Não é possível setar explicitamente os ângulos de uma junta – a única forma de alterá-los é através da aplicação de forças ou utilizando motores
- Angular Motors:
  - User (manual) e Euler (automático)
  - Permitem que sejam definidos o eixo de atuação, a velocidade e a força máxima de cada motor

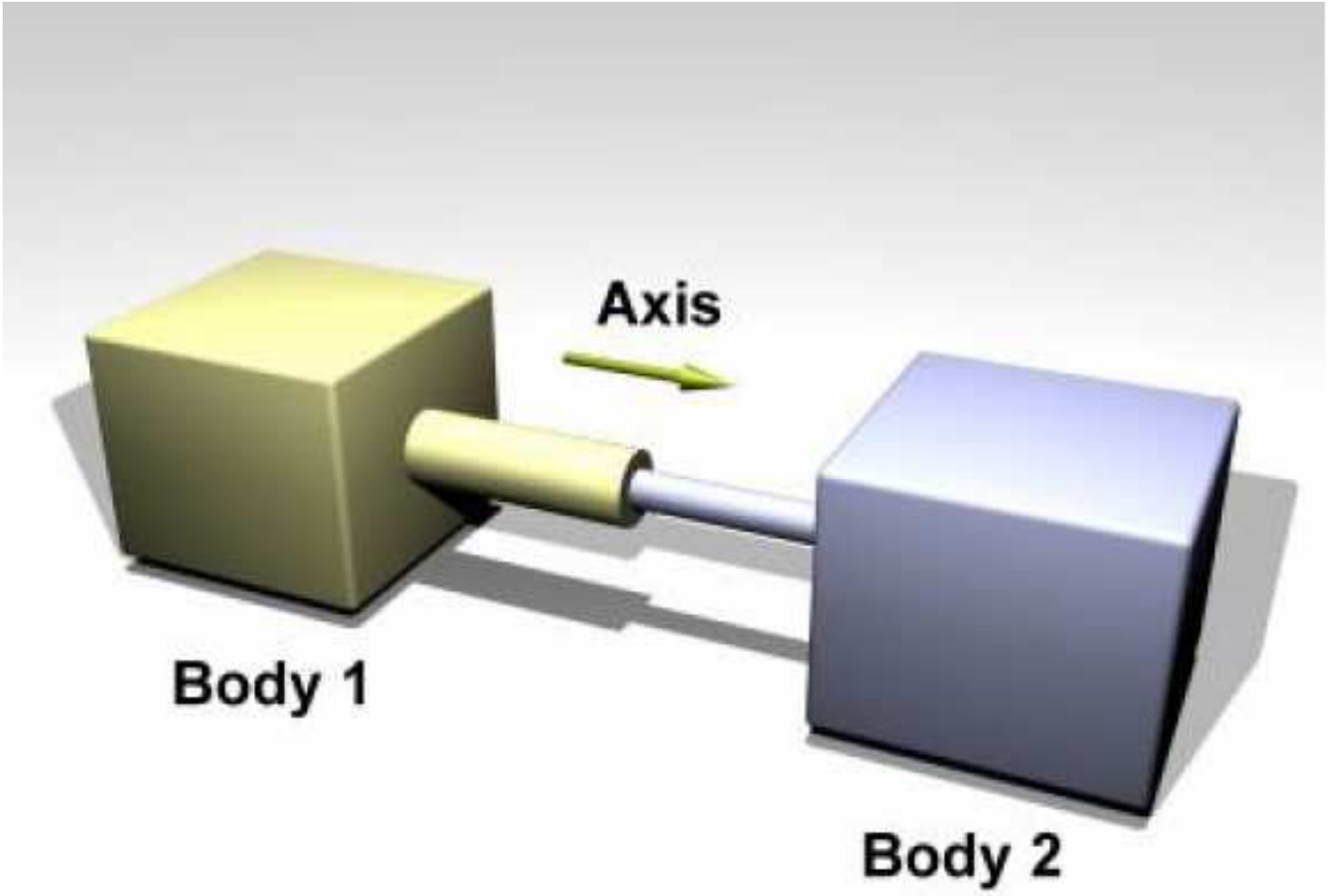
A ball and socket joint



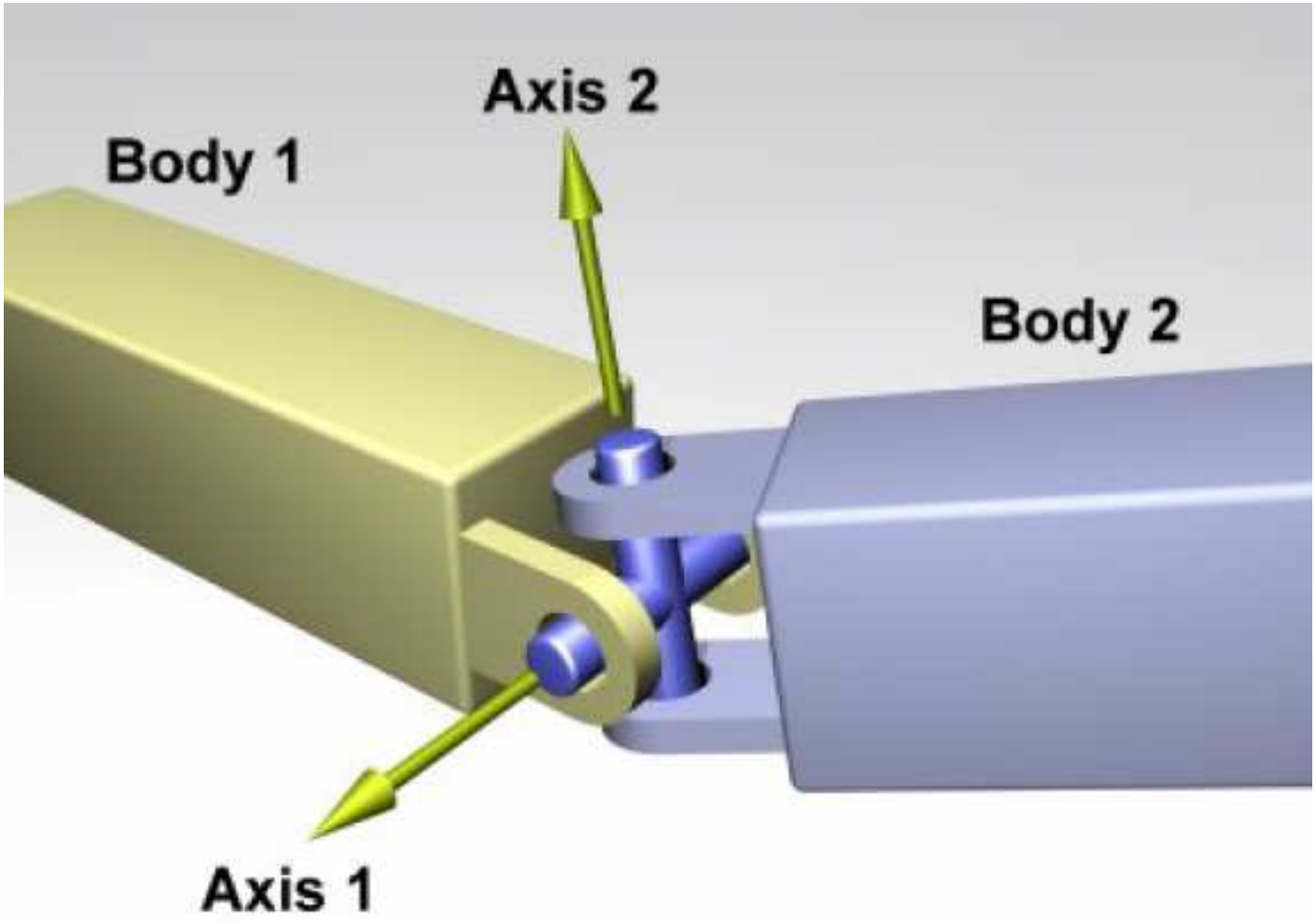
A hinge joint



A slider joint



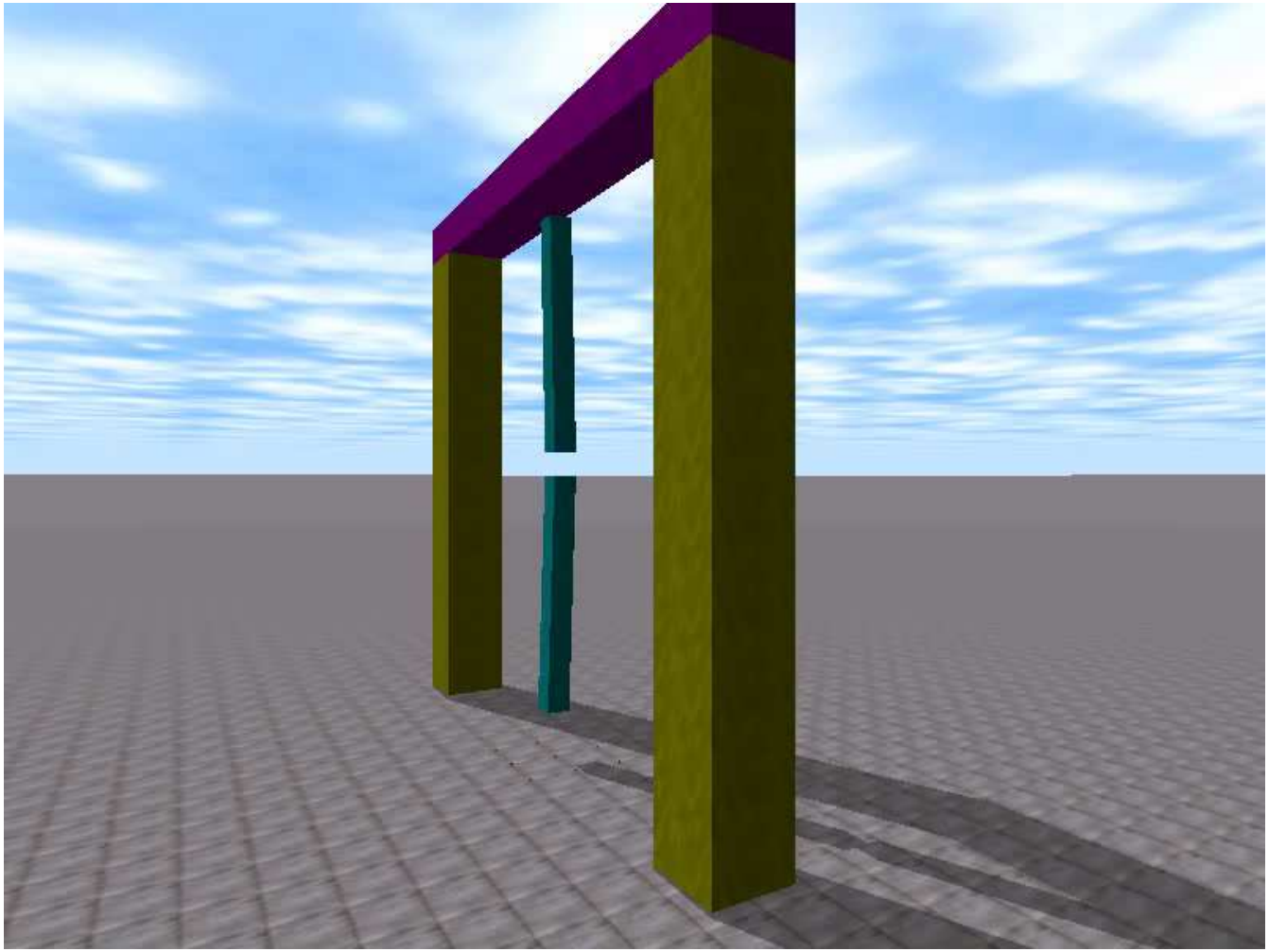
A universal joint



Open Dynamics Engine - ODE

Juntas podem ser utilizadas para unir dos objetos.

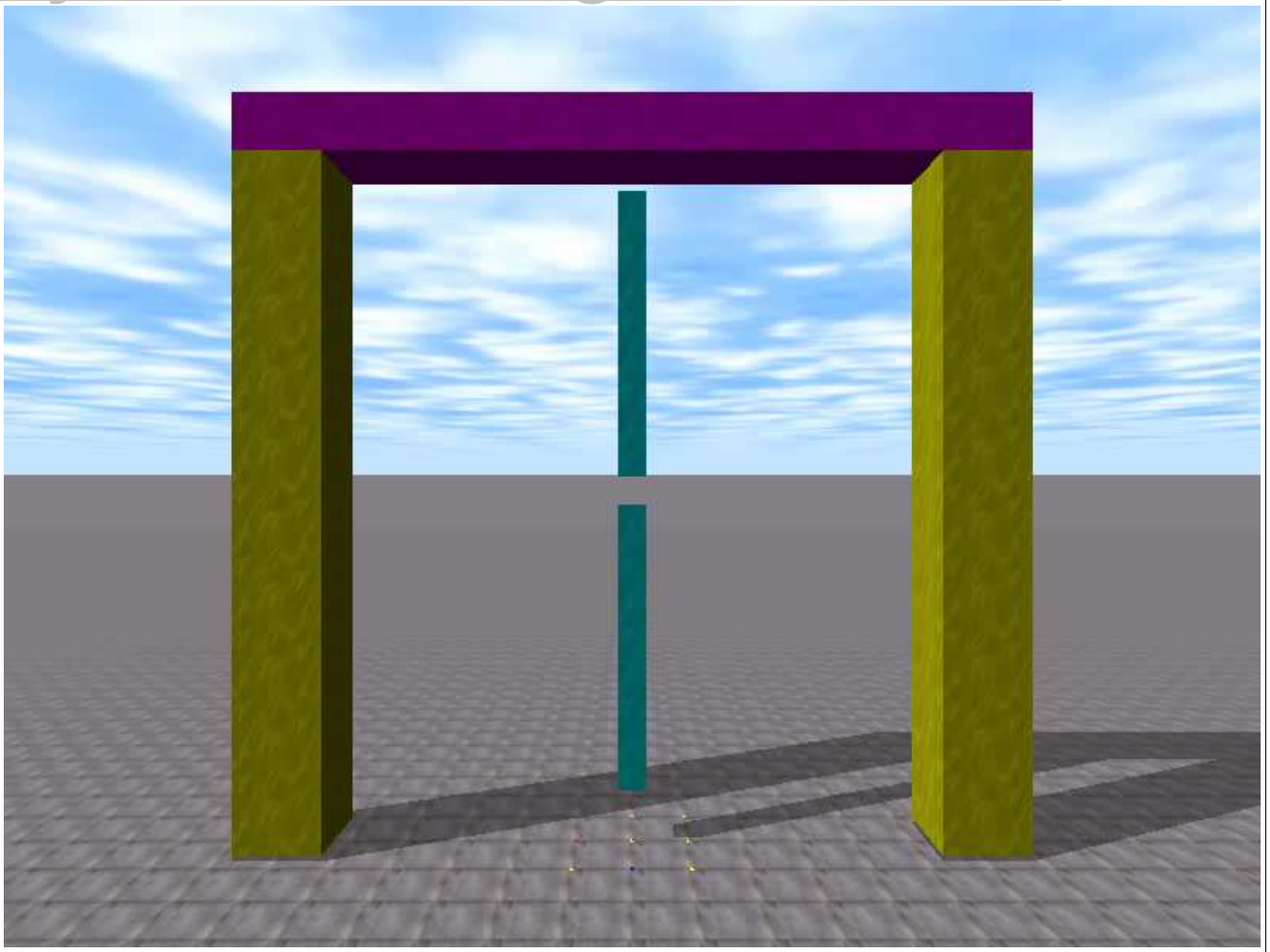
As juntas possuem um ponto de conexão, graus de liberdade (eixos) e limites máximos e mínimos.



Open Dynamics Engine - ODE

Quando aplicamos uma força em objetos unidos por juntas, eles reagem como se tivessem uma dobradiça.

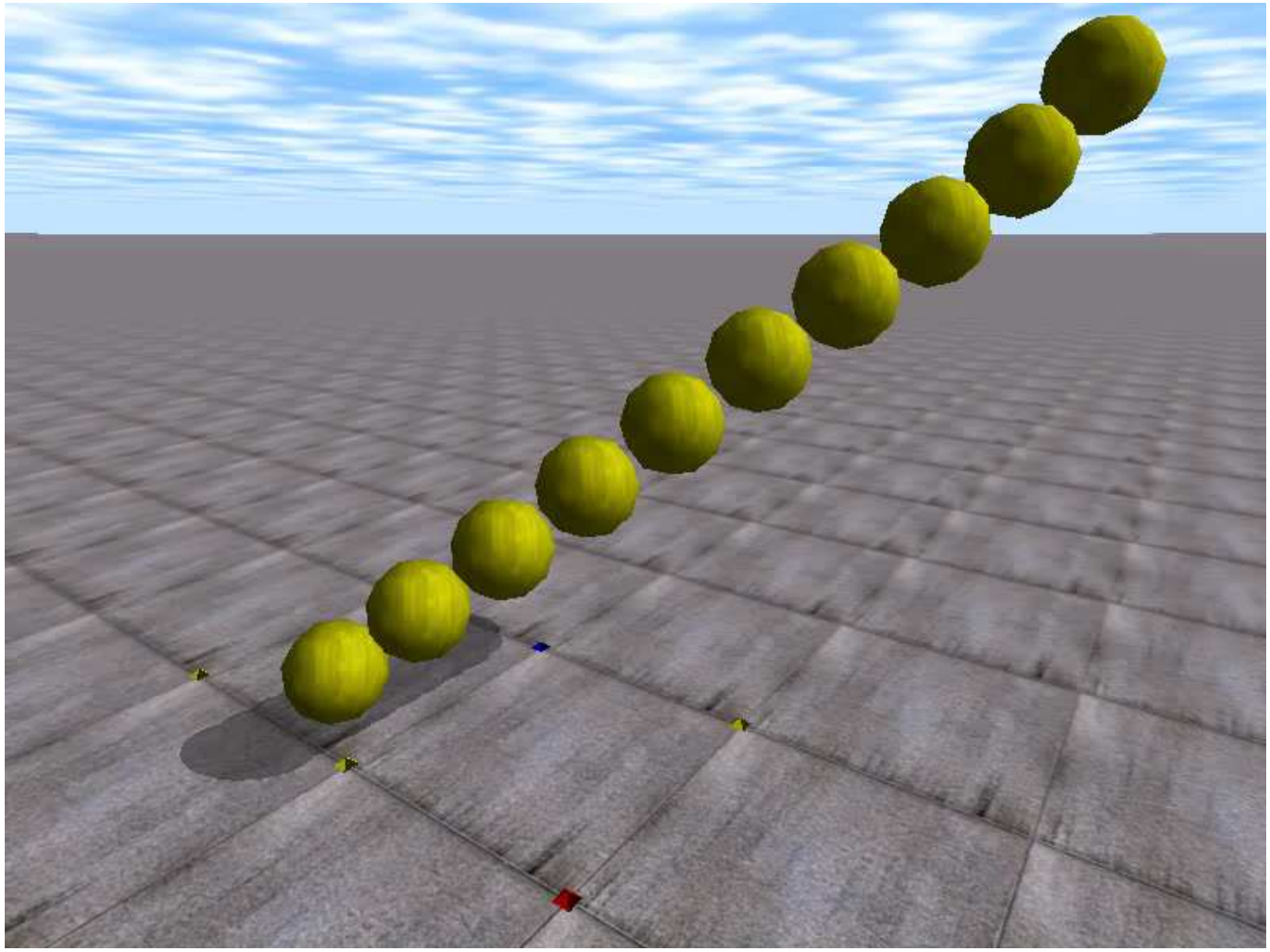
Isto faz com que eles balancem livremente, dentro do intervalo da junta.



Open Dynamics Engine - ODE

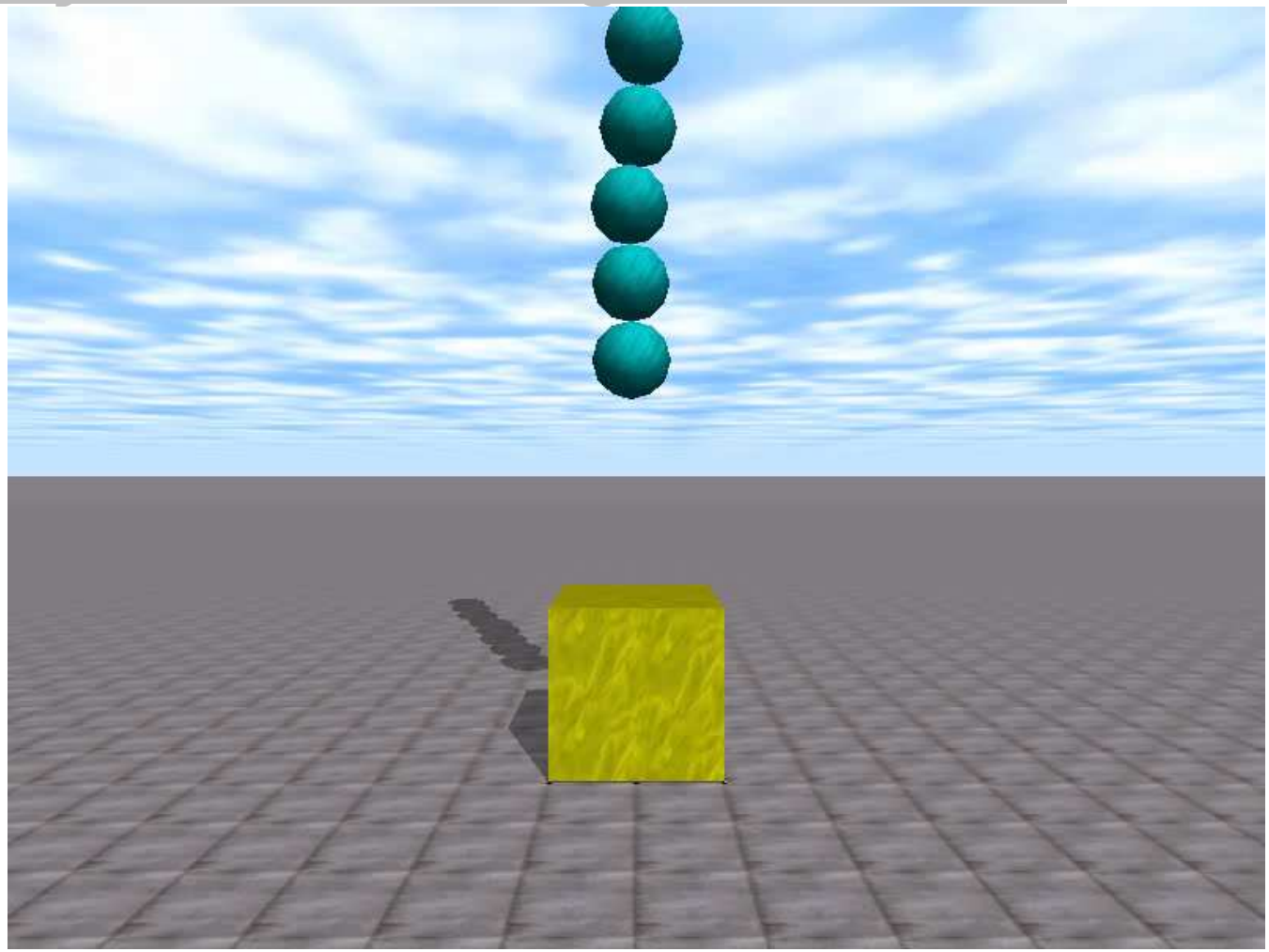
Vários objetos podem ser unidos utilizando juntas do tipo universal.

No vídeo ao lado, a serpente é formada por várias esferas unidas por uma “corda invisível”.



Open Dynamics Engine - ODE

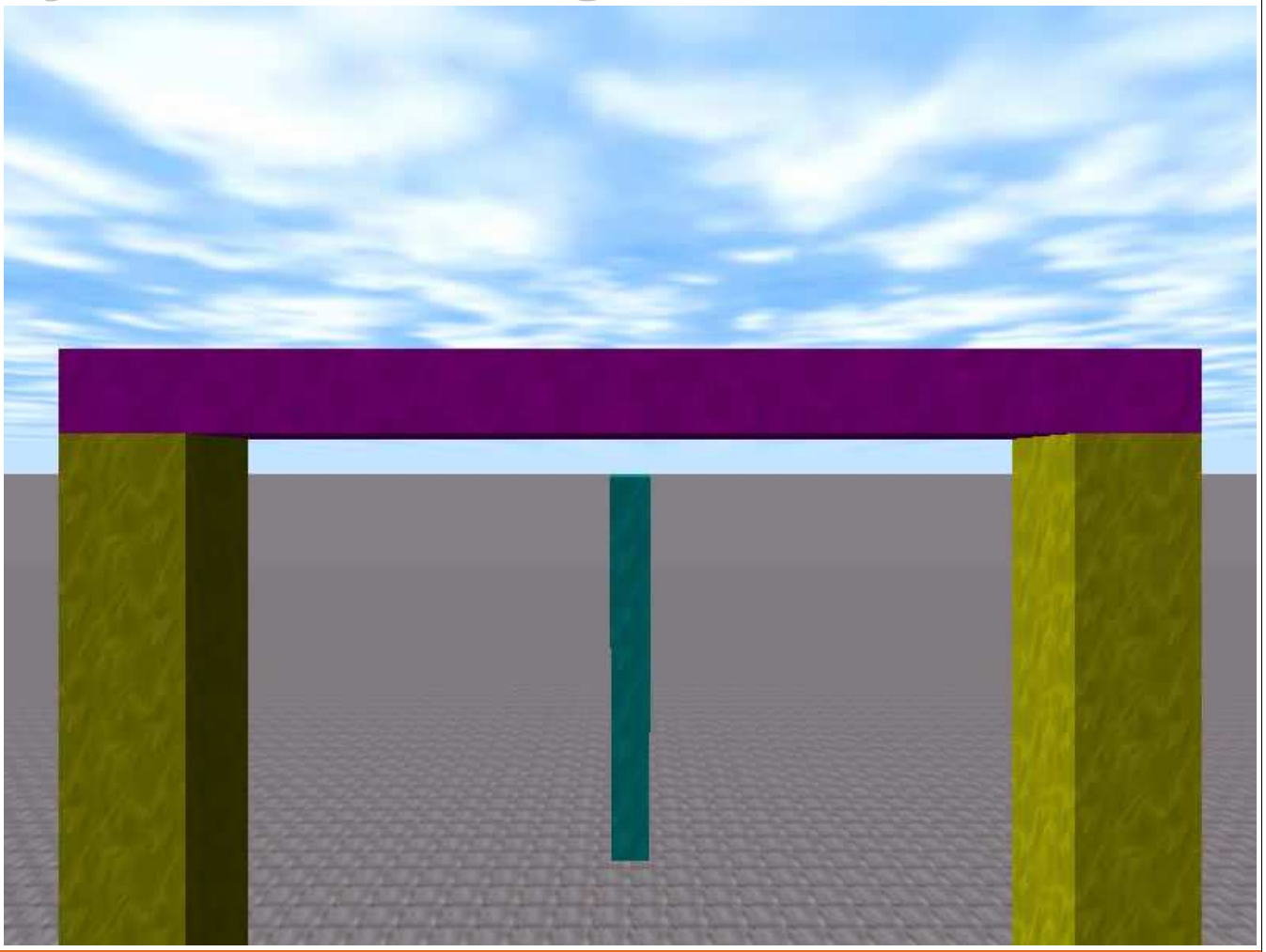
Neste vídeo, a serpente sofre a ação da gravidade e colide contra o cubo.



Open Dynamics Engine - ODE

Motores angulares podem ser utilizados para movimentar os objetos unidos por juntas.

Isto permite que sejam construídos veículos e robôs simulados.

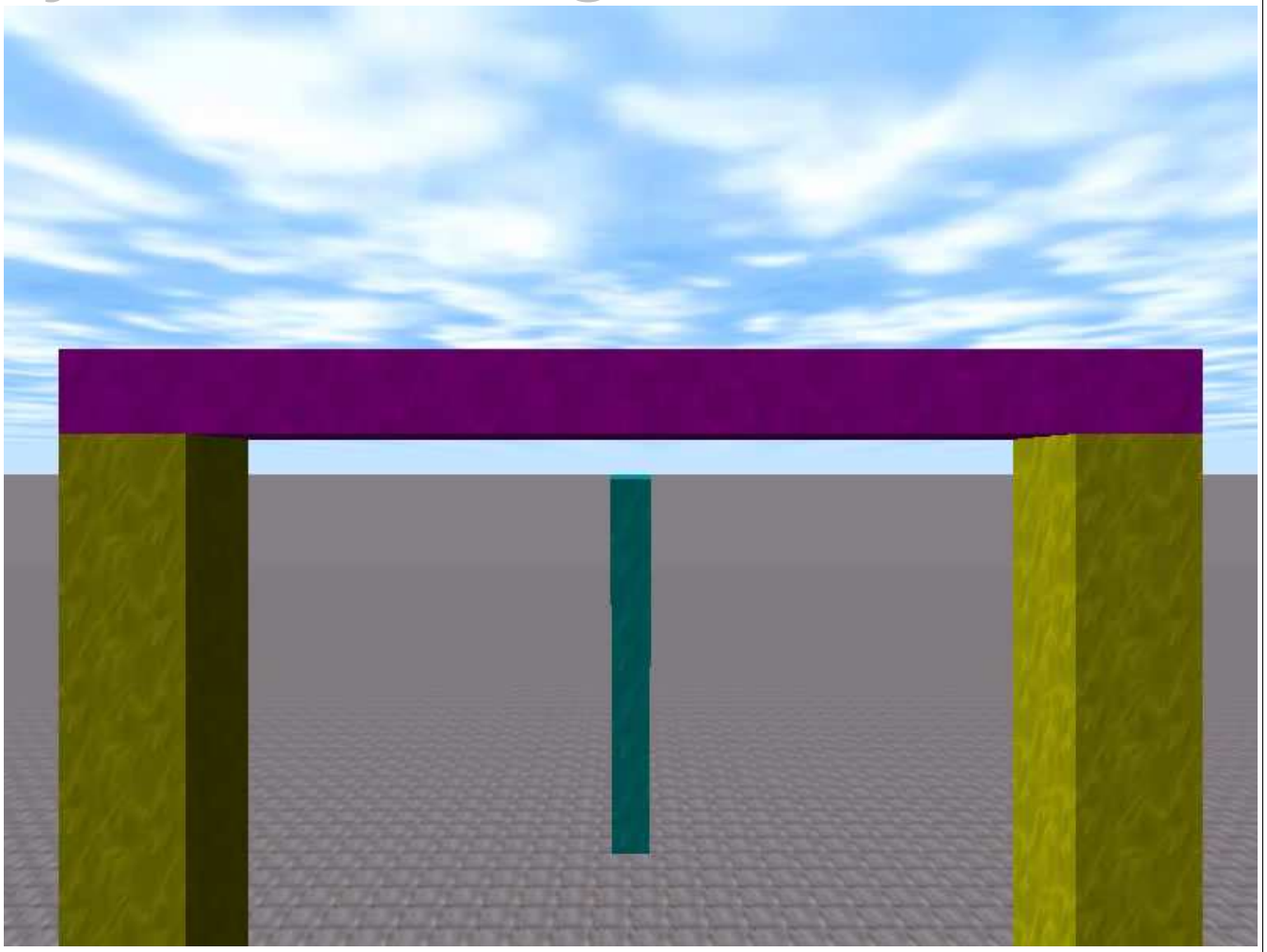


Open Dynamics Engine - ODE

A força máxima e a velocidade de um motor angular pode ser configurada e alterada durante a simulação.

O usuário pode controlar isto via teclado.

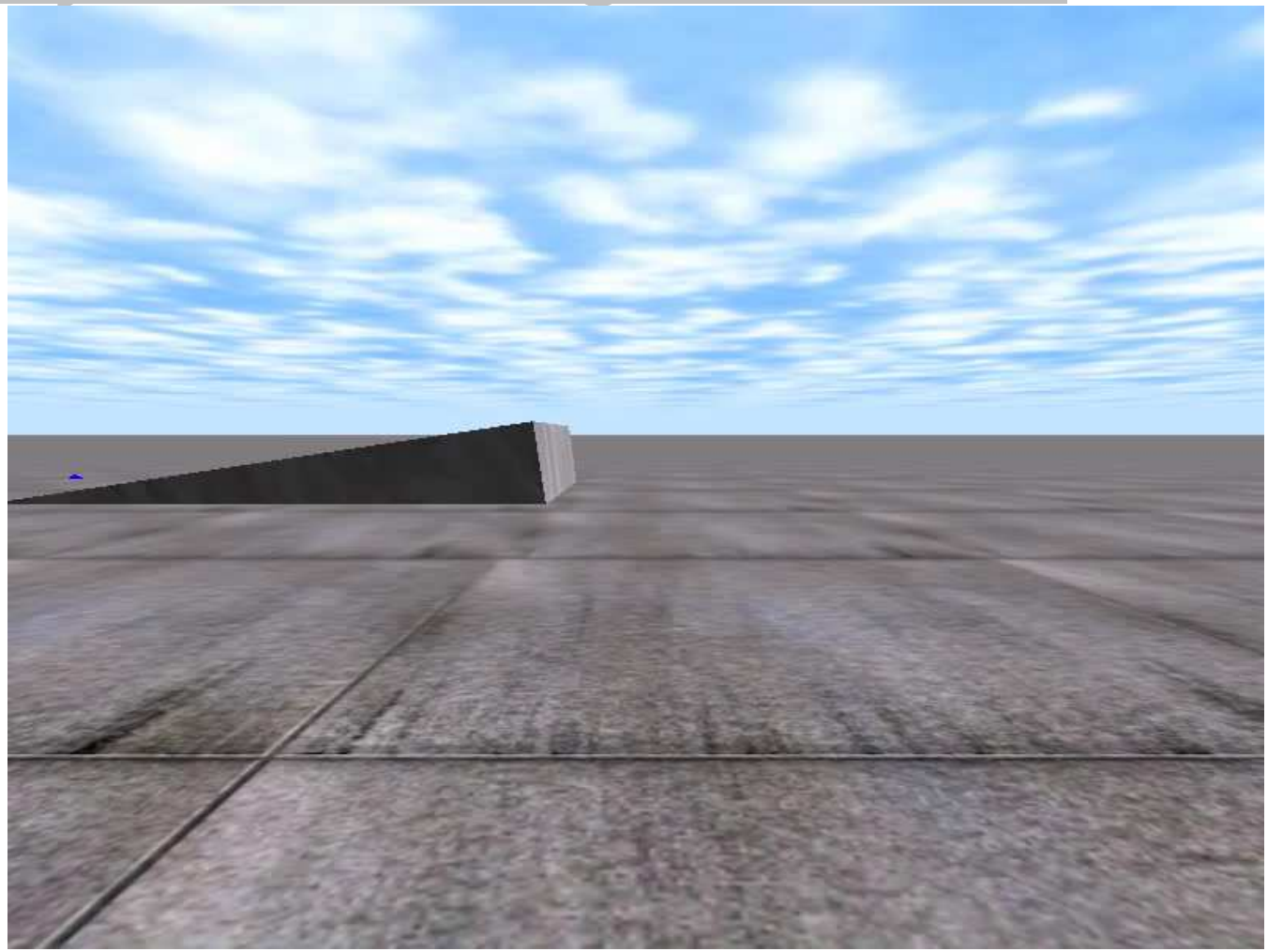
Juntas fixas prendem o suporte ao chão.



Open Dynamics Engine - ODE

A ODE permite efeitos espetaculares. Como no vídeo ao lado.

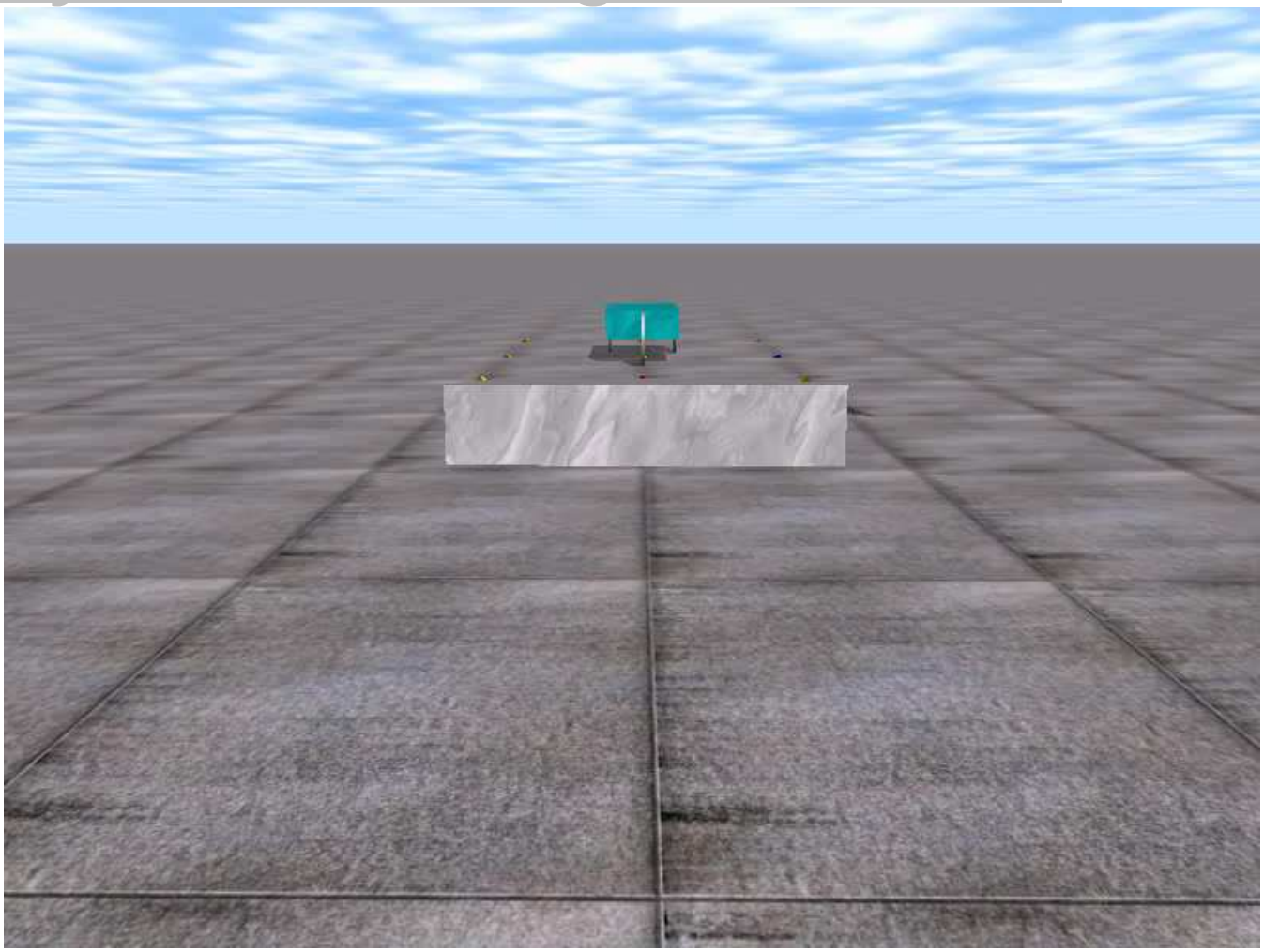
Repare que o carro reage as leis da dinâmica e da cinemática de forma realista.



Open Dynamics Engine - ODE

Reveja a cena sob outro ângulo.

A ODE permite que a cena possa ser visualizada sob qualquer ângulo.

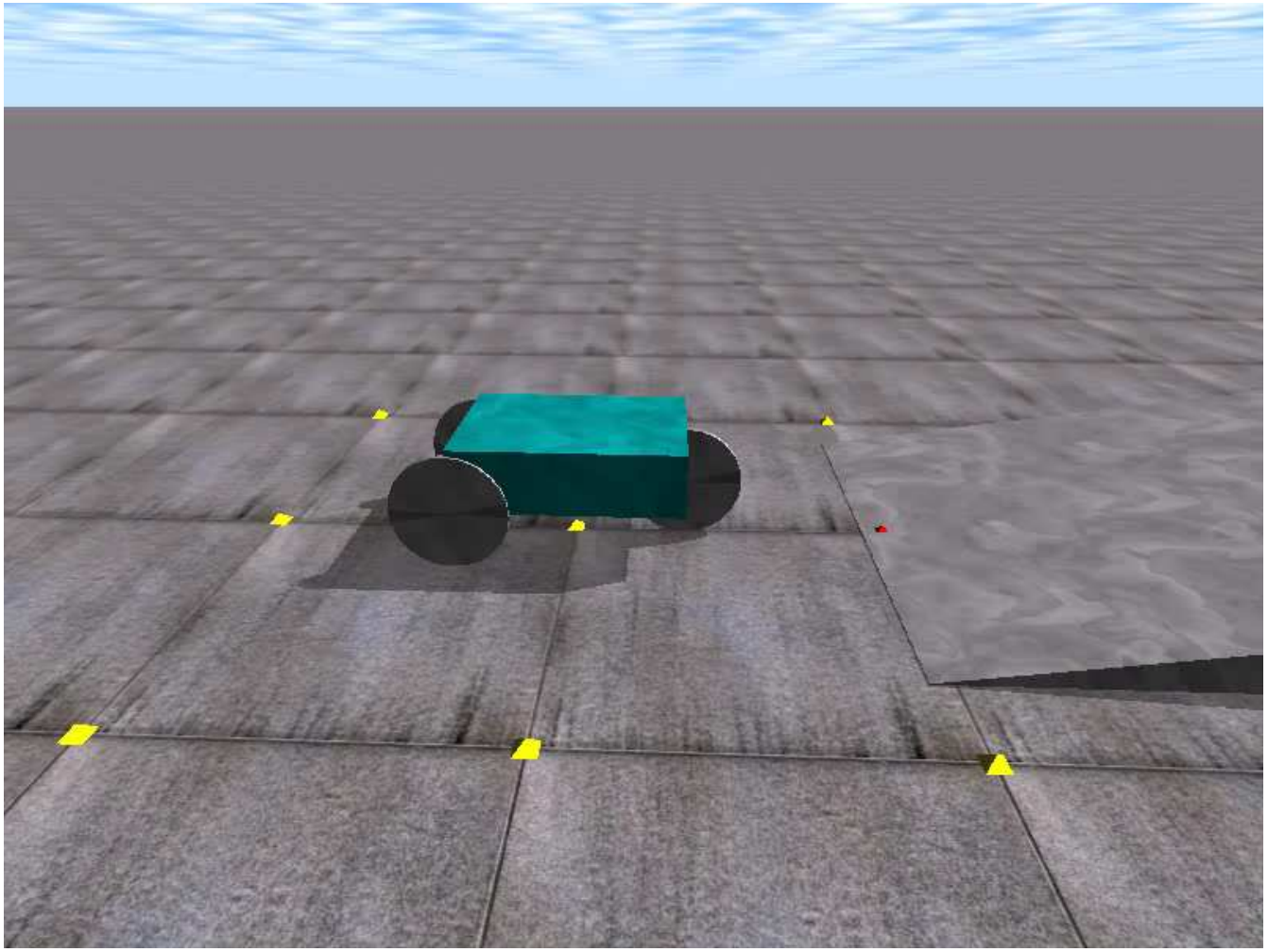


Open Dynamics Engine - ODE

Repare no momento em que o carro toca o chão.

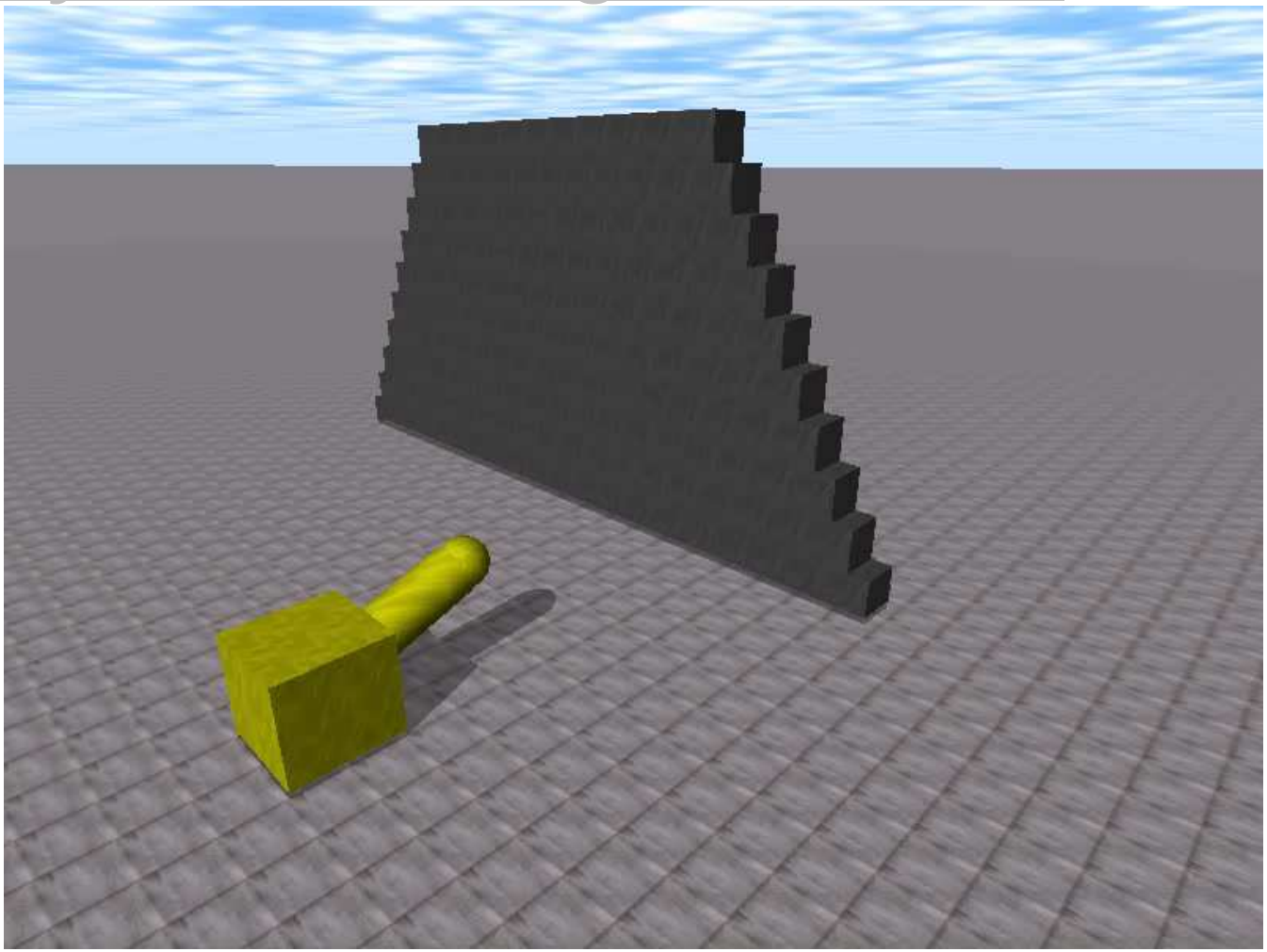
É como se o carro tivesse amortece-dores.

O impacto é bastante realista.



Open Dynamics Engine - ODE

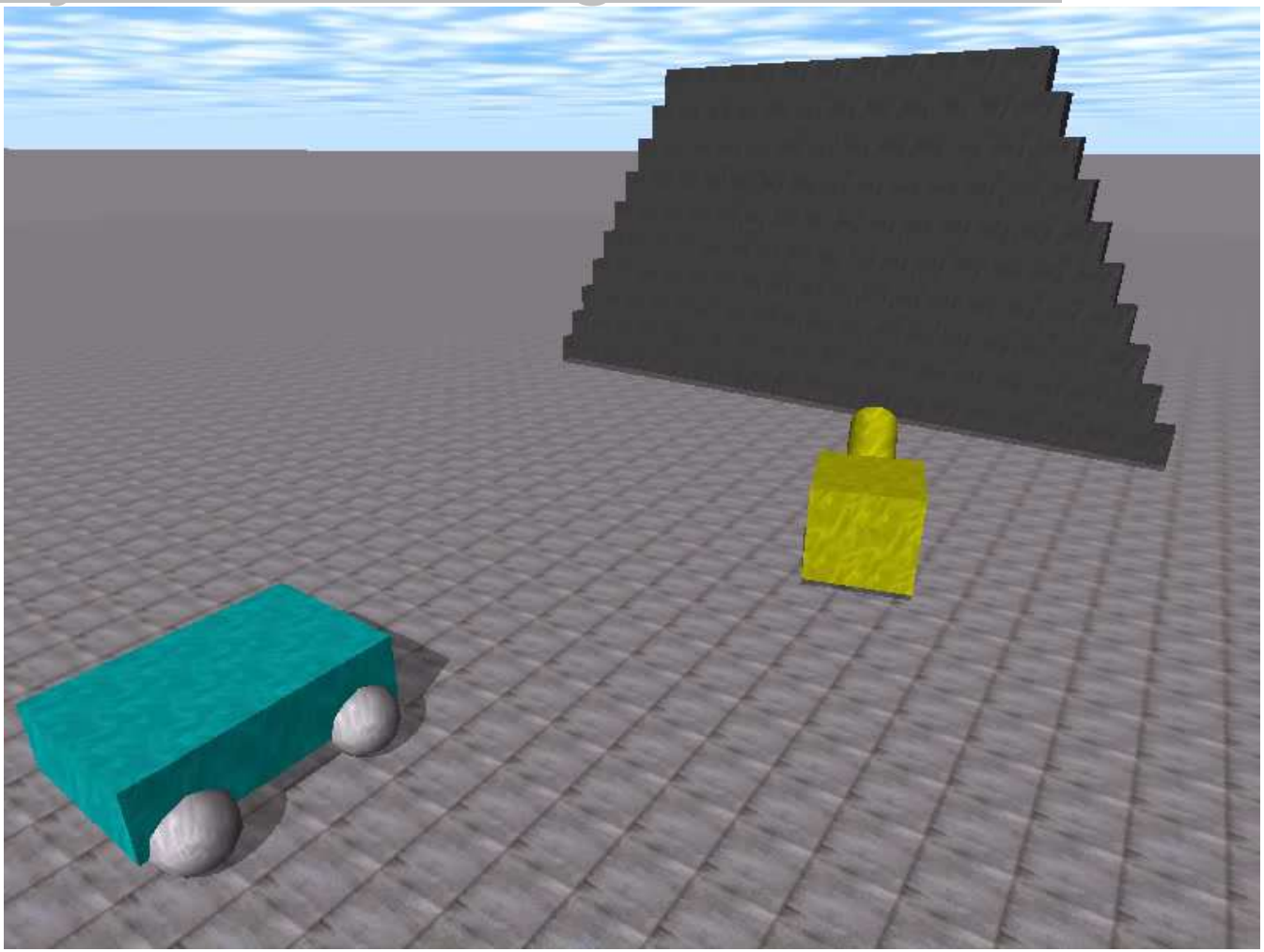
O tratamento de colisão funciona de forma realista mesmo quando existem muitos objetos na cena.



Open Dynamics Engine - ODE

E o melhor é que tudo acontece em tempo real.

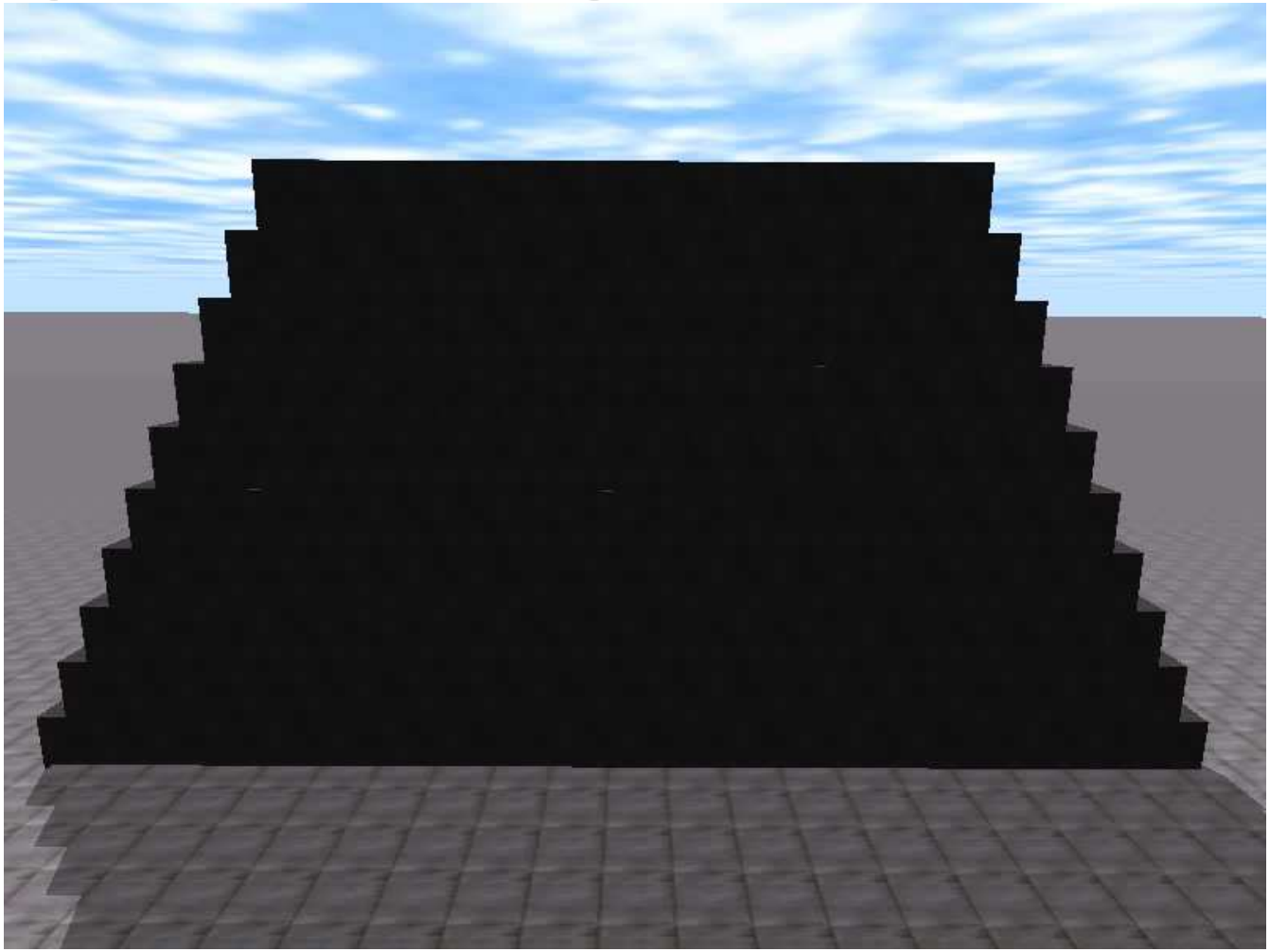
Ou seja, estes efeitos podem ser utilizados durante os jogos, e não apenas nos filmes introdutórios.



Open Dynamics Engine - ODE

Veja uma outra cena de impacto, desta vez pelo outro lado do muro.

A qualidade do vídeo é um pouco inferior devido ao processo de captura de codificação (DivX)



Controle Inteligente

- Fazer o agente interagir com o ambiente de simulação de forma automática, ou seja, dotar os agentes de autonomia
- Para que um agente seja autônomo, ele precisa perceber o ambiente (sensores), tomar decisões e realizar ações(Controle inteligente)
- As técnicas mais utilizadas para o controle inteligente são:
  - ⇨ Autômatos
  - ⇨ Redes Neurais Artificiais
  - ⇨ Algoritmos Genéticos

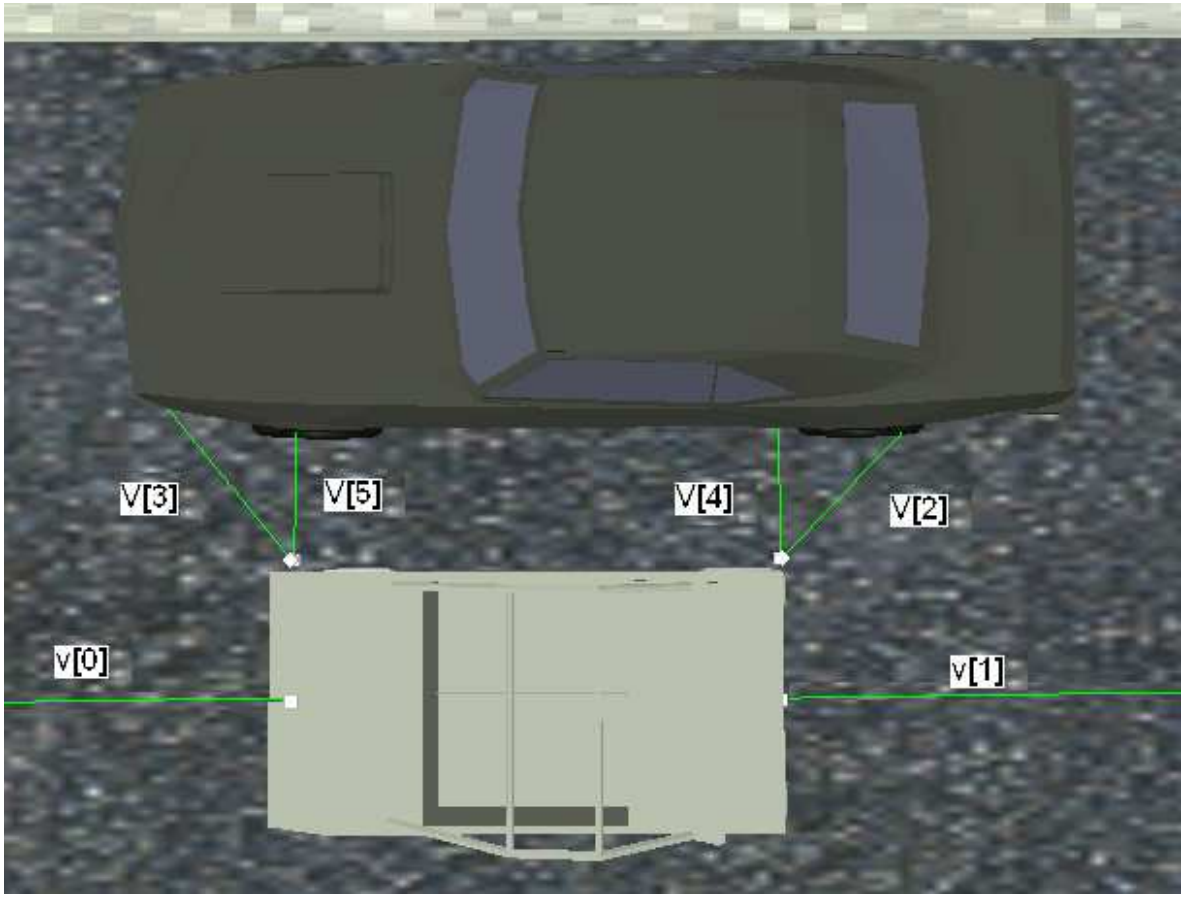
Aplicações práticas da RV

- Simulação de veículos
  - ⇨ SimRob3D
  - ⇨ Seva3D
- Simulação de Robôs
  - ⇨ LegGen
  - ⇨ Juice
  - ⇨ Webots
- Jogos

Simulador SEVA3D

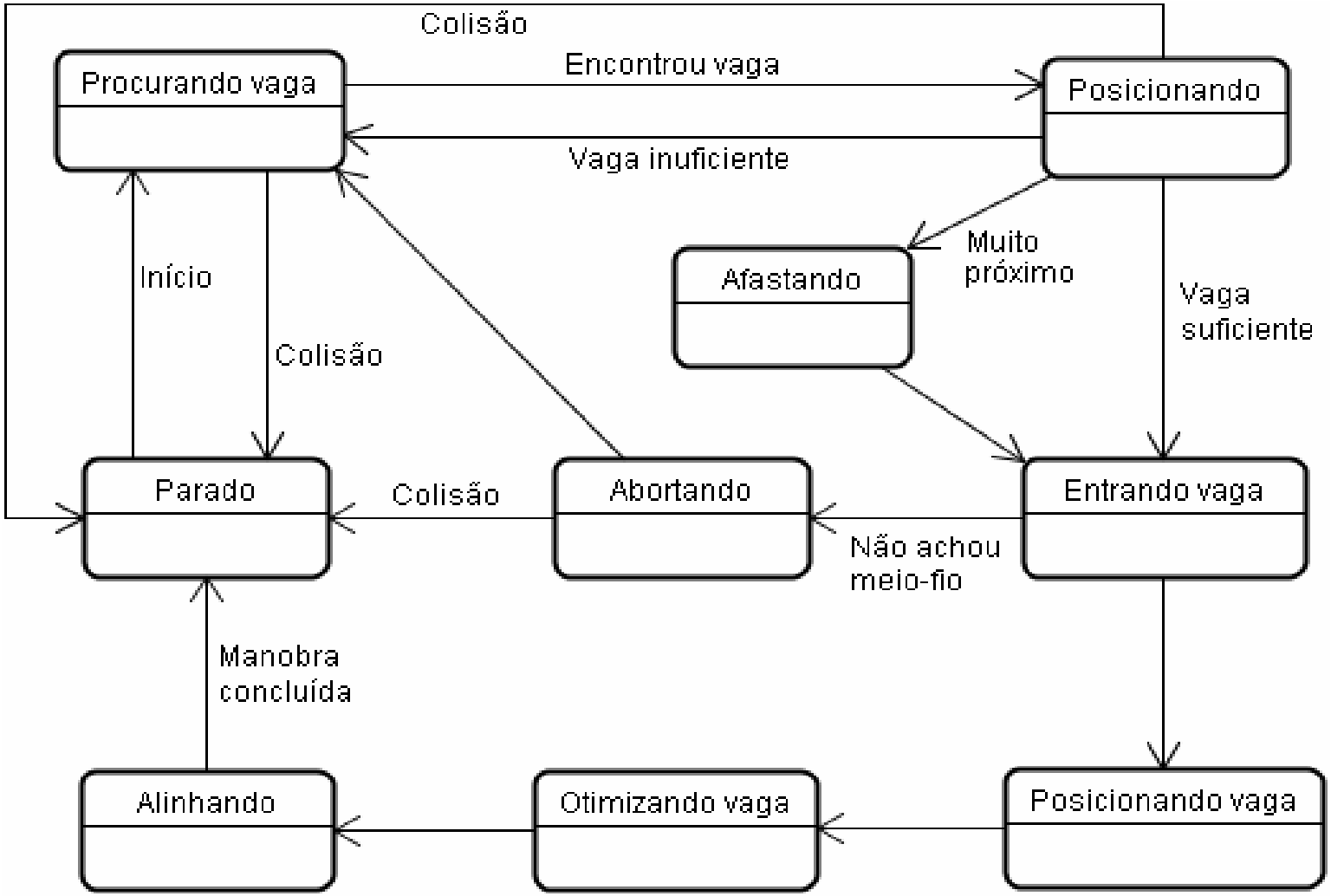
- Robô do tipo carro equipado com seis sensores de proximidade do tipo sonar
- Realiza o estacionamento de um carro em uma vaga paralela de forma autônoma
- Utiliza um ambiente tridimensional realístico
- Modelagem de ruídos sensoriais
- Controle realizado através de um autômato finito ou através de uma Rede Neural
- O simulador permite a visualização do veículo e do estado dos sensores

Modelo Sensorial

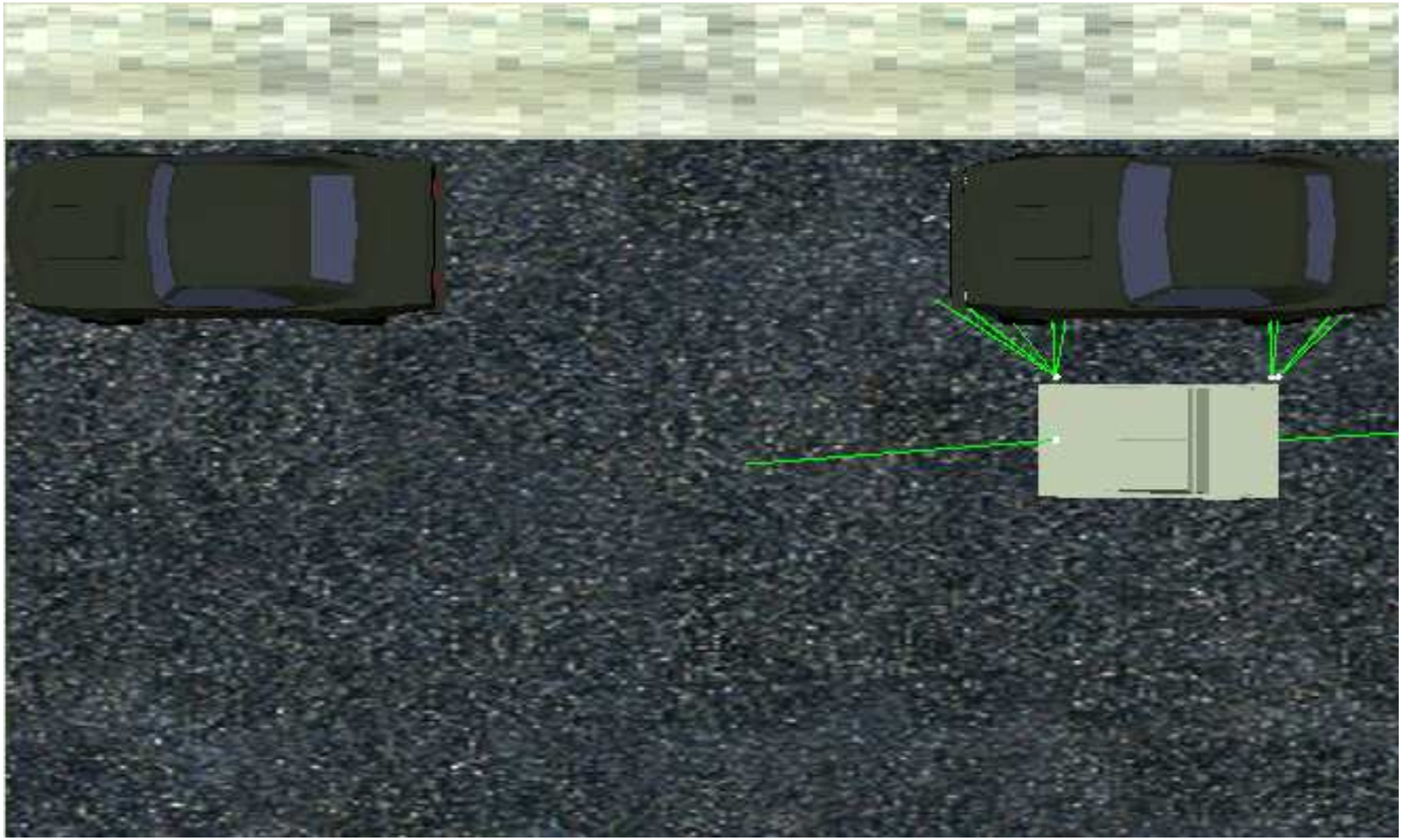


- Odômetro
- Seis sensores do tipo sonar posicionados estrategicamente
- Utilização da técnica de RayCast
- Modelagem de ruídos sensoriais de forma estocástica

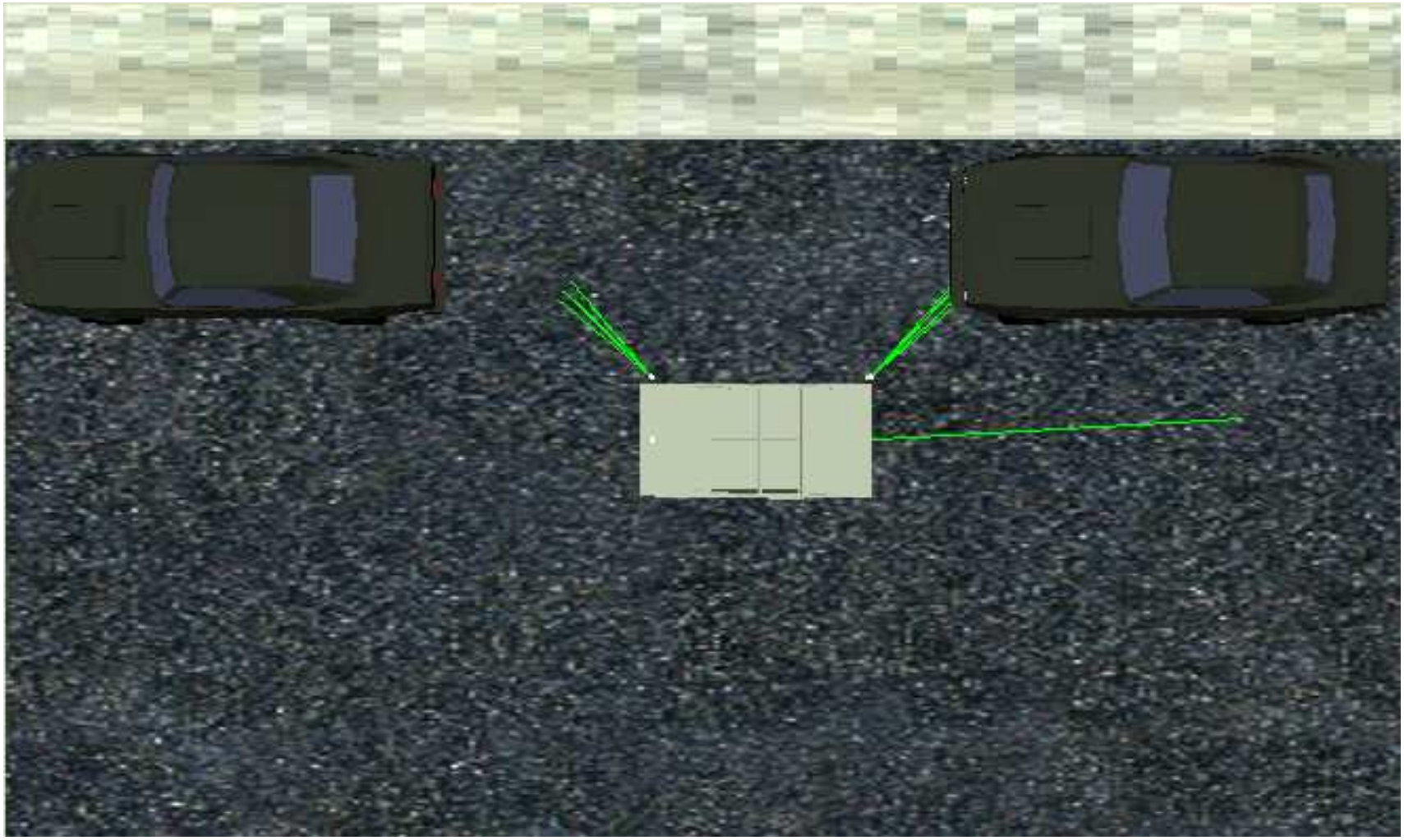
Autômato finito: Entra na vaga



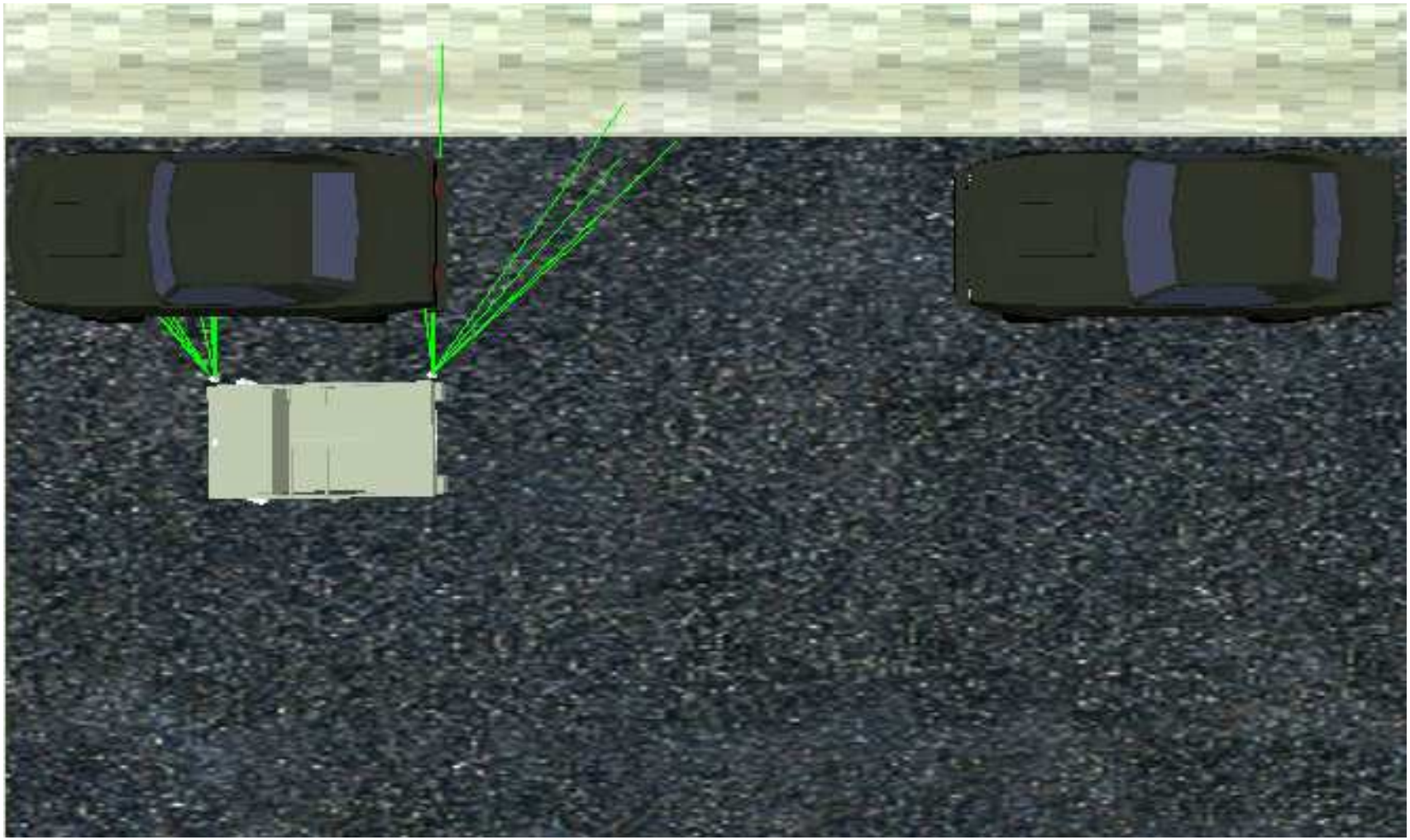
Estado: Procurando Vaga



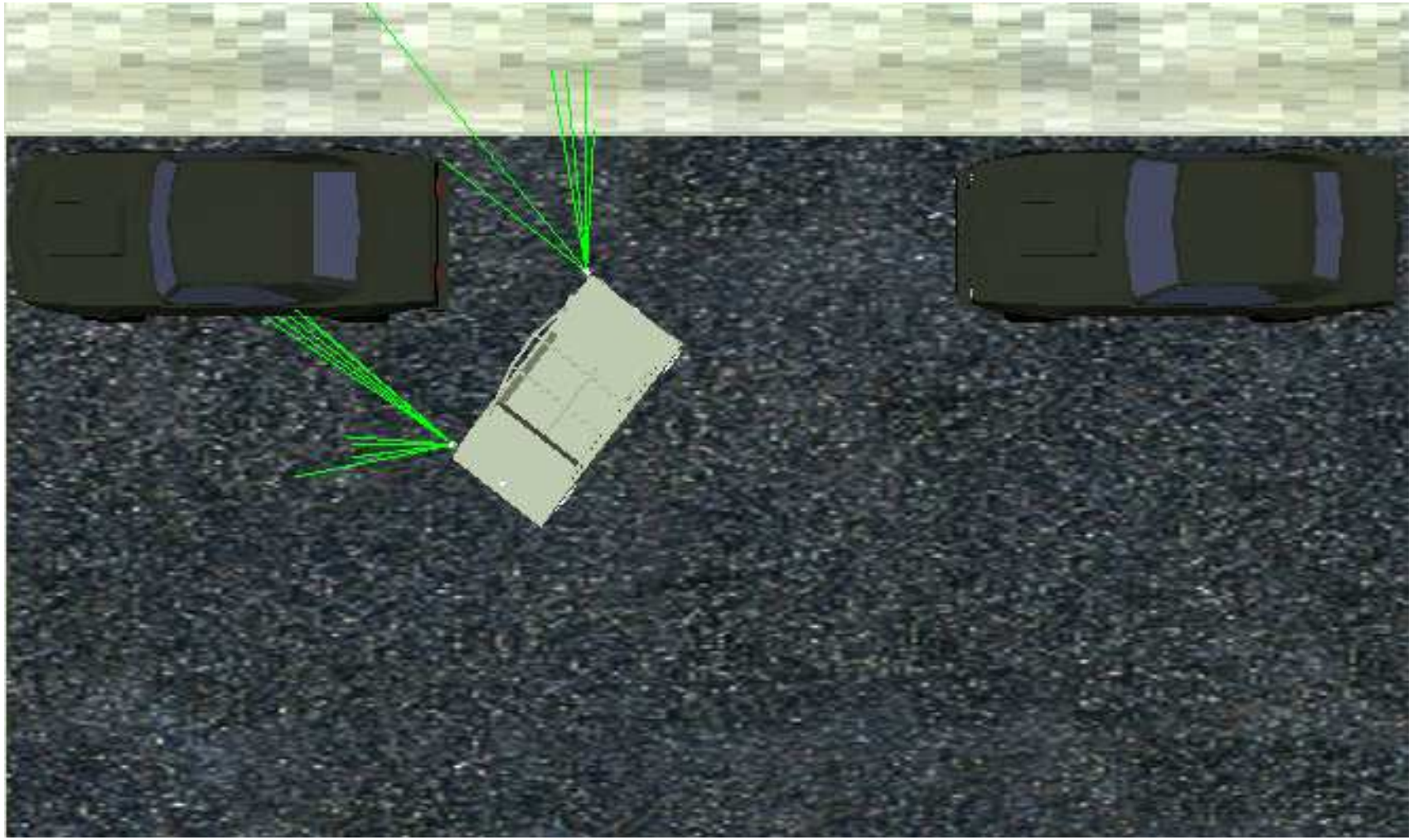
Estado: Posicionando



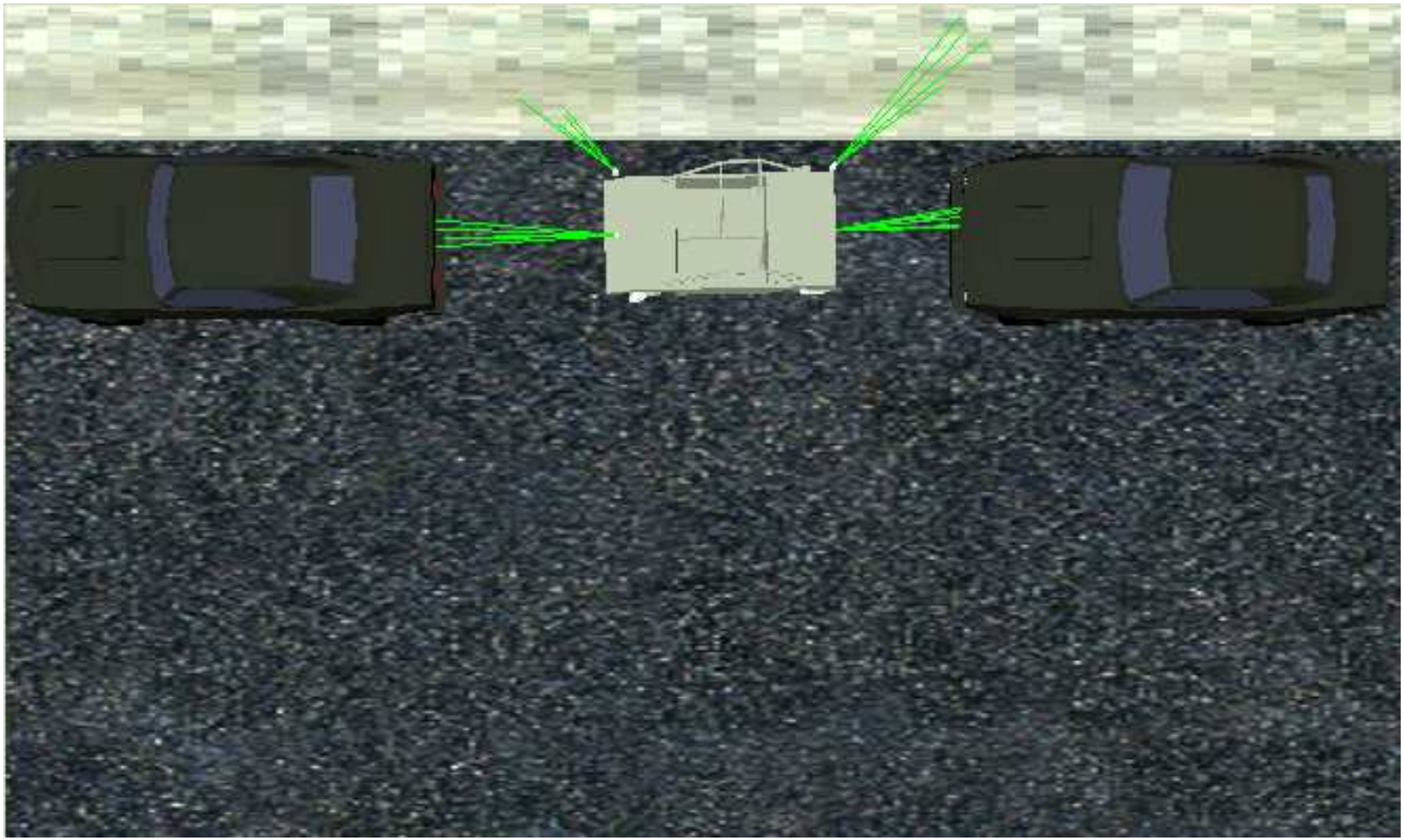
Estado: Entrando Vaga



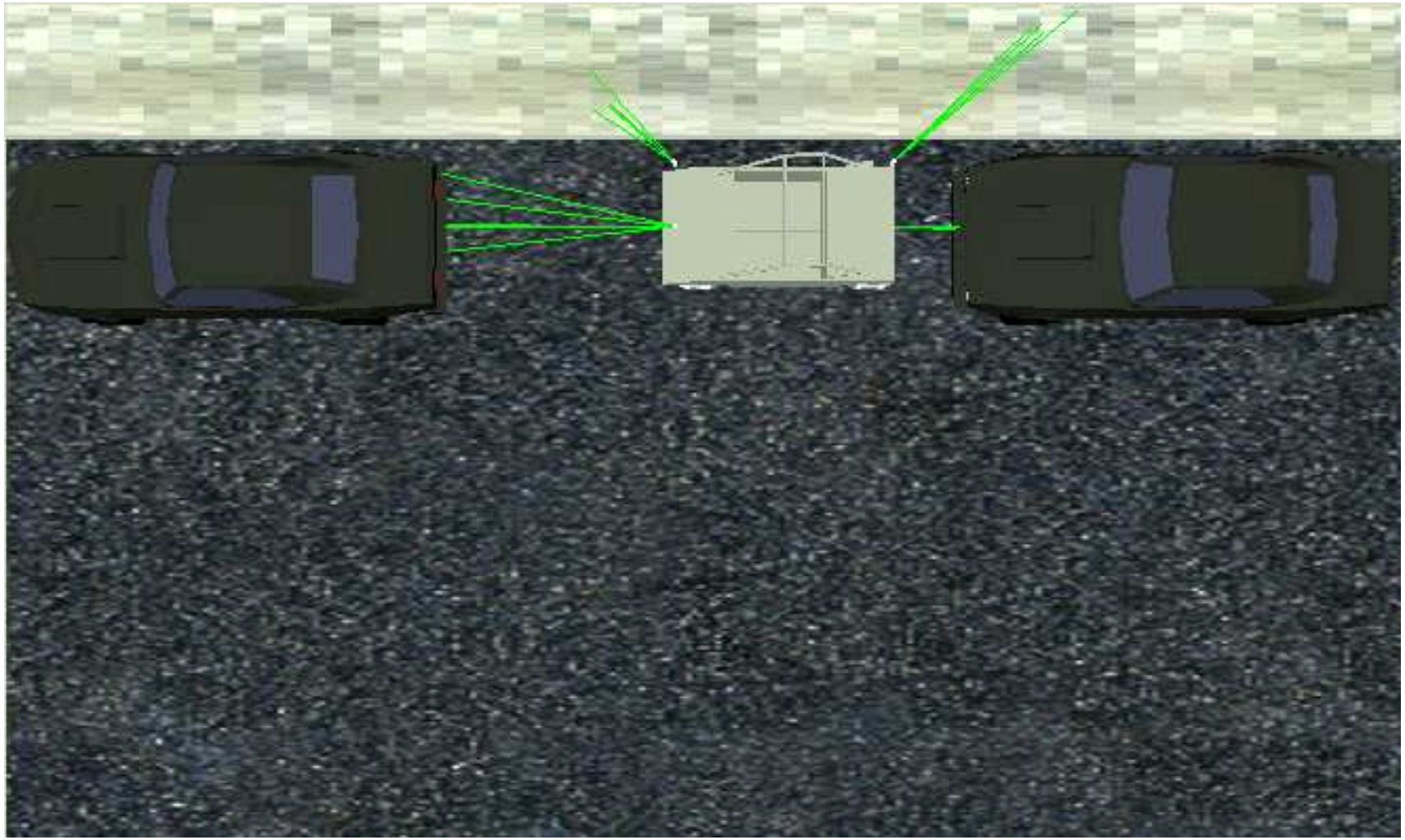
Estado: Posicionando vaga



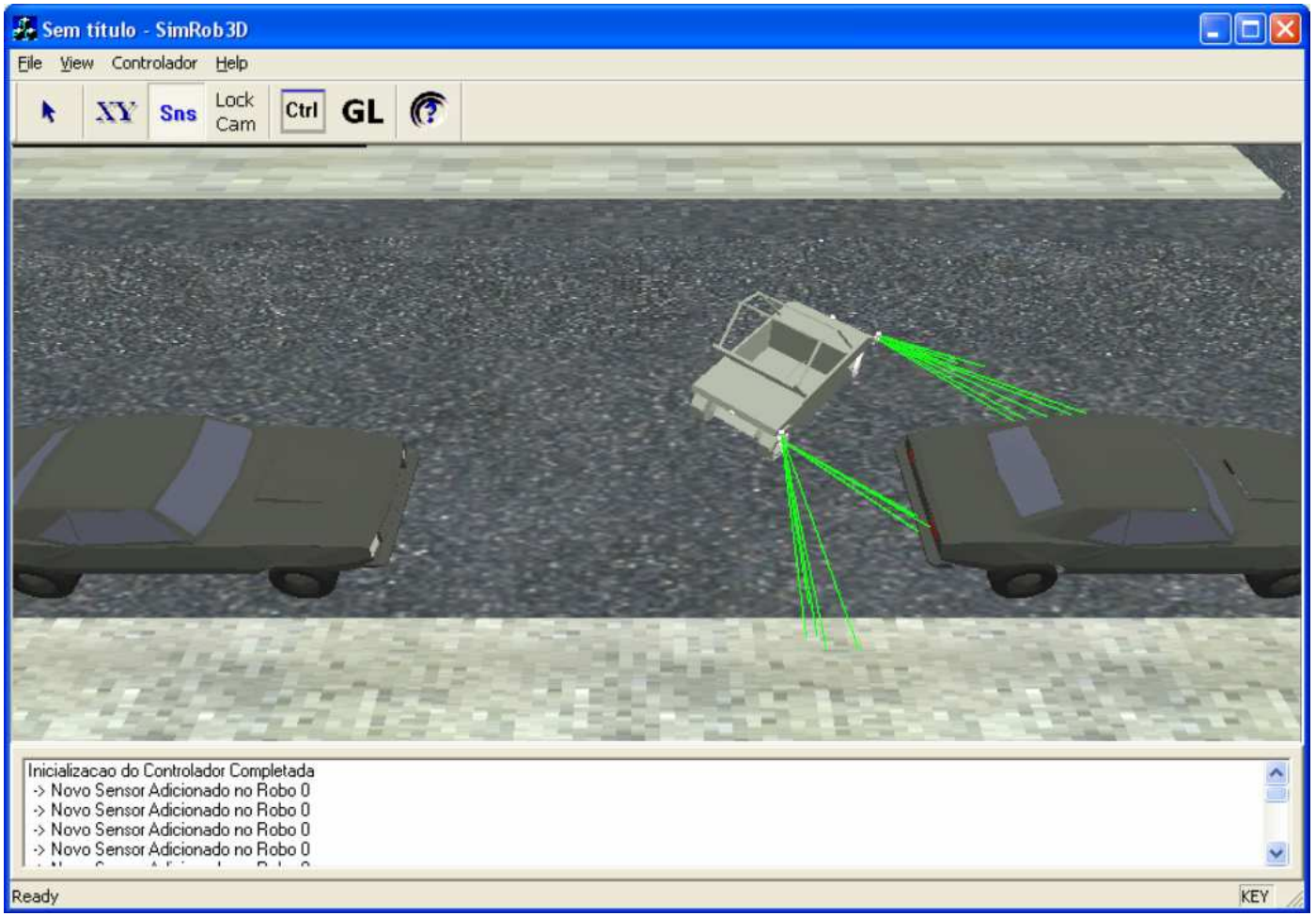
Estado: Otimizando Vaga



Estado: Alinhando



Simulador SimRob3D



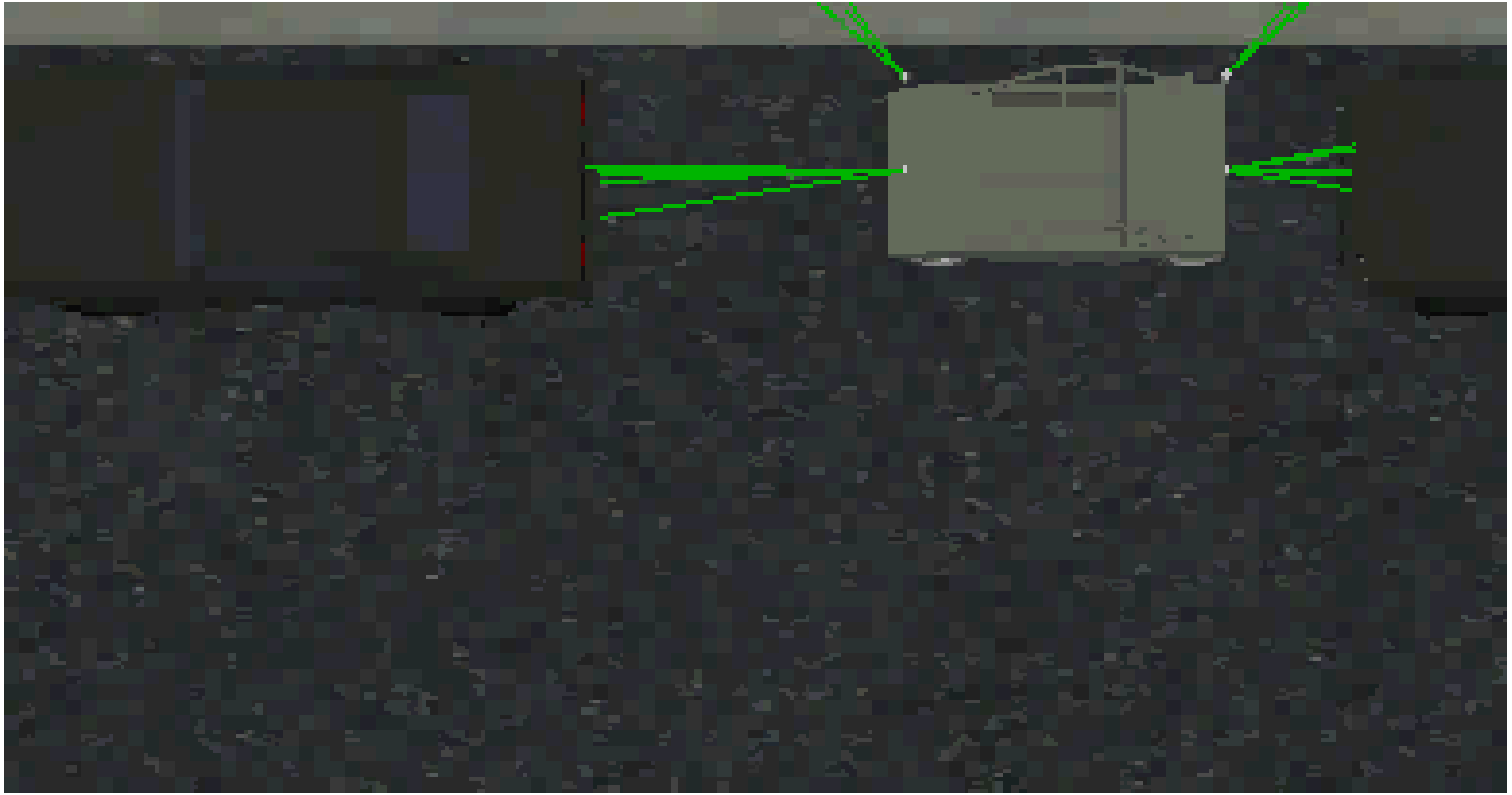
- Trabalha com modelos de ambiente 3DS
- Implementa modelo sensorial, cinemático e controle do veículo
- Permite visualizar a manobra

Veículo modelado

Resultados obtidos

Resultados obtidos

Resultados obtidos

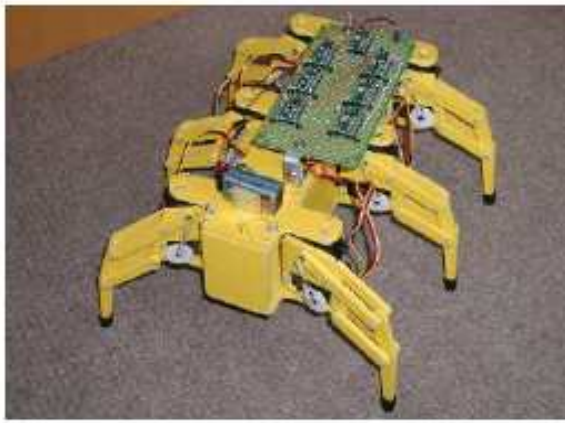


Perspectivas futuras

- Implementação do sistema em um protótipo real de tamanho reduzido...

- Implementação no veículo real...

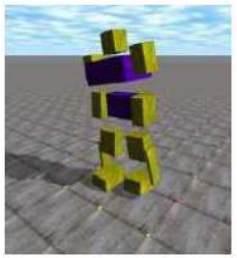
Robôs móveis autônomos com pernas



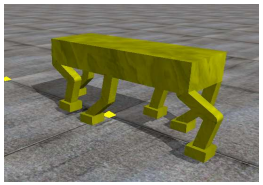
Robô Lynxmotion Hexapod II

Robô Genghis-II

(a) Robô real



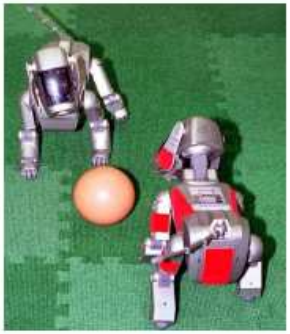
(b) Robô simulado



The Sony Dream Robot in the real world

The Sony Dream Robot simulated into Webots

(a)



(b)

Figura 27: Modelos de Robôs Sony Aibo [95]

(a) Honda Asimo

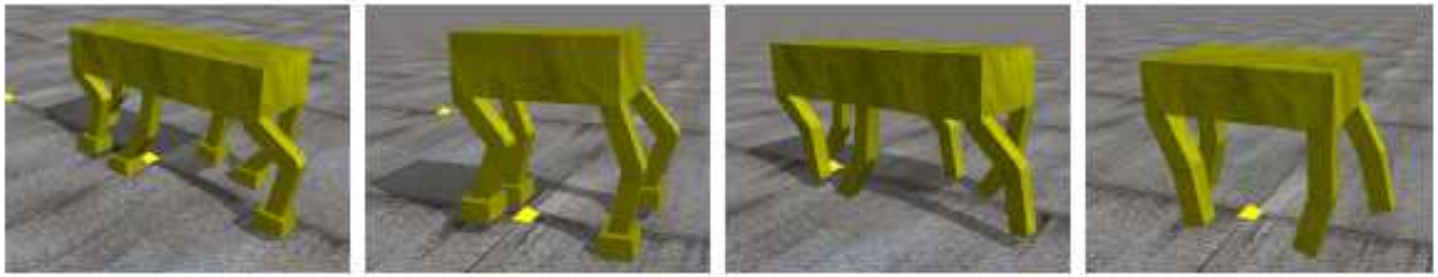
(b) Sony SDR-4X

(c) Kawada H6

(d) Fujitsu HOAP-2

Simulador LegGen

- Robôs que se locomovem utilizando as pernas

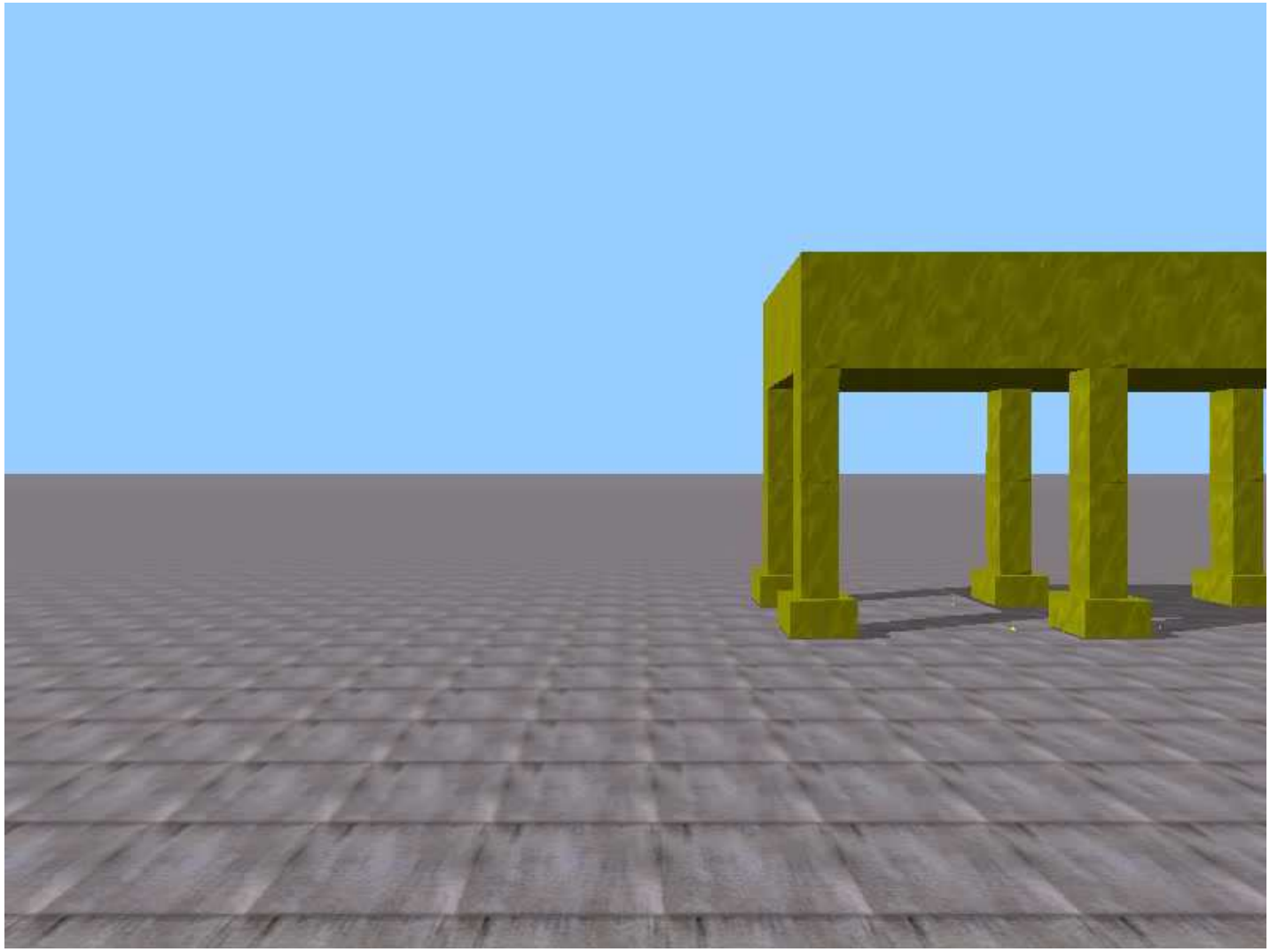


(a) HexaL3J (b) TetraL3J (c) HexaL2J (d) TetraL2J

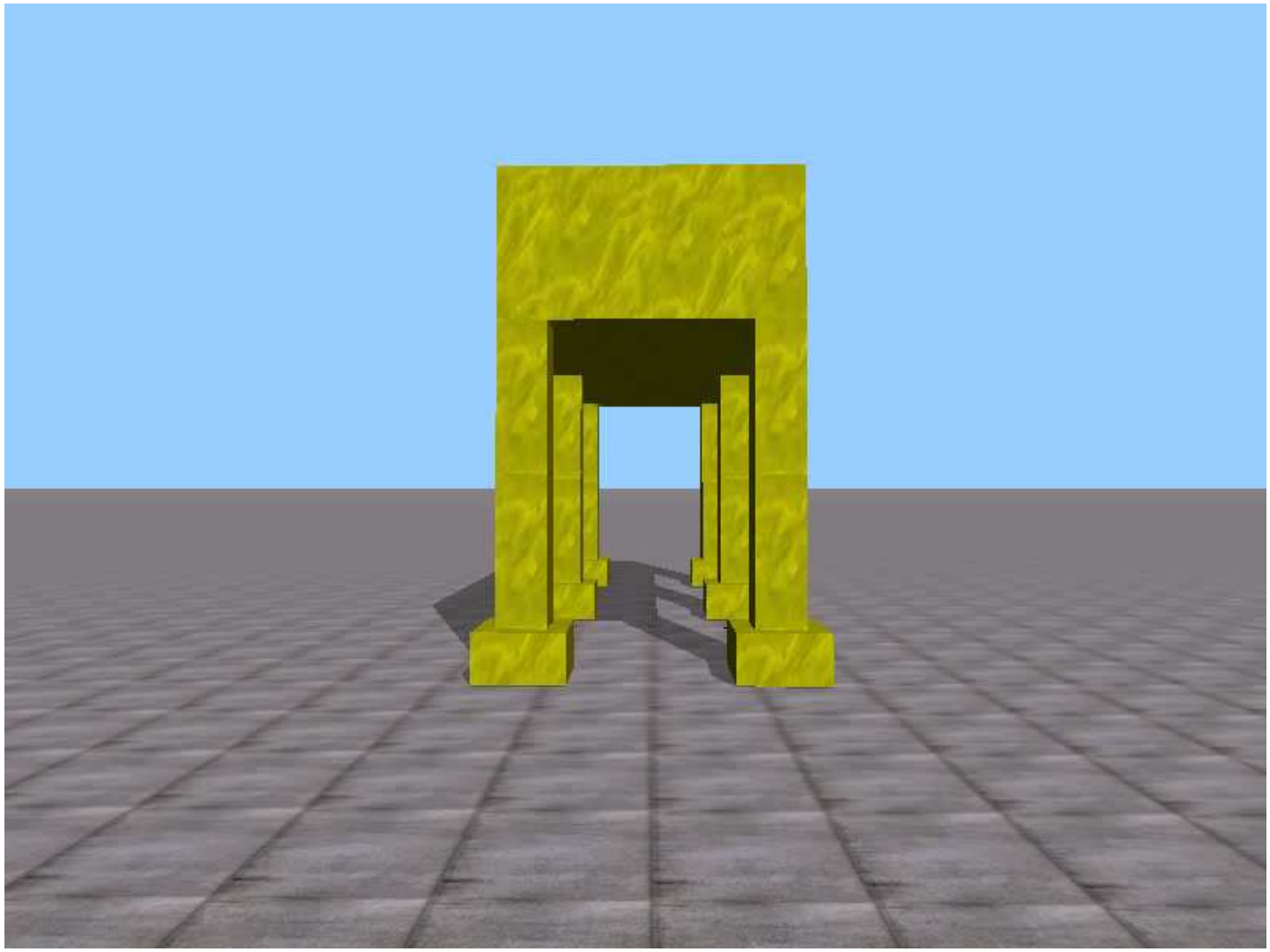
Modelos de robôs utilizados nas simulações

- Simulação em Ambiente Virtual 3D com uso da biblioteca ODE (simulação física)
- Utilização de algoritmos genéticos para o controle inteligente dos robôs

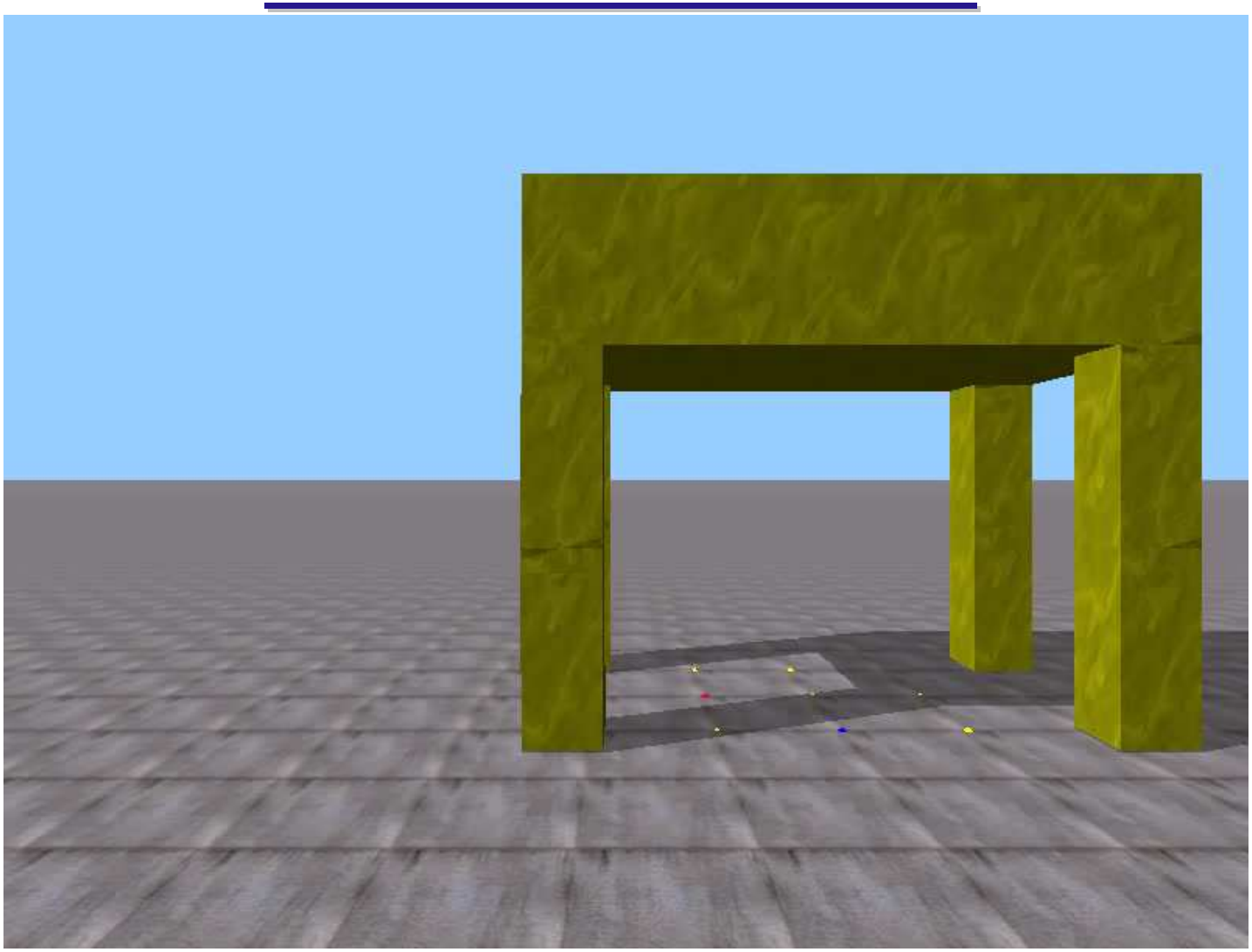
Resultados obtidos



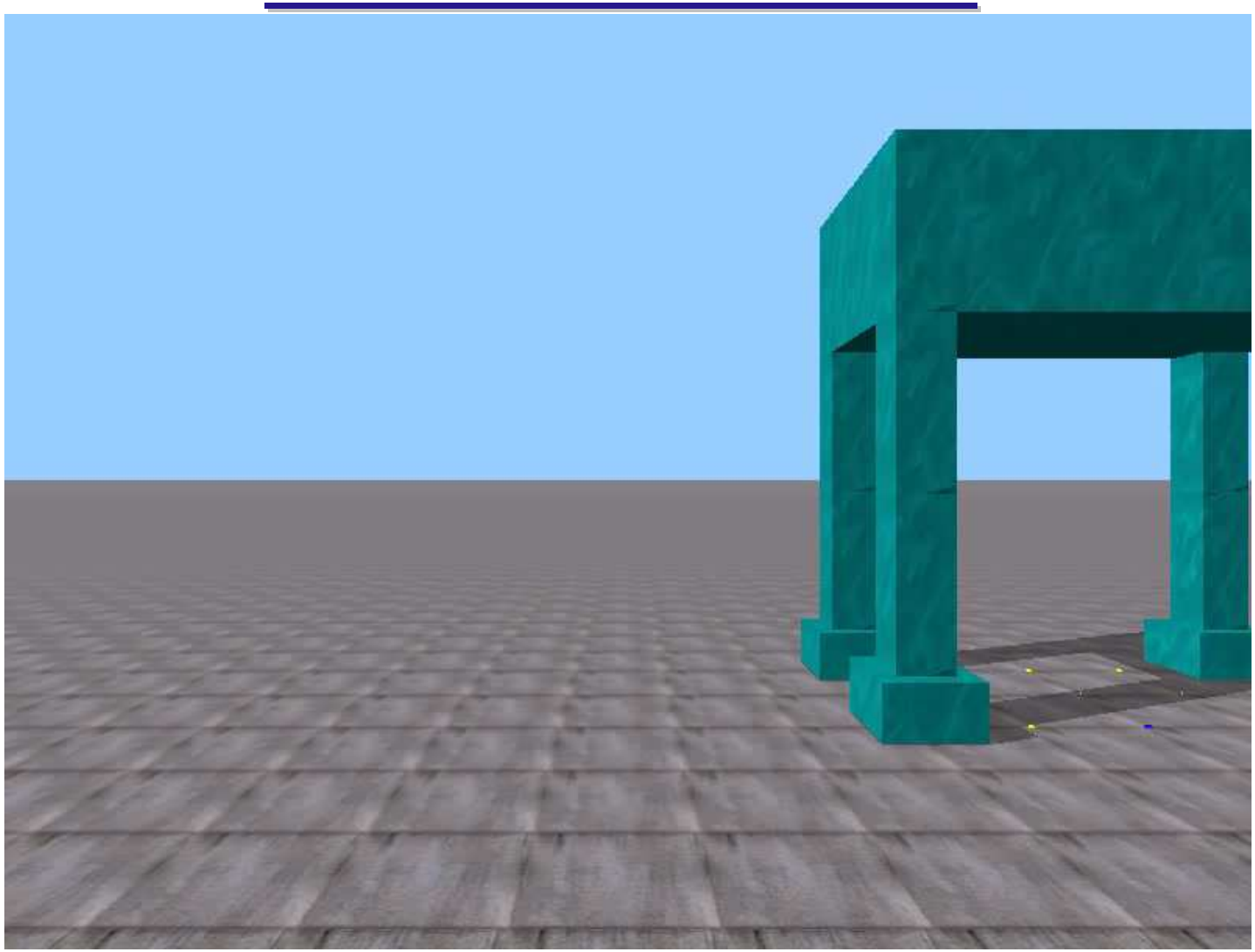
Resultados obtidos



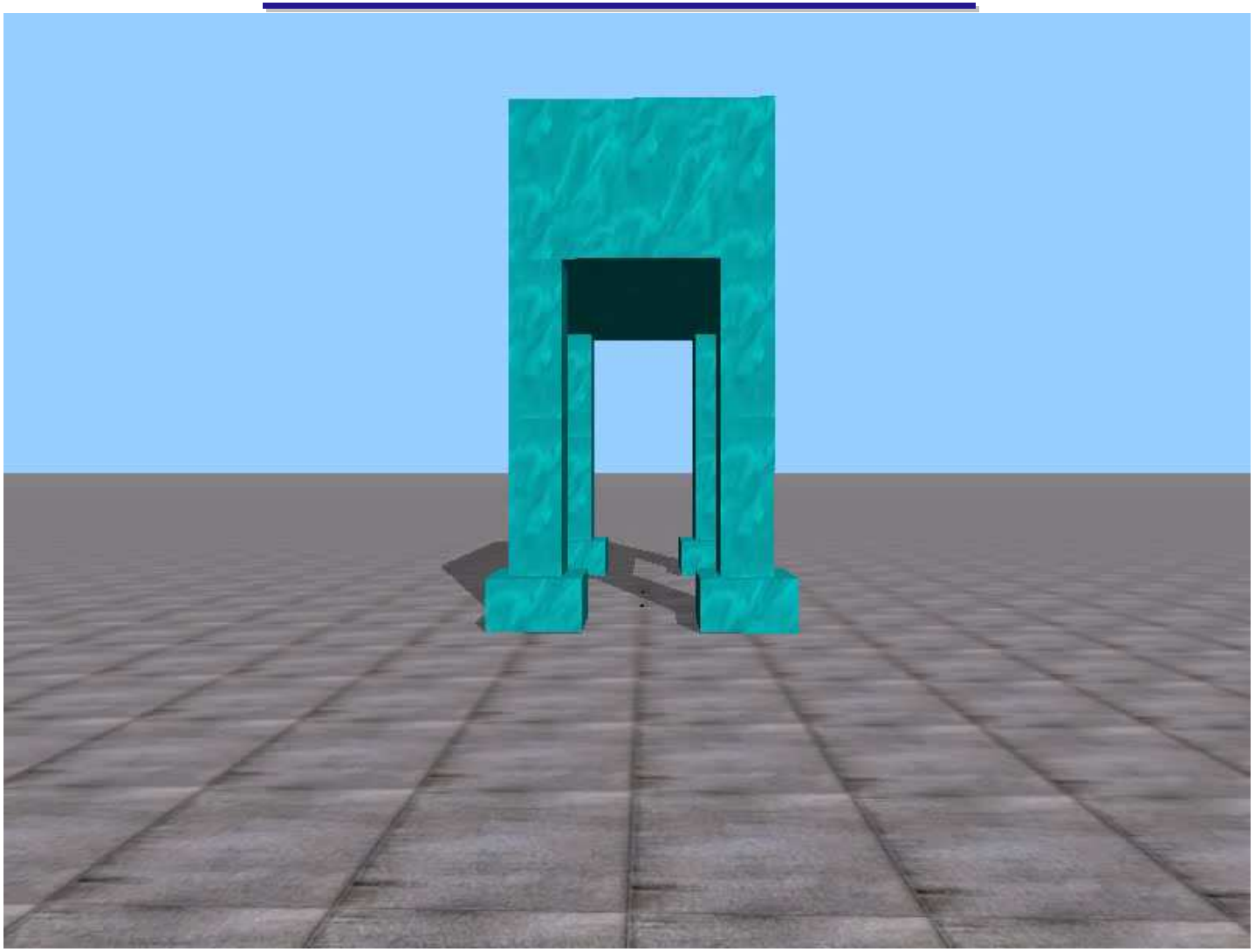
Resultados obtidos



Resultados obtidos



Resultados obtidos



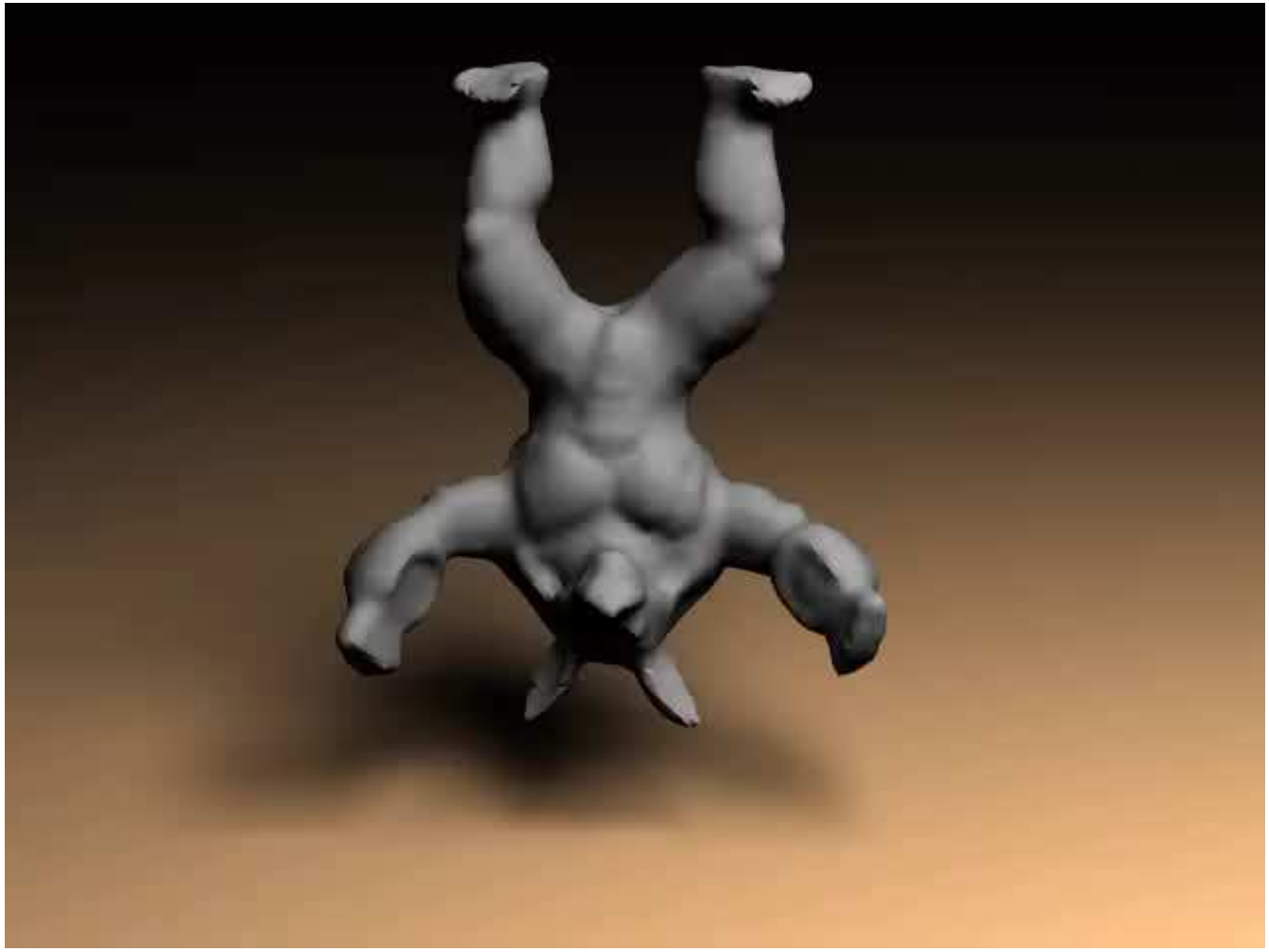
Jogos

- Vários jogos estão sendo desenvolvidos utilizando a biblioteca ODE, como por exemplo o jogo FragFist
  - ⇨ Vídeo: ode-videos\fragfist_trailer_gc05.avi

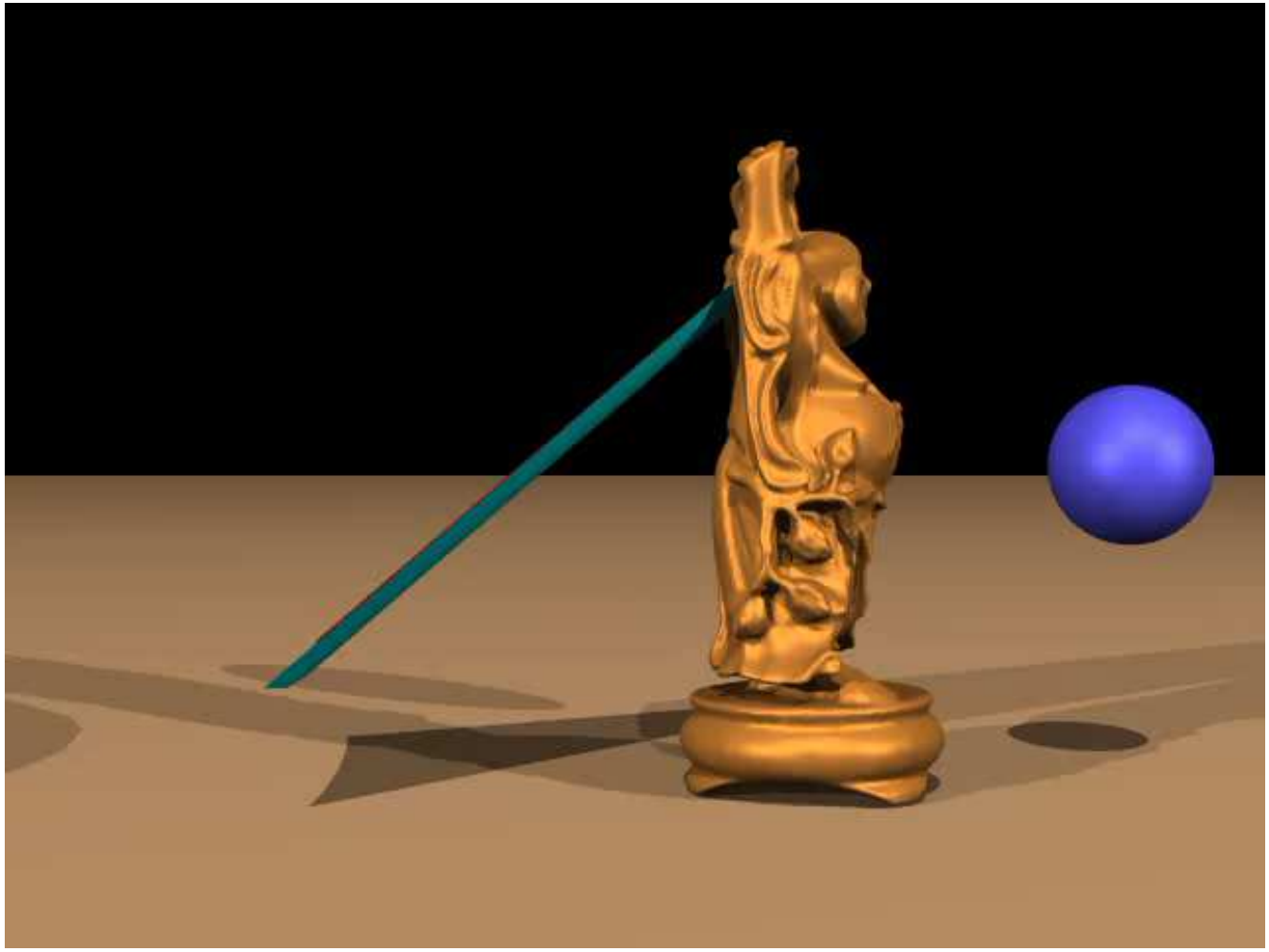
Simulação de corpos deformáveis

- A biblioteca ODE não dá suporte a simulação de corpos deformáveis, mas existem alguns softwares proprietários que podem ser utilizados para tal finalidade
- A maioria das abordagens se baseia no conceito de massa-mola da computação gráfica
- A deformação é necessária quando se modela elementos como roupas e cabelos – os efeitos precisam ser realista

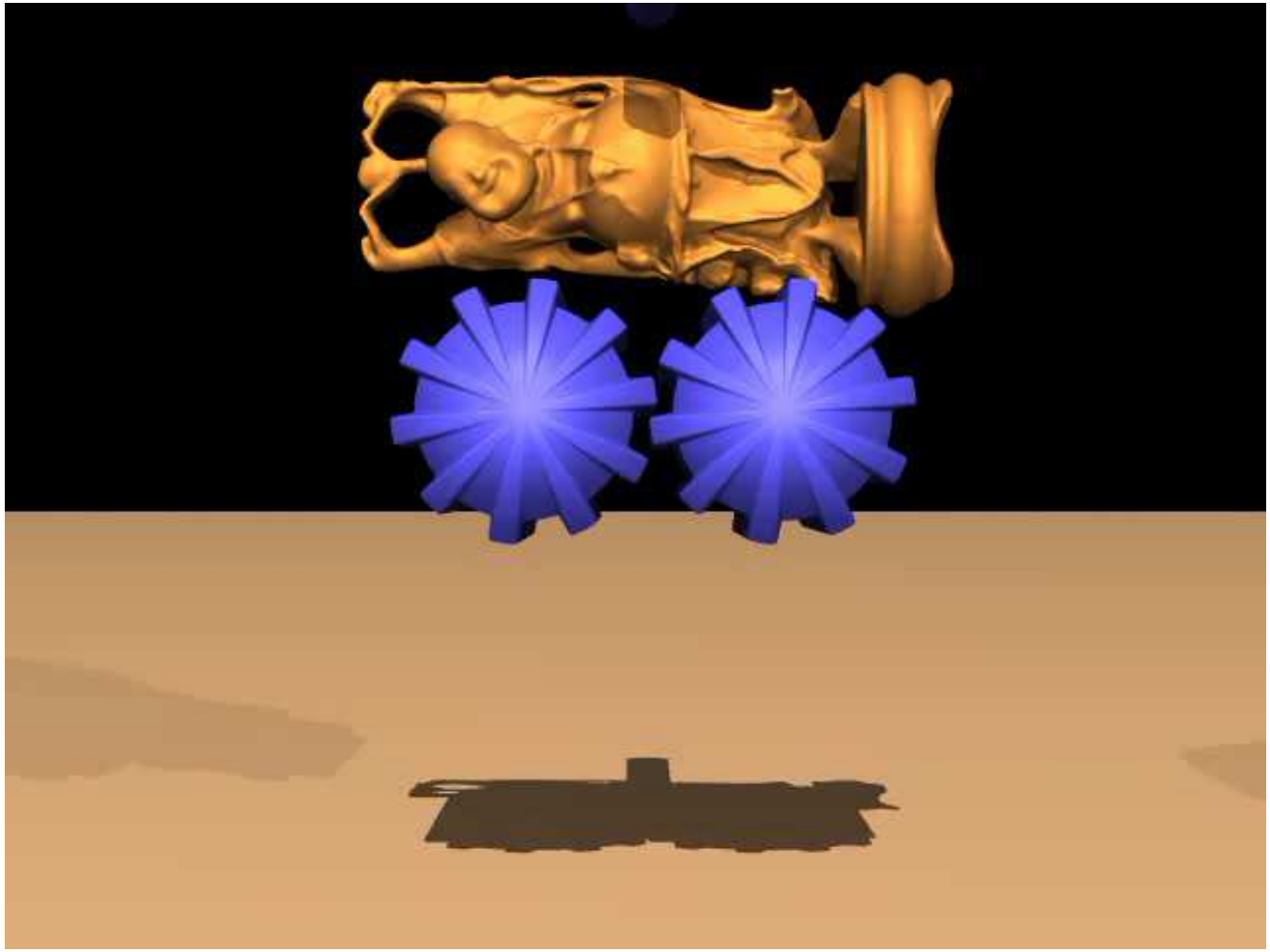
Simulação de corpos deformáveis



Simulação de corpos deformáveis



Simulação de corpos deformáveis



Simulação de corpos deformáveis

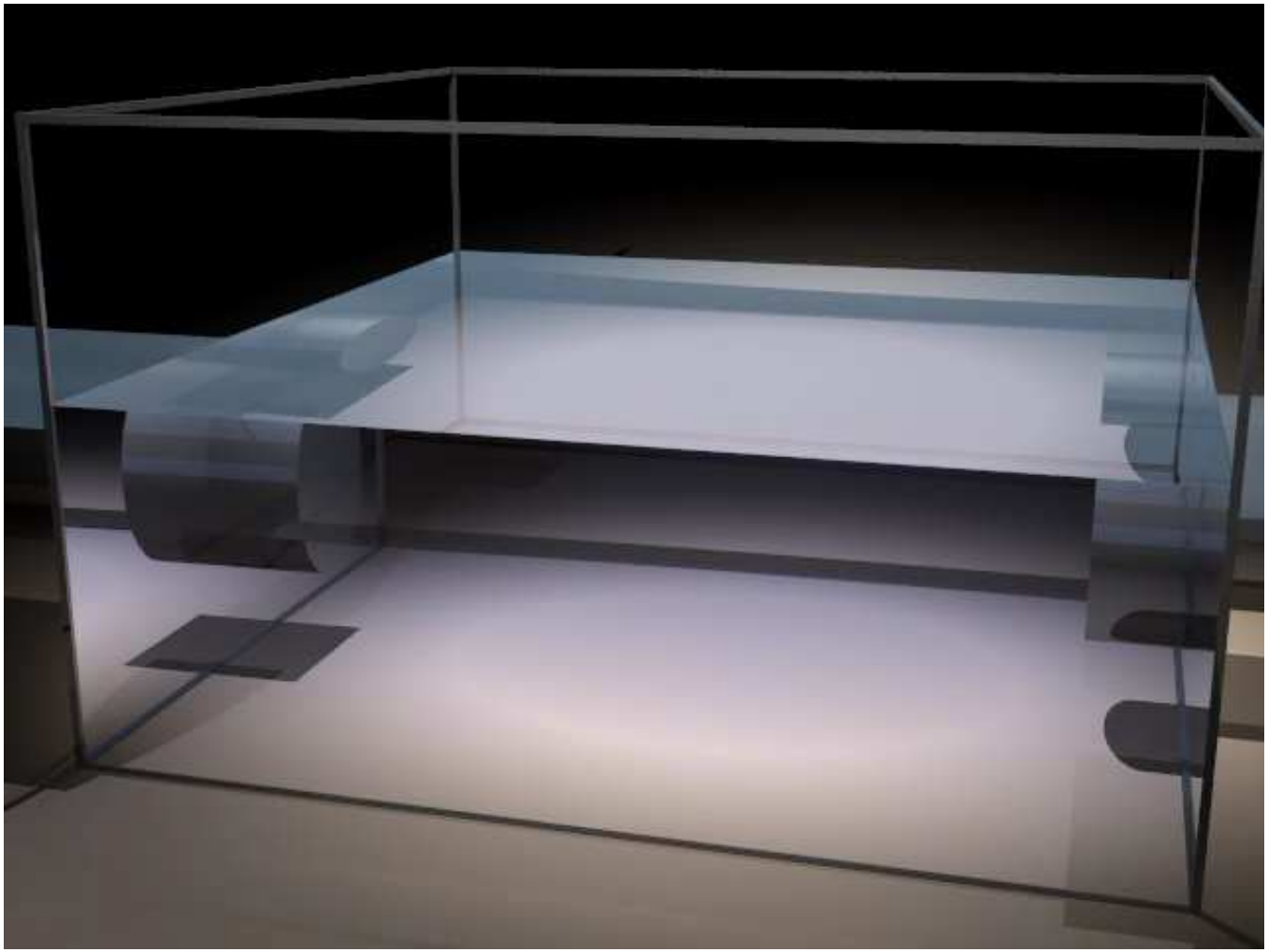
- Deformação de tecidos:
  - ⇨ ode-videos\cloth0.avi
  - ⇨ ode-videos\cloth1.avi
  - ⇨ ode-videos\cloth2.avi

Simulação de partículas

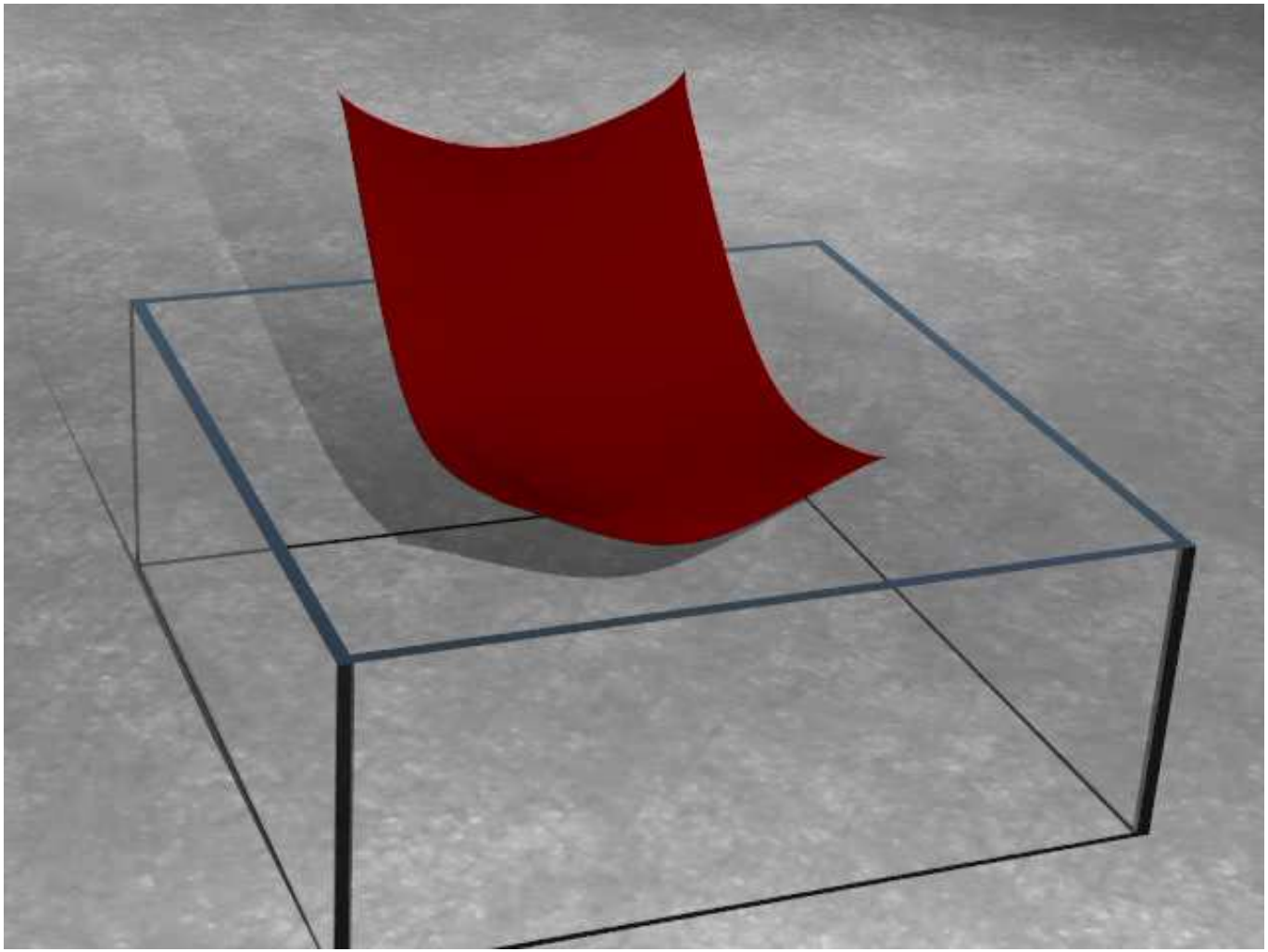
- Permite obter efeitos interessantes em elementos como:
  - ⇨ Líquidos
  - ⇨ Fogo
  - ⇨ Fumaça e névoa
  - ⇨ Bolhas
- A biblioteca OSG permite que sejam obtidos alguns efeitos de partículas em tempo real

OSG – Efeitos de partículas

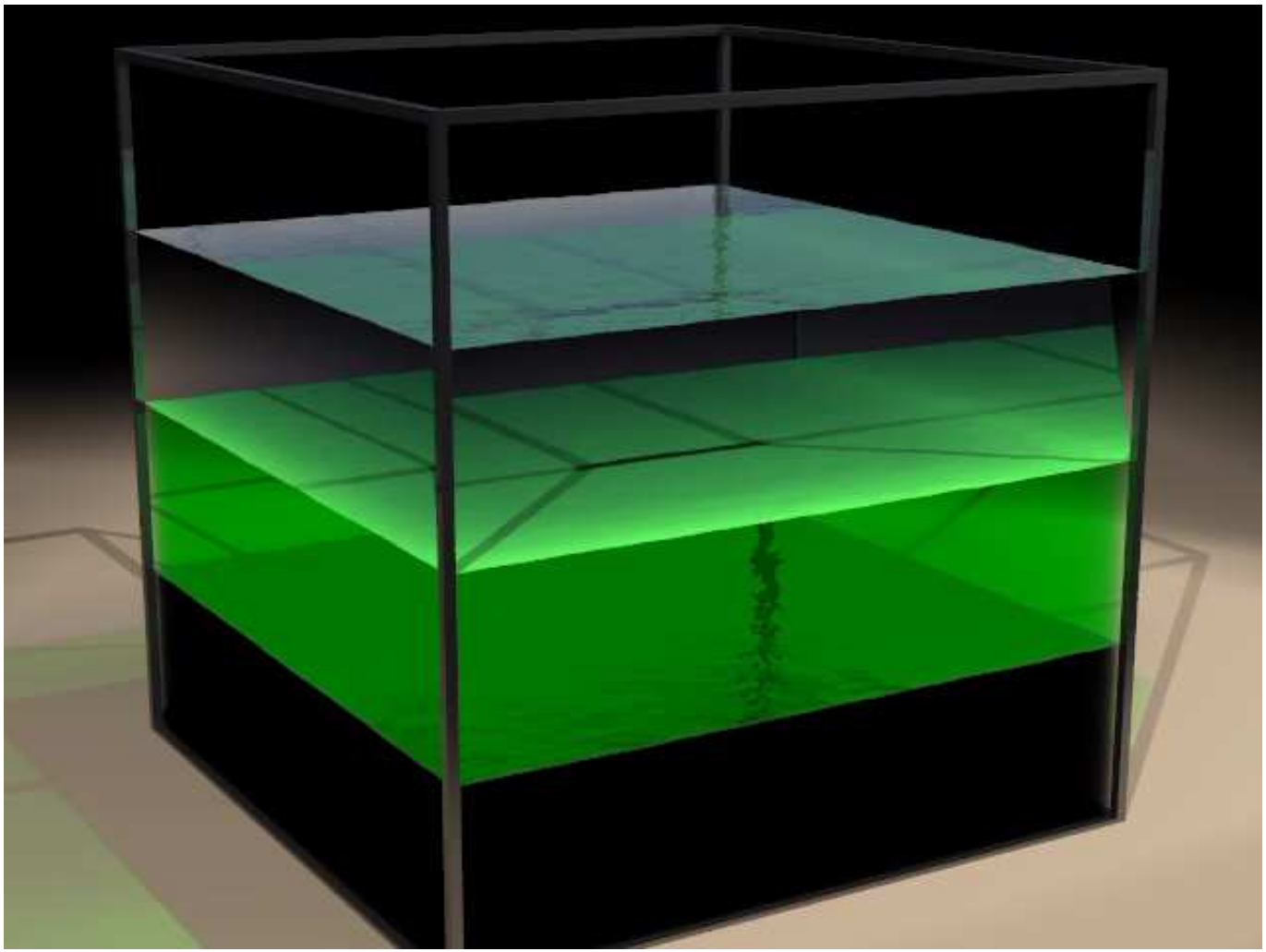
Simulação de partículas



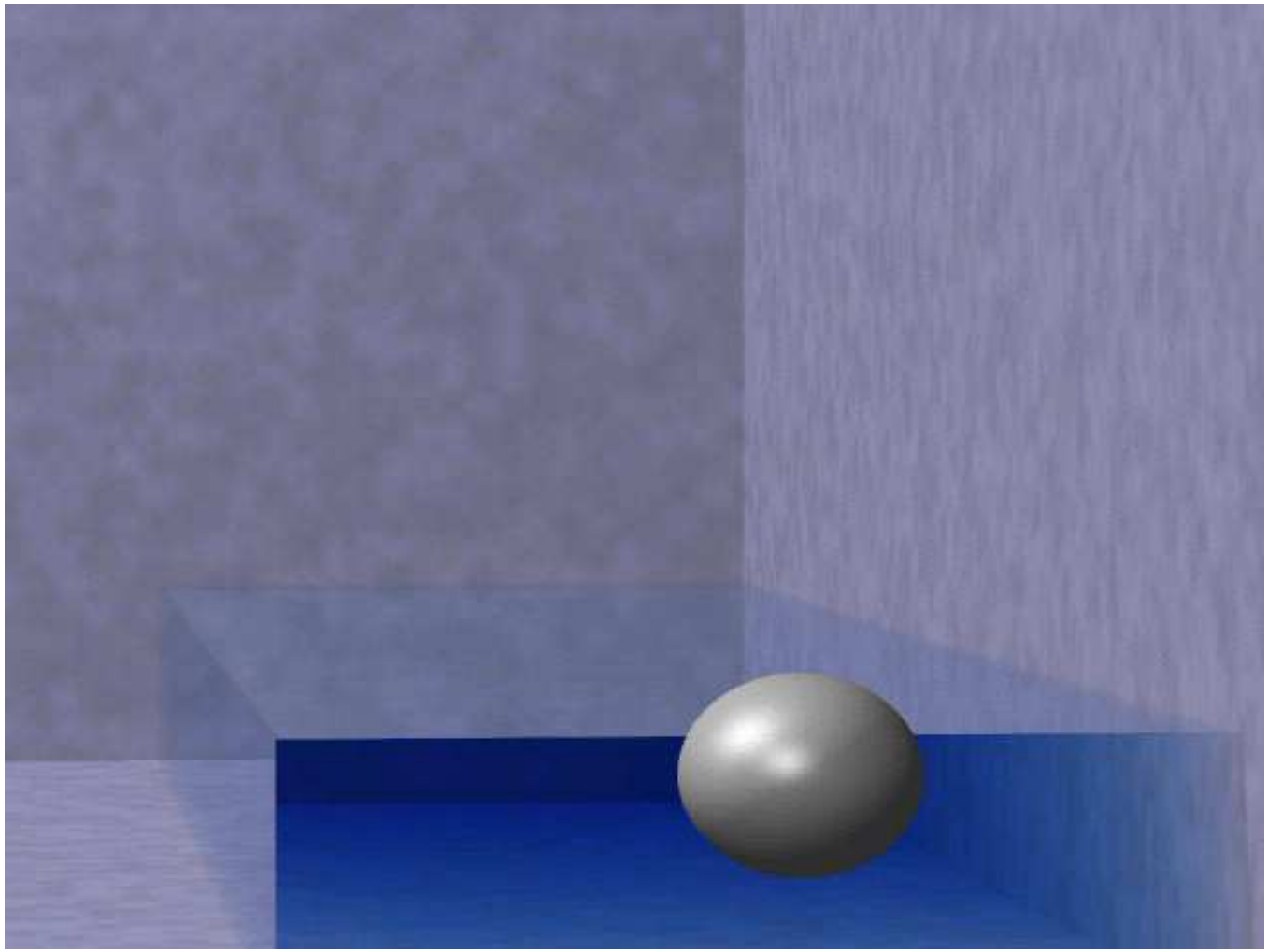
Simulação de partículas



Simulação de partículas



Simulação de partículas



Simulação de partículas

- Vídeos:
  - ⇨ ode-videos\particles_video_3_new.avi
  - ⇨ ode-videos\particles_video_4_new.avi
  - ⇨ ode-videos\flammable.avi
  - ⇨ ode-videos\nonflammable.avi

Futuro da simulação baseada em física

- Implementação em hardware:
  - ⇨ Já estão sendo desenvolvidas placas que realizam a simulação física, como por exemplo a PhysX
  - ⇨ Esta placa é utilizada de forma similar a placa de vídeo, mas é voltada para os cálculos de física
  - ⇨ Os fabricantes disponibilizam uma API. O computador que não possui a placa pode emular os efeitos em software

Futuro da simulação baseada em física

- Coma implementação em hardware, será possível obter um desempenho muito melhor em tempo real
- Vídeos:
  - ⇨ ode-videos\physx_bundle.avi
  - ⇨ ode-videos\divxphysxairtight720x400.avi
  - ⇨ ode-videos\Movie-AGEIA.wmv

Realidade Virtual – Aumentando ainda mais o Realismo – VR++

UNISINOS

UNIVERSIDADE DO VALE DO RIO DOS SINOS

PIPCA

Programa de Pós-Graduação em Computação Aplicada
Grupo de Inteligência Artificial - GIA