2
Revisão Bibliográfica

2.1.

Controle de Tração em Terrenos Acidentados

Trabalhos de pesquisas aplicados a controle de tração de robôs móveis em terrenos acidentados, ou irregulares, foram desenvolvidos por Barral [4] e Auderi [5]. Em [4], o autor apresenta um método de controle de tração para um robô móvel que permite, ao mesmo rover, o deslocamento por vários tipos de terrenos acidentados, de forma a minimizar o consumo de potência em terrenos planos e regulares, e maximizar a estabilidade em terrenos irregulares. Em [5], são desenvolvidas técnicas de controle de capotagem e deslizamento de um robô móvel, visando garantir a locomoção do robô em terrenos irregulares e inclinados. No estudo, diante das condições encontradas, a estabilidade do veículo passou a ser um fator fundamental para o desenvolvimento de um controle eficaz que garantirá segurança nas operações, evitando capotagens e ajudando nas tomadas de decisões, e até recusando a trajetória comandada, se esta lhe oferecer um obstáculo não superável.

Iagnemma [2] apresenta um método que propõe minimizar a razão entre a força de tração e a força normal, medidas com sensores colocados em cada roda, para controlar melhor o deslizamento. Este método tem a vantagem de não precisar conhecer as características do terreno e a velocidade do robô. Simulações realizadas demonstraram bons resultados, entretanto não foram apresentados experimentos.

O método de controle de tração em terrenos acidentados, apresentado por Iagnemma em [1], utiliza como dados de entrada medições das propriedades do terreno e de sua geometria, com o intuito de otimizar o torque nas rodas e obter máxima tração ou mínimo consumo de potência, variando de acordo com o grau

de dificuldade de cada terreno. Os resultados apresentados tanto na simulação como nas situações reais demonstraram a efetividade do método proposto.

2.2.
Simuladores Avaliados

Entre os simuladores de veículos robóticos disponíveis no mercado, a grande parte está direcionada para exploração de terrenos em outros planetas.

- **CLARAty**

CLARAty [10] é um dos softwares de simulação de rovers mais conhecidos na comunidade acadêmica. Na verdade ele deve ser considerado não simplesmente um simulador, mas uma estrutura capaz de reunir várias bibliotecas de veículos e visualizadores 3D para rovers. Chamado oficialmente de framework e não de simulador, é um padrão para Coupled-Layer Architecture for Robotic Autonomy [27, 28, 29 e 30], sendo um esforço colaborativo entre várias instituições, inclusive a NASA Jet Propulsion Laboratory (JPL), NASA Ames Research Center, Carnegie Mellon e a Universidade de Minnesota. O software completo inclui um grande número de módulos para a programação de robôs, mas até o momento a NASA apenas liberou um conjunto de funcionalidades ao público. O CLARAty não é um software de código aberto, pois não está enquadrado às regras da GNU General Public License [13] – ou GNU GPL, ou simplesmente GPL. Essa designação de licença foi idealizada por Richard Stallman em 1984, com o objetivo de criar um sistema operacional totalmente livre. A parte pública das bibliotecas do CLARAty pode ser baixada em [39], inclusive sua licença de uso em [40] (JPL Open Source License). Contudo, a maior parte das bibliotecas ainda continua de uso restrito. Segundo informações no site da NASA JPL, qualquer tipo de uso do software CLARAty, fora das atividades acadêmicas ou amadoras, deve ser encaminhado ao JPL a fim de que possa ser analisado.

Na página do CLARAty em [39] há informações relatando que a parte pública não foi testada completamente por causa de constantes mudanças em alguns de seus módulos; entretanto, a parte do repositório que contém o material

privado vem sendo testada há anos. Isso significa que a versão liberada é classificada como *lite*, ou seja, uma versão gratuita e com várias restrições de uso, defasada em relação à sua versão completa.

São 44 os módulos liberados, equivalentes a 10% de todos os módulos compreendidos no software CLARAty, que abrangem desde recursos de matemática, rotação de matrizes com ângulos de Euler, quatérnios e transformações de coordenadas, inclusive transformações de quatérnios, até a infraestrutura responsável pela interface entre transformações e mecanismos com as partes móveis do rover. Englobam também modelos de rodas, pernas, veículos híbridos, câmeras, motores, suporte para I/O (input/output) analógico e digital, dentre vários outros recursos.

A Figura 3 mostra um dos vários exemplos recuperados do site da CLARAty, que ilustra a simulação do rover Rocky 8, criado com o simulador ROAMS [25] e utilizando o framework CLARAty com o módulo Morphing Navigator para a simulação de navegação. Essa capacidade de utilização de várias bibliotecas para criação de rovers e os mais diversos tipos de controles faz do CLARAty um software bastante modular, ideal para reutilização.

Figura 3 – CLARAty navegando o Rocky 8 ROAMS Simulator [37]

- **Rover Graphical Simulator**

O Rover Graphical Simulator (RGS) [11 e 14] é outro software comercial produzido pela NASA e opera com rovers em terrenos planos com obstáculos. Seu objetivo é desviar deles, em vez de superá-los. Entretanto, esse software é uma simulação gráfica em 2D. Inclui bibliotecas de lógica fuzzy para o desvio de obstáculos.

- **ROAMS**

O ROAMS [25 e 36], ou Rover Analisys, Modeling and Simulation, é mais um software produzido pelo Jet Propulsion Laboratory da NASA. Ele tem o objetivo de modelar e simular futuras missões de pouso e exploração em Marte (Figura 4). Para isso, conta com uma ferramenta de simulação para análise, projeto, desenvolvimento e teste de operação de rovers em superfícies planetárias. Não é um software comercial e está sendo usado atualmente pelo NASA's Mars Program como um teste virtual para vários tipos rovers, terrenos, ambientes e componentes, tais como sensores, câmeras, dentre outros. Um dos recursos mais fortes desse programa é o mapeamento de terrenos.

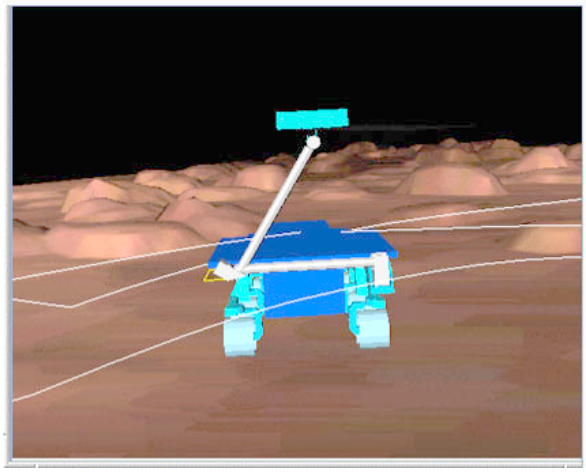

Figura 4 – ROAMS, ou Rover Modeling and Simulation, produzido pelo Jet Propulsion Laboratory da NASA, em uma missão de superfície [25]

- **Universal Mechanisms**

O Universal Mechanisms (UM) [12] é um software comercial de simulação de veículos desenvolvido pelo Laboratório de Mecânica Computacional da Bryansk State Technical University, na Rússia, capaz de lidar com terrenos acidentados. No entanto, ele gera esses terrenos por meio de equações, o que o torna restrito e lento. Embora faça a simulação de veículos robóticos (Figura 5), o foco deste software está na simulação de trens e caminhões de cargas em rodovias, com terrenos planos e pequenos obstáculos como quebra-molas e lombadas (Figura 6).

Figura 5 – Universal Mechanisms simulando um veículo robótico de seis rodas para transporte [12]

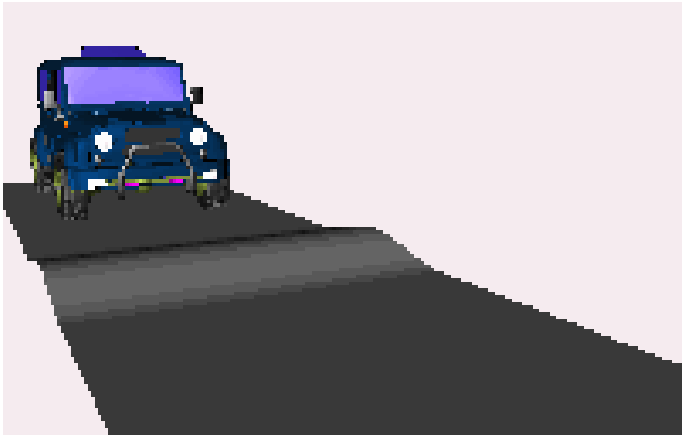

Figura 6 – Universal Mechanisms simulando a dinâmica de um utilitário sobre uma lombada [12]

- **Gazebo**

Gazebo [15 e 31] é um software de simulação em três dimensões, e o seu forte é a simulação de populações de robôs. Para isso, ele conta com recursos como sensores de obstáculos que dão um retorno realístico da interação entre os objetos e inclui uma simulação precisa da física de corpos rígidos (Figura 7).

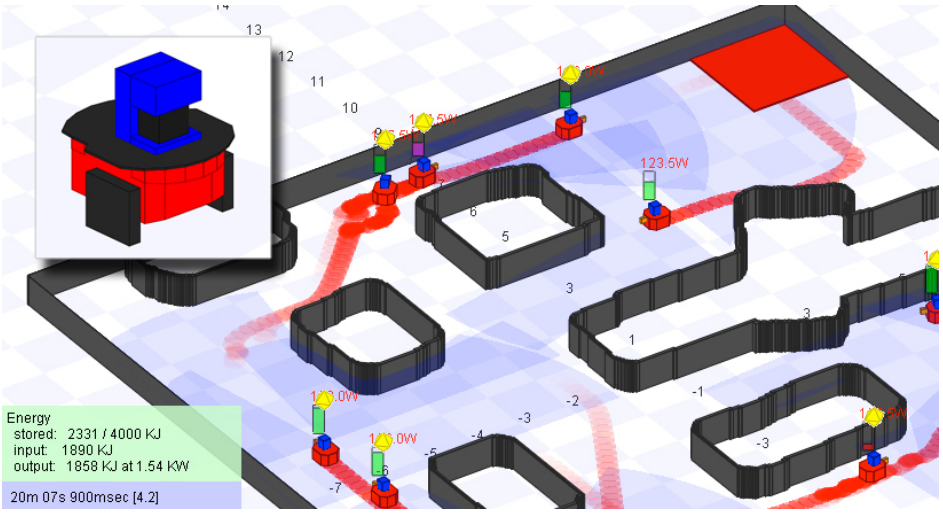

Figura 7 – Gazebo simulando multirrobôs em ambiente controlado, modificado de [15]

Entre os simuladores avaliados, o Gazebo tem um ferramental bem superior aos demais, incluindo sensores virtuais de varrimento de imagem a laser, sonar, GPS, IMU (Inertial Measurement Unit), tipos diferentes de câmeras, adição de objetos com interação entre si e com os robôs, e ainda é um programa freeware. Sua licença está dentro das regras GNU General Public License.

- **RCAST**

O RCAST [16], ou Rover Chassis, Analysis and Simulation Tools, estuda o comportamento e otimização da suspensão de um rover planetário específico. Simula (Figura 8) a dinâmica de multicorpos e a correspondente interação de roda-terreno (Bauer et al., 2005a). Desenvolvido no ambiente Simulink do MATLAB MathWorks, ele utiliza o software comercial AESCO Soft Soil Tire

Model (AS2TM) na modelagem computacional da interação roda-terreno para predizer a sua locomoção, comparado com uma análise quase-estática.

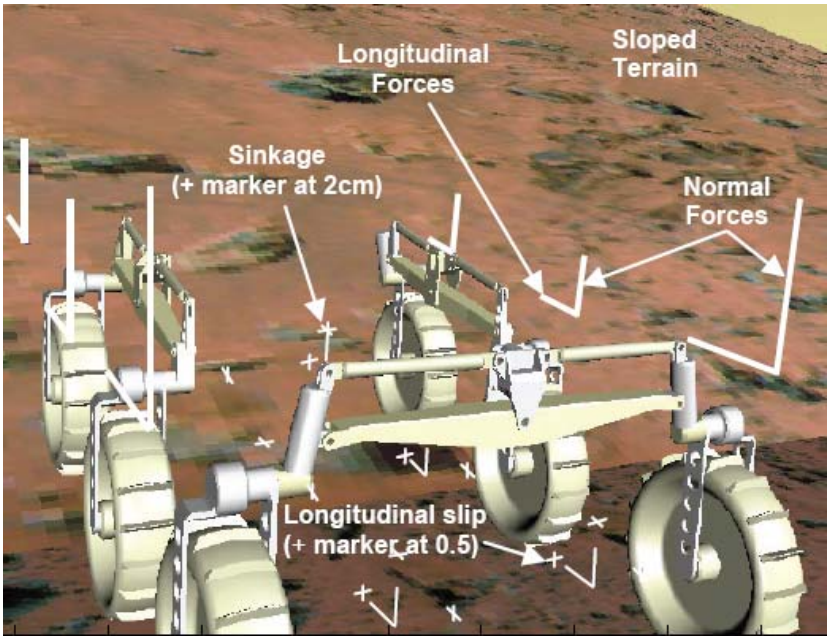

Figura 8 – Um exemplo da configuração de rover no RCAST [16]

- **Working Model 2D**

O Working Model 2D [17], embora seja um simulador dinâmico 2D, é capaz de detectar a colisão de corpos, além de ter uma ferramenta satisfatória [18] na criação e simulação de robôs móveis em terrenos acidentados. O WM2D é um simulador comercial.

- **CRAB Rover**

O CRAB Rover [19] é um robô suíço articulado de seis rodas do Autonomous Systems Lab (ASL). Projetado para exploração em terrenos acidentados, sua tarefa é identificar obstáculos e, se possível, evitá-los (Figura 9). Ele compete diretamente com o MER, Mars Exploration Rover, da NASA [20]. Simulações computacionais foram feitas para validar o modelo CRAB e desenvolver algoritmos de controle eficientes. Toda a simulação foi desenvolvida

em C e C++, e permitiu a criação dentro do simulador dos algoritmos de controle reais, utilizados no rover, sem maiores modificações. A biblioteca ODE [21] também foi utilizada no projeto CRAB Rover para criar o rover e os modelos de terrenos baseados na dinâmica de corpos rígidos.

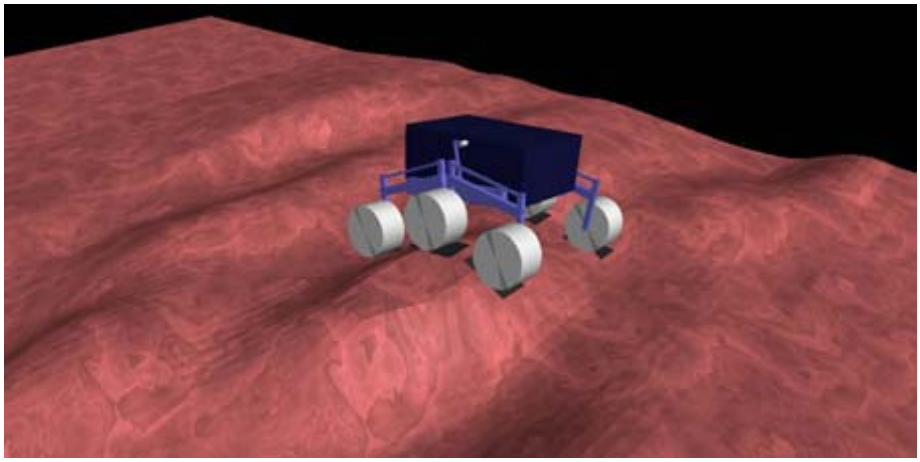

Figura 9 – CRAB Rover durante uma simulação [38]

- **Adams**

O Adams Multbody Dynamics, da MSC Software [22], é um programa bem reconhecido e considerado padrão em muitos setores da indústria, utilizado basicamente como ferramenta de simulação em sistemas de múltiplos corpos. Ele é baseado na fórmula de Newton-Euler e usa Euler-Lagrange para equações do movimento. O software resolve um sistema de equações diferenciais e usa os mais recentes métodos numéricos. Entretanto, para gerar resultados, ele consome tempo e capacidade de processamento demais. Tem a facilidade de modelar rovers de qualquer topologia, mas exige grandes recursos para desenhar um novo modelo e preparar a simulação.

- **Microsoft Robotics Developer Studio 2008**

Microsoft Robotics Developer Studio 2008 (RDS) [33] é um software para o ambiente Microsoft Windows desenvolvido para criação de aplicações robóticas

(Figura 10). Na simulação de aplicações robóticas em ambientes 3D virtuais, se faz necessário o uso da biblioteca PhysX AGEIA Technologies Inc. [34]. O RDS tem também recursos de interação com o robô, monitoração em tempo real dos sensores virtuais robóticos e respostas aos motores e atuadores. É uma plataforma bastante amigável, que permite aos *end-users* (ou usuários finais, não programadores ou especialistas) criar aplicações robóticas por meio de um ambiente de programação visual (Figura 11).

Por ser produto da Microsoft, é o único simulador, dentre os avaliados, a utilizar a biblioteca gráfica DirectX [35].

Figura 10 – Aplicação robótica utilizando o simulador RDS [33]

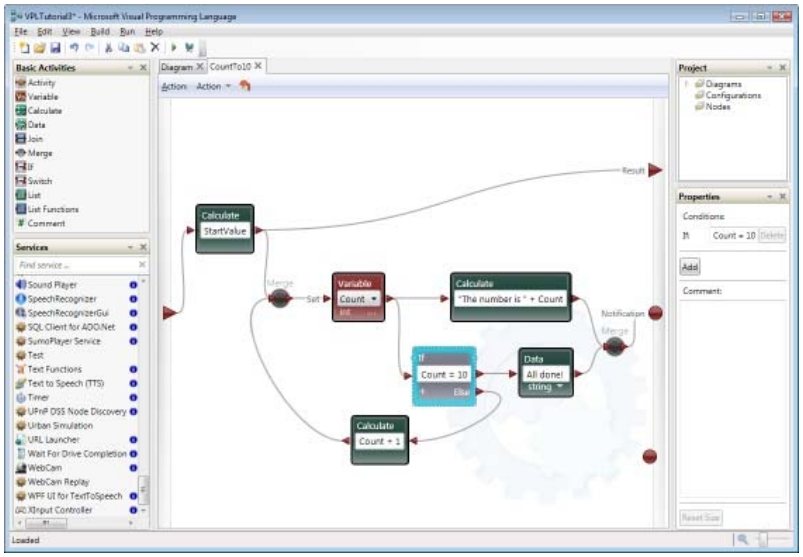

Figura 11 – Ambiente visual do RDS para criação de aplicações robóticas [33]

- **SIMPACK – Multi-Body Simulation Software**

O SIMPACK [47] é uma ferramenta de simulação não-linear de propósito geral para múltiplos corpos em 3D. Ideal para simular sistemas mecânicos, análise de vibrações, cálculo de forças e aceleração, descreve e prediz o movimento de sistemas complexos de múltiplos corpos. Os modelos utilizados podem ser construídos pela ferramenta que o acompanha, ou importados de modelos feitos em CAD e FEM. Utiliza o OpenGL como interface gráfica e foi desenvolvido em Fortran90 (última versão). Não é um software freeware e tem bibliotecas para licenciamento, além de permitir trabalhar com o MATLAB/Simulink.

O SIMPACK está dividido em quatro pacotes: Automotivo, Motores, Veículos sobre Trilhos e Sistemas de Turbinas Eólicas. O pacote mais relacionado a esta dissertação é o Automotivo. Na Figura 12 tem-se um exemplo retirado do vídeo "jump_start_spider_03.avi" do website da SIMPACK, que ilustra o comportamento dinâmico de um veículo passando por uma rampa unilateral.

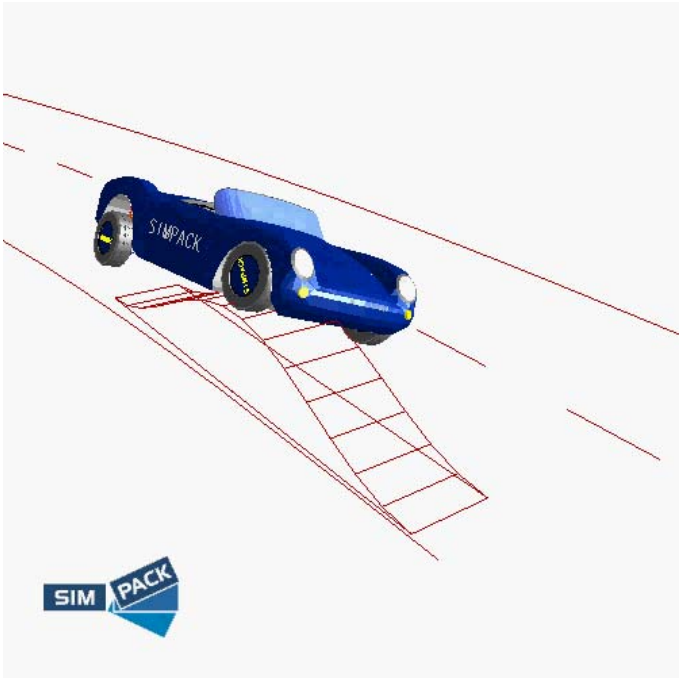

Figura 12 – Veículo passando sobre uma rampa unilateral, SIMPACK Automotivo

A Tabela 1 resume os recursos de cada simulador avaliado, para efeitos comparativos.

Tabela 1 – Comparativo entre os simuladores avaliados

	3D	GNU GPL (freeware)	Terreno Acidentado	Diferentes Rovers	Real Time	Sistema Operacional
ADAMS	✓	✗	✓	✓	não informado	Windows e Linux
CLARAty	✓	✗	✓	✓	✓	Linux
CRAB Rover	✓	✗	✓	CRAB I, II e III	não informado	não informado
Gazebo	✓	✓	✓	✓	✓	Linux
RCAST	✓	✗	✓	dedicado (um único)	não informado	não informado
ROAMS	✓	✗	✓	✓	✓	Linux
Rover Graphical Simulator	✗	✗	✓	não informado	✓	Linux
Universal Mechanisms	✓	✗	✓	✓	não informado	Windows
Working Model 2D	✗	✗	✗	✓	✓	Windows
Microsoft Robotics Developer Studio	✓	✗	✗	Lightweight	✓	Windows
SIMPACK	✓	✗	✓	Veículos	✓	Windows, HP UX, Silicon Graphics

2.3.

Arquitetura dos Simuladores Avaliados

Neste item, tem-se uma breve visão de como os softwares avaliados são implementados, e quais os recursos gráficos utilizados. Infelizmente, poucos são os softwares que expõem a sua arquitetura, até mesmo porque aqueles que são comercializados utilizam a sua discrição como uma vantagem competitiva. Entretanto, a arquitetura mais discutida é a do CLARAty, associada a vários artigos já publicados.

- **CLARAty**

O termo CLARAty é um padrão para Coupled-Layer Architecture for Robotic Autonomy [27-30]. Ele propõe uma arquitetura de duas camadas (Figura 13): uma de decisão e uma funcional. Na camada de decisão estão envolvidas as

tarefas de planejamento, persistência de dados e execução. Na camada funcional estão todas as interfaces do sistema/hardware e suas capacidades. Eis algumas razões para a arquitetura ter sido desenvolvida com orientação ao objeto:

- por ser modular, pode-se implementar facilmente as características de hardware de um sistema robótico;
- em todos os níveis de modularização, funcionalidade e persistência de dados do sistema de informação, os dados podem ser codificados e compartimentalizados em um único local lógico;
- a própria estruturação do software permite o uso das propriedades de herança para administrar a complexidade do desenvolvimento do software; e
- essa estrutura pode ser graficamente desenhada e documentada utilizando o padrão de linguagem gráfica UML (*Unified Modeling Language*).

Figura 13 – Arquitetura em camadas do CLARAty, modificado de [10]

O diagrama da Figura 14 mostra uma breve ilustração da hierarquia de classes encontrada na camada funcional do CLARAty. É uma abstração para demonstrar a estrutura de classes na camada Funcional. Como subclasse tem o objeto rover, que agrega o braço (manipulador) robótico e a estrutura de locomoção. Enquanto esses objetos têm um compromisso específico com a classe "Meu Rover", cada uma dessas classes é derivada de uma classe pai, que tende a ser muito mais genérica, subindo até as superclasses.

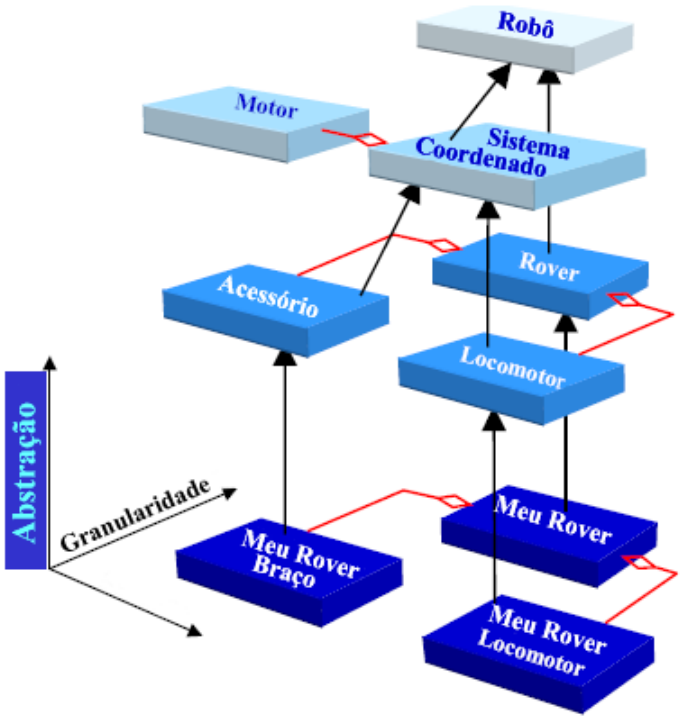

Figura 14 – Exemplo, modificado de [27], ilustrando a hierarquia de objetos e o conceito de herança de classes

- **Gazebo**

A arquitetura do aplicativo Gazebo (Figura 15) é bem clara e modularizada. Essencialmente, existem três blocos principais:

  - modelo – envolve o mundo, forma dos objetos em cena, forma dos robôs, sensores e dispositivos que representam os objetos no mundo real

- interface – praticamente todos os recursos gráficos da simulação utilizando a biblioteca GLUT [59] do OpenGL
- dinâmica – cálculos dinâmicos utilizando a biblioteca ODE [21]

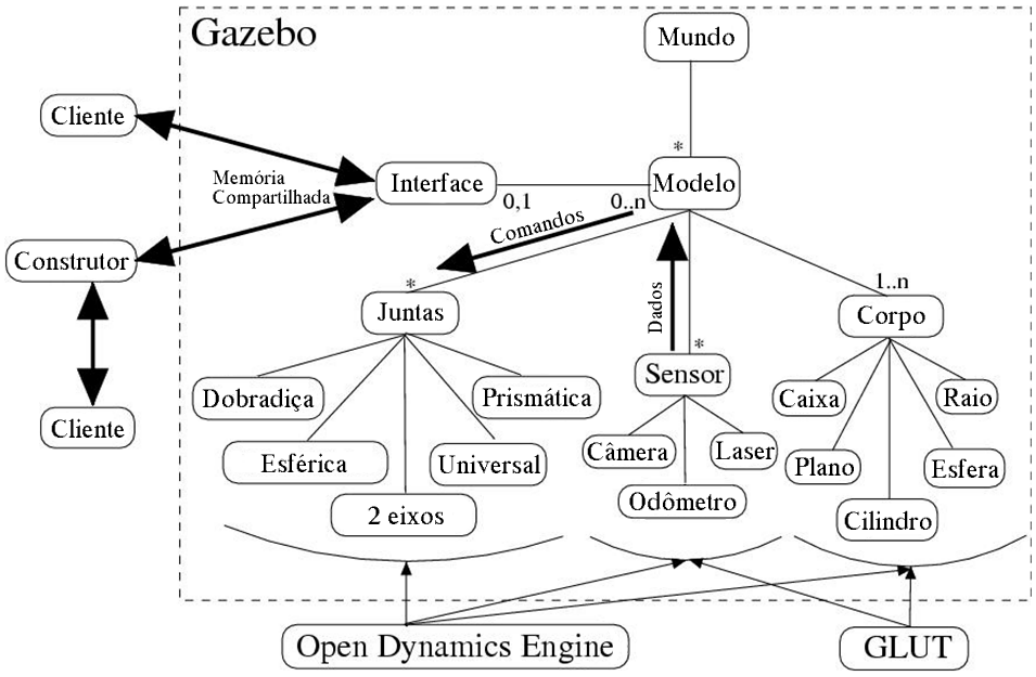

Figura 15 – Diagrama da arquitetura do aplicativo Gazebo, modificado de [60]

2.4.

Biblioteca Gráfica dos Simuladores Avaliados

As principais bibliotecas gráficas em evidência para a construção de objetos em 3D são a OpenGL e Direct3D.

A Direct3D é parte do pacote DirectX desenvolvido pela Microsoft® Corporation. É uma API que provê funcionalidades para criar objetos gráficos em três dimensões, podendo usar os aceleradores de hardware das placas gráficas, se houver. Seu uso é restrito à plataforma Microsoft Windows.

A OpenGL é outra API que provê funcionalidades para criar objetos gráficos em três e duas dimensões, também há a possibilidade de usar os aceleradores da placa gráfica. Está disponível em quase todos os sistemas operacionais atuais, não se limitando apenas ao Microsoft Windows, Mac OS X e Linux.

Abaixo, uma tabela informa a biblioteca gráfica de cada simulador avaliado.

Tabela 2 – Biblioteca gráfica dos sistemas avaliados

	Biblioteca Gráfica	
	OpenGL	**DirectX**
ADAMS	✓	✗
CLARAty	✓	✗
CRAB Rover	✓	✗
Gazebo	✓	✗
RCAST	✓	✗
ROAMS	✓	✗
Rover Graphical Simulator	✓	✗
Universal Mechanisms	✓	✗
Working Model 2D	não informado	não informado
Microsoft Robotics Developer Studio	✗	✓
SIMPACK	✓	✗

2.5. Biblioteca Utilizada na Dinâmica dos Simuladores Avaliados

O desenvolvimento de simuladores dinâmicos requer a utilização de bibliotecas para solucionar as equações da dinâmica de corpos rígidos, que podem ser públicas, comerciais ou proprietárias (Tabela 3). Entretanto, nem todas as bibliotecas existentes no mercado são de fácil compreensão. Além disso, elas têm mais recursos que o necessário para muitos dos projetos. Para contornar essas situações, algumas das bibliotecas tendem a ser específicas e resolver apenas parte do problema.

Para evitar que partes do código da biblioteca sejam adicionadas sem necessidade, boa parte delas tende a ser modularizada, permitindo a sua link-edição (ou ligação, que consiste em unir ao programa principal, no caso o simulador, apenas as referências resolvidas, ou seja, as chamadas de funções que são referenciadas) com a parte necessária ao simulador ou jogo.

A indústria de jogos tem se beneficiado muito dessas bibliotecas, já que elas permitem a otimização, resultando em menor tempo de execução e mais estabilidade face à precisão, fazendo com que elas sejam vocacionadas para a simulação em tempo real (requerida pelos jogos).

Tabela 3 – Simuladores avaliados e suas bibliotecas de dinâmica

	Biblioteca de Dinâmica
ADAMS	solução proprietária
CLARAty	solução proprietária
CRAB Rover	ODE
Gazebo	ODE
RCAST	AS2TM
ROAMS	Darts/Dshell (solução proprietária)
Rover Graphical Simulator	solução proprietária
Universal Mechanisms	solução proprietária
Working Model 2D	solução proprietária
Microsoft Robotics Developer Studio	PhysX AGEIA
SIMPACK	solução proprietária/ tem também opção para importar equações (apenas para versão licenciada)

Uma atenção especial é dada para o fato de a biblioteca PhysX estar inclusa no hardware de placas de vídeo como ATI e NVidia. Logo, ao utilizar esses recursos, obtém-se um ganho computacional superior ao de qualquer outra biblioteca que não esteja implementada em hardware. Atualmente, a PhysX AGEIA [34] é parte da empresa NVidia. A PhysX é muito eficiente para gerar simulações realísticas, mas não necessariamente realistas. Ou seja, o usuário (em geral de jogos envolvendo efeitos dinâmicos) tem a sensação de realismo ao visualizar as saídas do PhysX, mas os cálculos são aproximados e normalmente não obedecem às condições de contorno em articulações. Isso acontece porque o PhysX modela qualquer sistema como um sistema multicorpos de parâmetros concentrados, conectados por molas e amortecedores. Assim, para definir, por exemplo, uma junta prismática (telescópica) é preciso unir as partes móveis por molas muito rígidas nas direções perpendiculares ao movimento da junta.

A ODE [21] é uma biblioteca voltada para estruturas articuladas, ideal para veículos com pernas. Em sua documentação, [41] enfatiza-se que ela é direcionada para simulações em tempo real, com prioridade para velocidade e estabilidade face à precisão. Dividida em dois níveis de simulação, dinâmica e colisões, tem o objetivo de permitir a flexibilidade e maior utilidade da biblioteca, maximizando com isso seu uso e composição. A parte da dinâmica tem por objetivo cuidar das propriedades dinâmicas dos corpos, tais como massas, velocidades e momentos de inércia, sem levar em consideração as formas dos corpos, enquanto a parte da colisão cuida da forma dos corpos.

A ODE é uma das bibliotecas públicas de junções (articulações) mais ricas, se não for a mais rica. Permite oito tipos diferentes de junções de até 3 graus de liberdade cada. Além disso, permite também a composição de corpos e junções em série, tudo bem documentado e com ilustrações [41].

A biblioteca dinâmica DARTS é uma solução proprietária do laboratório de mesmo nome, da NASA, The DARTS Simulation Laboratory [57]. DARTS também é o acrônimo de *Dynamics And Real-Time Simulation*. Essa biblioteca é vencedora do *NASA Software Award* de 1997, tida como uma biblioteca computacional do mais alto desempenho para a dinâmica de multicorpos flexíveis.

Dshell [57] é um framework (abstração estrutural capaz de reunir diversas ferramentas e funcionalidades para geração de aplicativos) responsável pela criação de simulação em tempo real.

A biblioteca AS2TM [58], AESCO Soft Soil Tire Model, está disponível apenas sob licença de uso comercial e tem um modelo computacional de interação roda-terreno.

No próximo capítulo, o simulador desenvolvido é descrito.