

34º JAI - Jornadas de Atualização em Informática



CSBC2015

De 20 a 23 de julho de 2015.

**XXXV CONGRESSO DA SOCIEDADE
BRASILEIRA DE COMPUTAÇÃO**

a internet de tudo, toda observada
RECIFE | PERNAMBUCO | BRASIL

Simulação de Robôs Móveis e Articulados: Aplicações e Prática

Fernando Santos Osório
Rafael Alceste Berri

Simulação de Robôs Móveis e Articulados

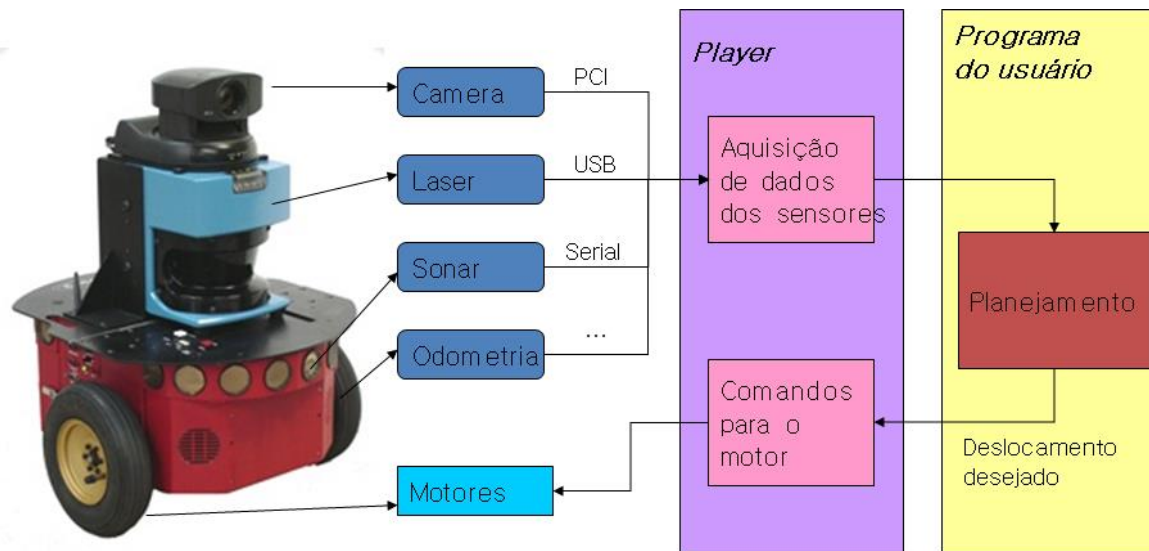
Fernando Santos Osório

Rafael Alceste Berri

Ambientes de Software: Aplicações Robóticas



- Middleware: Acesso aos Dispositivos de Hardware
Atuadores e Sensores => Ex. PLAYER/STAGE [2000-2010]



https://en.wikipedia.org/wiki/Player_Project

Player Software

2010



[Home](#)
[Wiki](#)

[Software](#)
[Player](#)
[Stage](#)
[Gazebo](#)
[Extensions](#)

[Docs](#)
[Manuals](#)
[FAQ](#)
[Publications](#)

[People](#)
[Authors](#)
[Users](#)
[Funding](#)

sourceforge

[Download](#)
[Project](#)
[Bugs](#)
[Help](#)

github
SOCIAL CODING

[Downloads](#)
[Project](#)
[Bugs](#)



Cross-platform robot device interface & server

About Player

Player is a network server for robot control. Running on your robot, Player provides a clean and simple interface to the robot's sensors and actuators over the IP network. Your client program talks to Player over a TCP socket, reading data from sensors, writing commands to actuators, and configuring devices on the fly.

Player supports a variety of robot hardware. The original Player platform is the ActivMedia Pioneer 2 family, but several other robots and many common sensors are supported. Player's modular architecture makes it easy to add support for new hardware, and an active user/developer community contributes new drivers.

Player runs on Linux (PC and embedded), Solaris and *BSD.

Features of Player

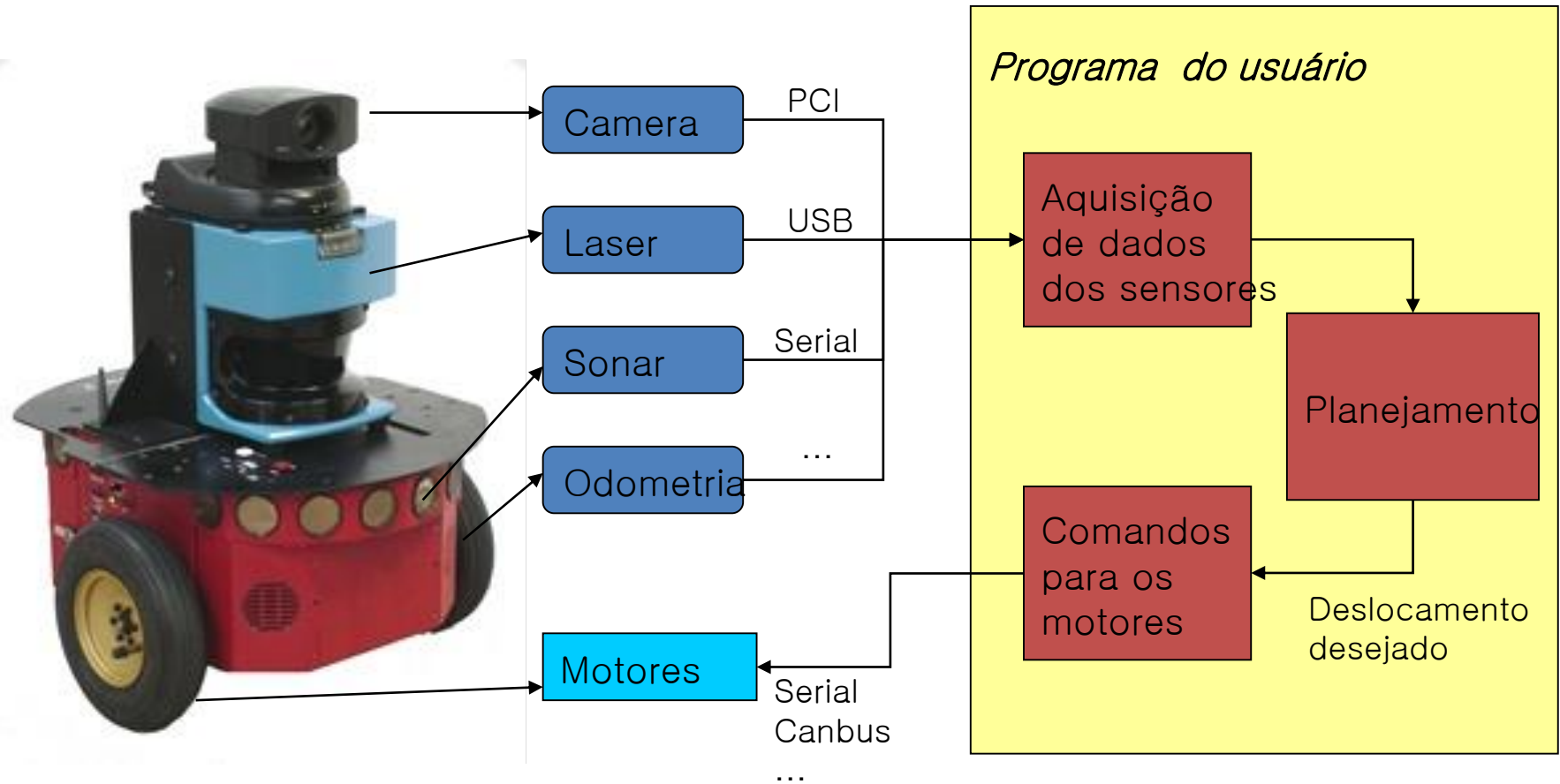
Player is designed to be language and platform independent. Your client program can run on any machine that has a network connection to your robot, and it can be written in any language that supports TCP sockets. We currently have **client-side utilities available** in C++, Tcl, Java, and Python. Further, Player makes no assumptions about how you might want to structure your robot control programs. In this way, it is much more "minimal" than other robot interfaces. If you want your client to be a highly concurrent multi-threaded program, write it like that. If you like a simple read-think-act loop, do that. If you like to control your robot interactively, try our Tcl client (or write your own client utilities in your favorite interactive language).

Player allows multiple devices to present the same interface. For example the Pioneer 2 and RWI drivers both use Player's 'position' interface to allow control of the robot's movement. Thus the same control code could drive both kinds of robot. This feature is very useful when combined with the **Stage** simulator; control programs written for Stage's simulated robots will often work unchanged on real hardware.

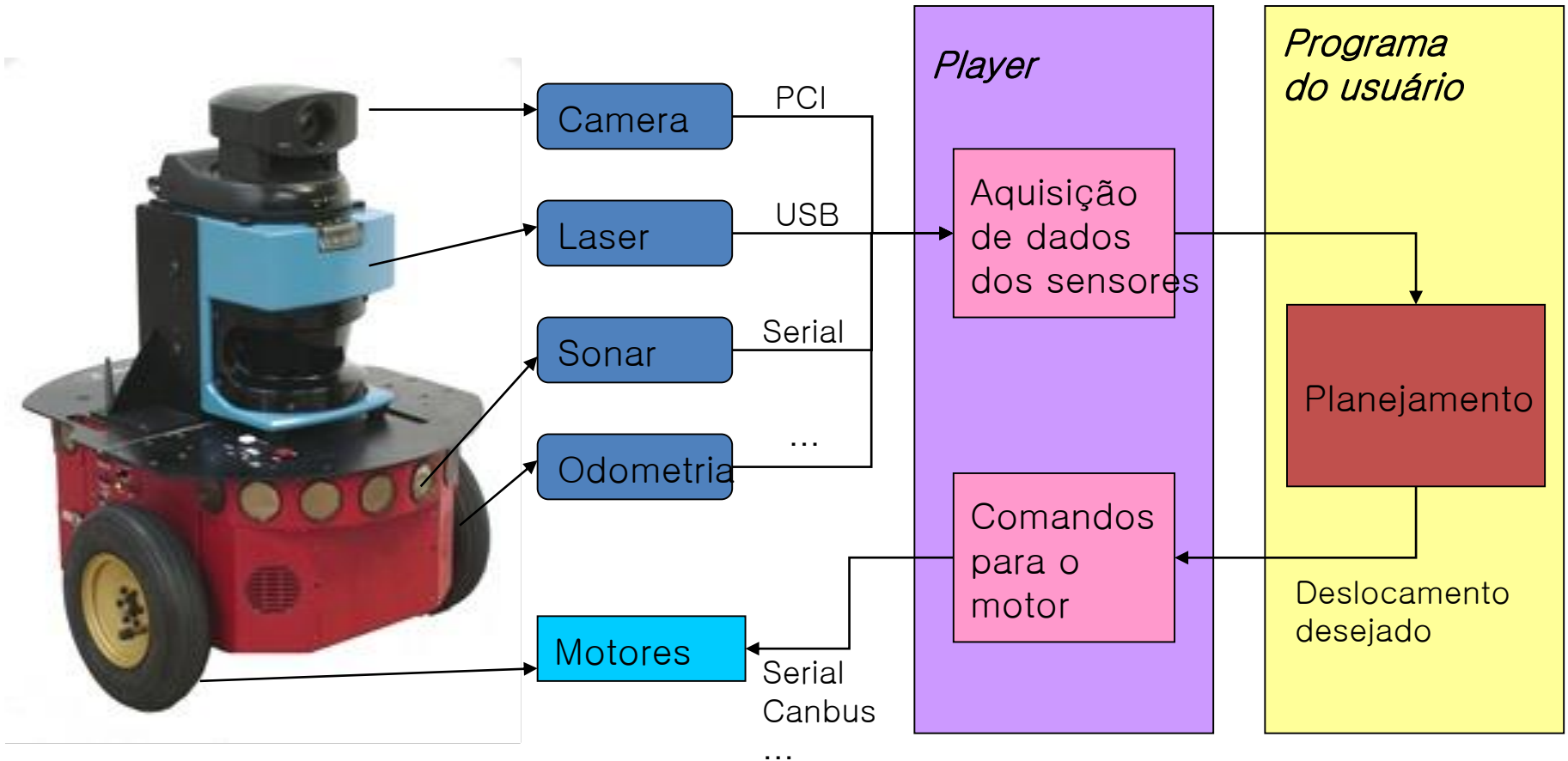
Player is also designed to support virtually any number of clients. Have you ever wanted your robots to "see" through each others' eyes? Now they can. Any client can connect to and read sensor data from (and even write motor commands to) any instance of Player on any robot. Aside from distributed sensing for control, you can also use Player for monitoring of experiments. For example, while your C++ client controls a robot, you can run a graphical visualization tool elsewhere that shows you current sensor data and a logger program to save data for later analysis. On-the-fly device requests allow your clients to gain access to different sensors and actuators as needed for the task at hand.



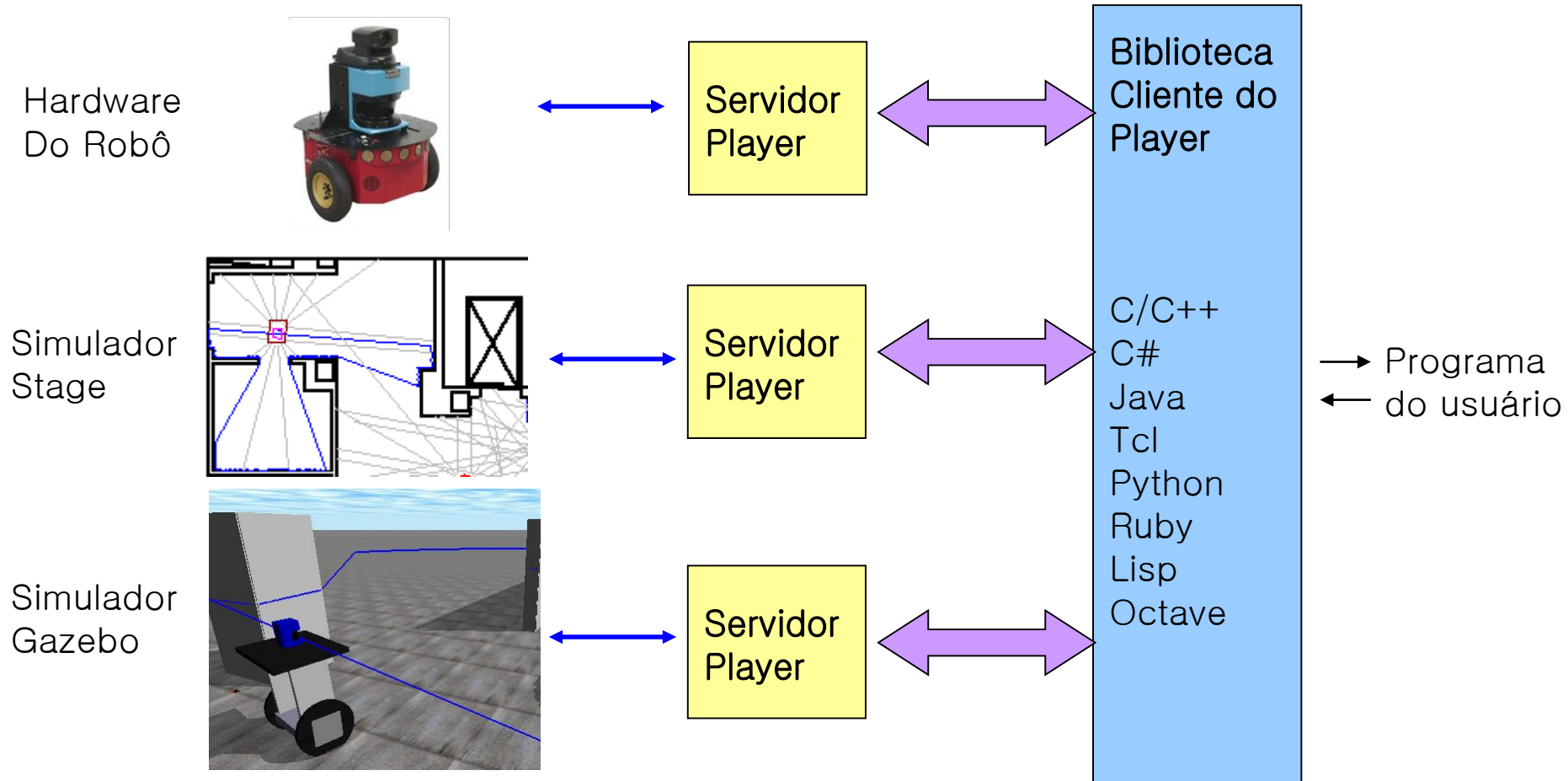
Programa de controle



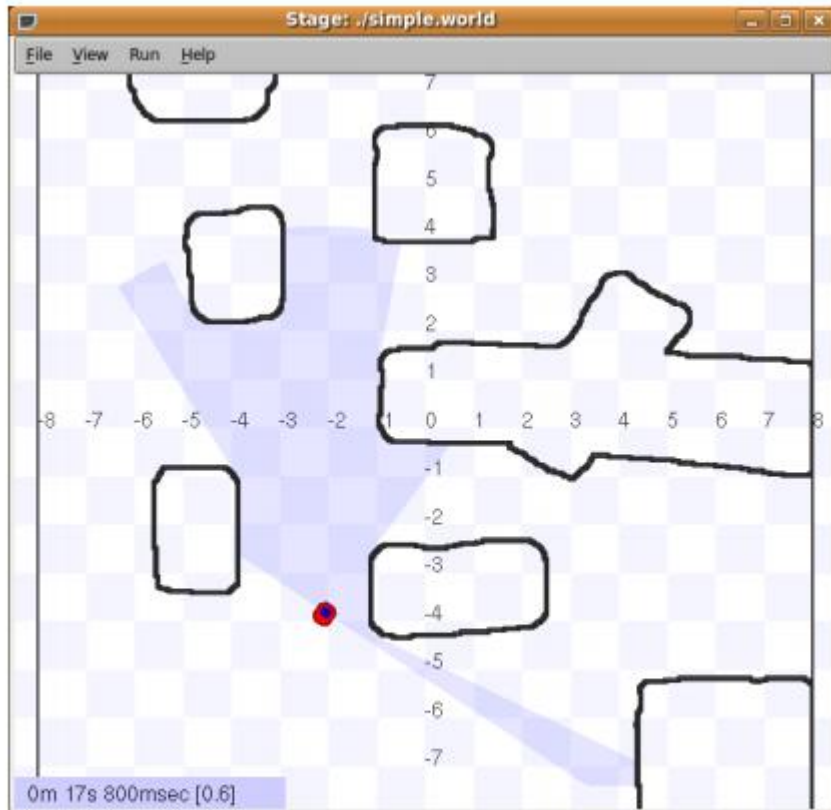
Player



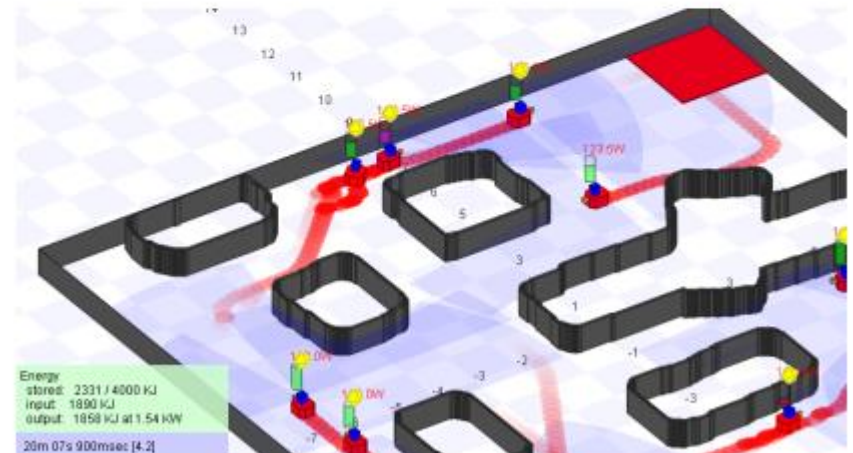
Abstração de hardware



Player + Stage (simulação)



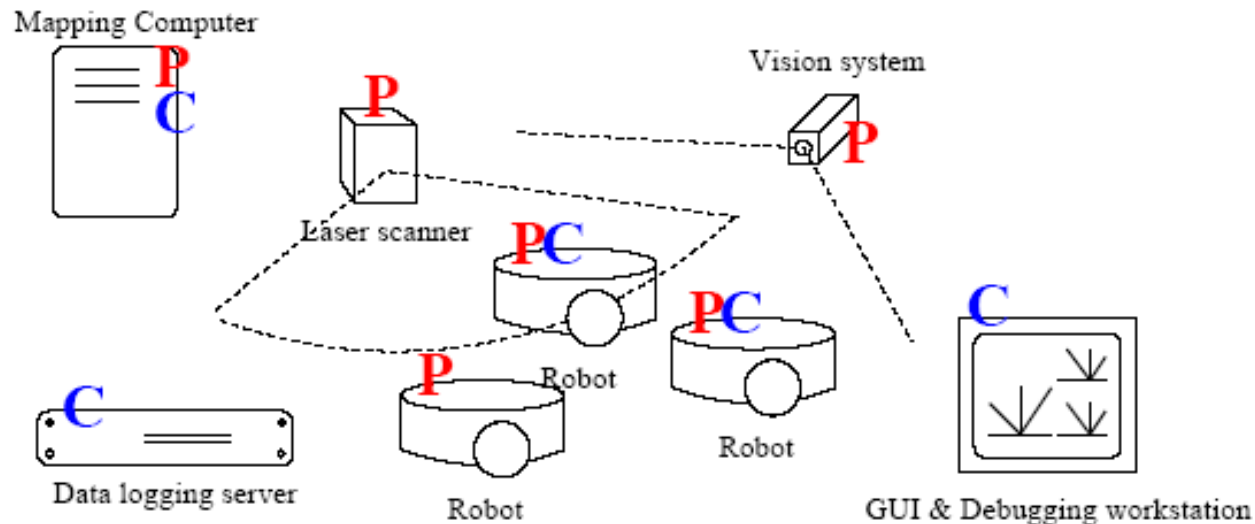
(a) 2D



(b) 2,5D

STAGE

Modelo Cliente/Servidor



- Clientes podem se conectar a múltiplos servidores
- Servidores aceitam conexão de múltiplos clientes
- Diferentes programas/processos/threads podem processar dados de diferentes sensores do mesmo servidor.
- Operação remota

Ambientes de Software: Aplicações Robóticas



- Middleware: Acesso aos Dispositivos de Hardware
Atuadores e Sensores => Ex. PLAYER/STAGE
- Sistema Operacional Robótico => ROS
ROS = Robot Operating System [2010-2016-...]

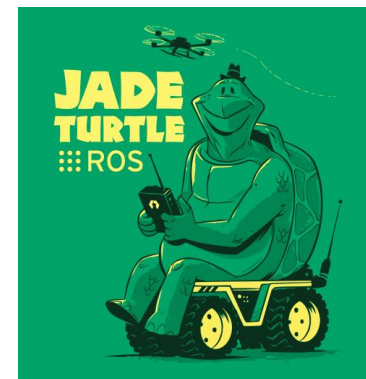
ROS.org

<http://ros.org/>

<http://wiki.ros.org/Distributions>



Open Source Robotics Foundation



Software



[ABOUT US](#) | [DOWNLOAD](#) | [JOBS](#) | [CONTACT](#) | [SUPPORT](#)

[ROBOTS](#)

[SOFTWARE](#)

[RESEARCH](#)

[BLOG](#)

SEARCH

Software

Overview

Overview →

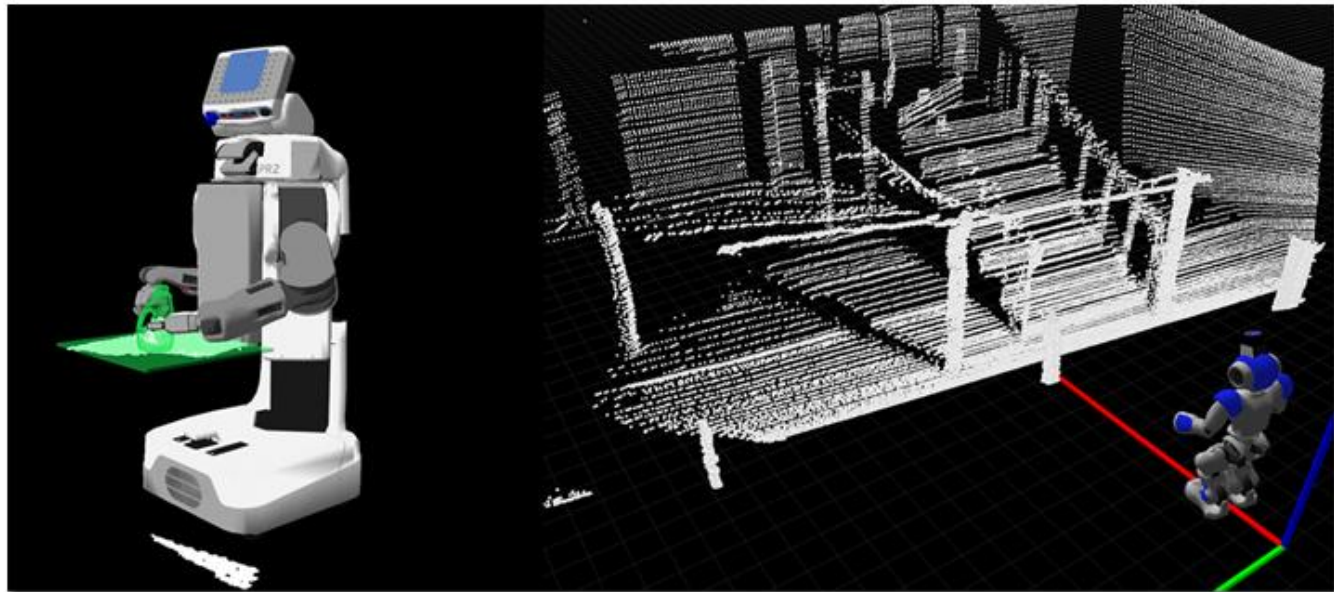
[ROS](#)

[OpenCV](#)

[PCL](#)

[Gazebo](#)

[Player](#)



Our mission is to help advance the state-of-the-art in autonomous robotics technologies. We believe that great advances in robotics will come from the open source robotics community. That is why we work hard to support and contribute to the open source robotics community.

We're working on software from the lowest level with **ROS (Robot Operating System)** to the highest and in-between with our

What is ROS?

ROS

[About](#)[Why ROS?](#)[Getting Started](#)[Get Involved](#)[Blog](#)

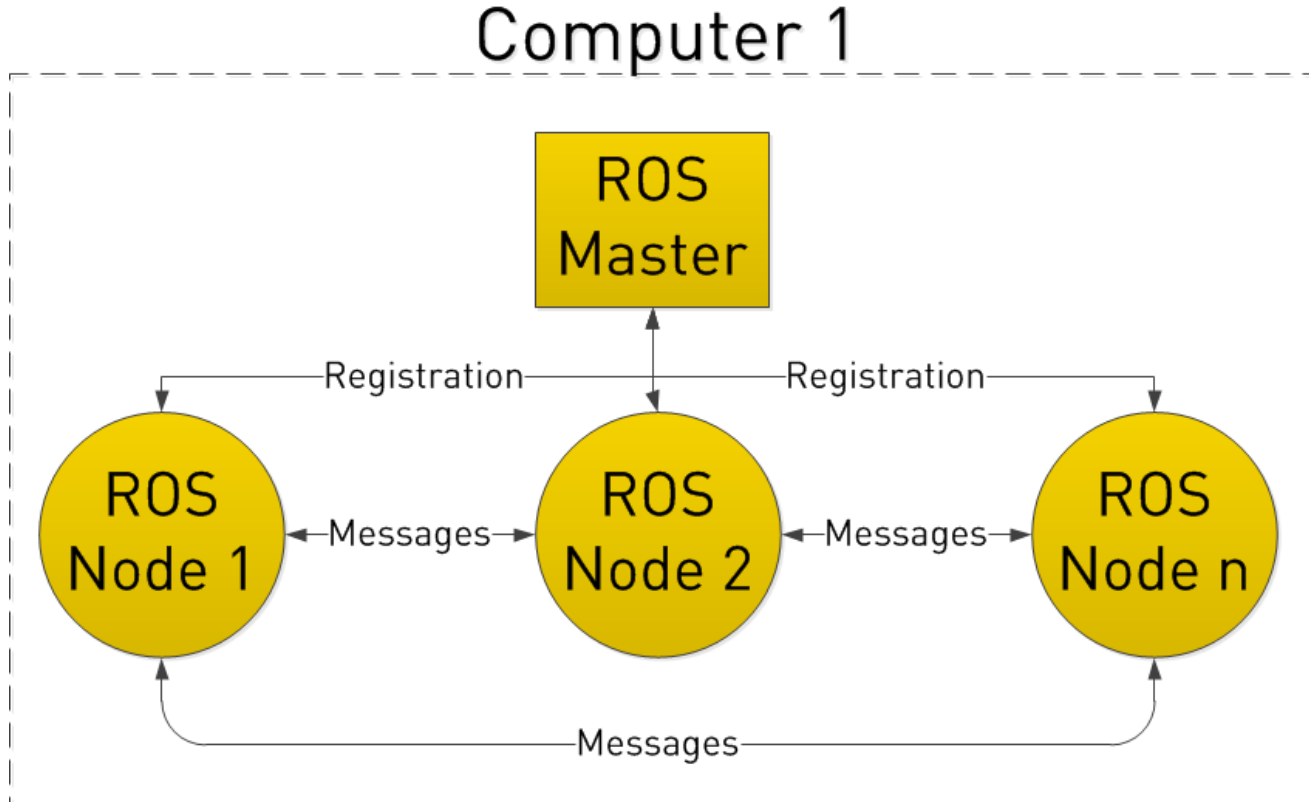
What is ROS?

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.

[Read More](#)

What is ROS?

- Exemplos...



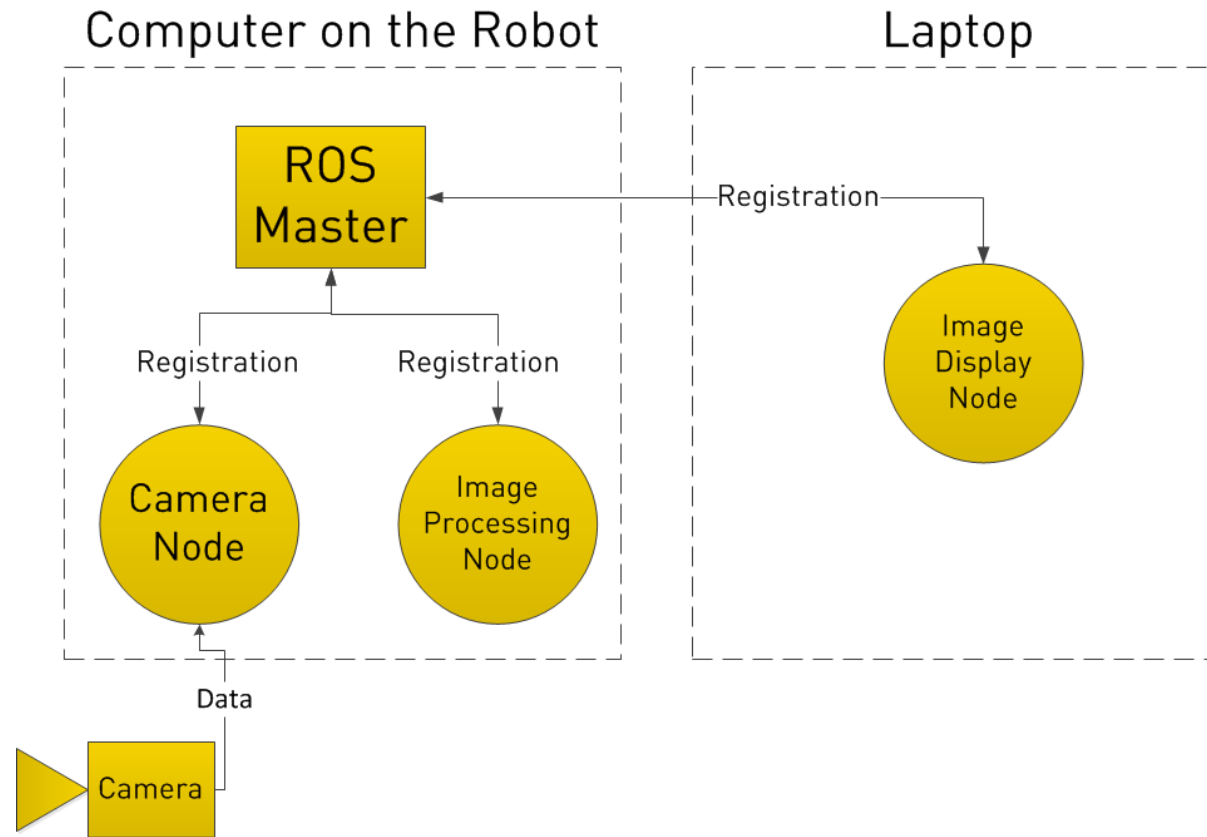
Services as
Distributed Nodes:

- Publish
- Subscribe
- Messages

- Nodes &
Packages

What is ROS?

- Exemplos...



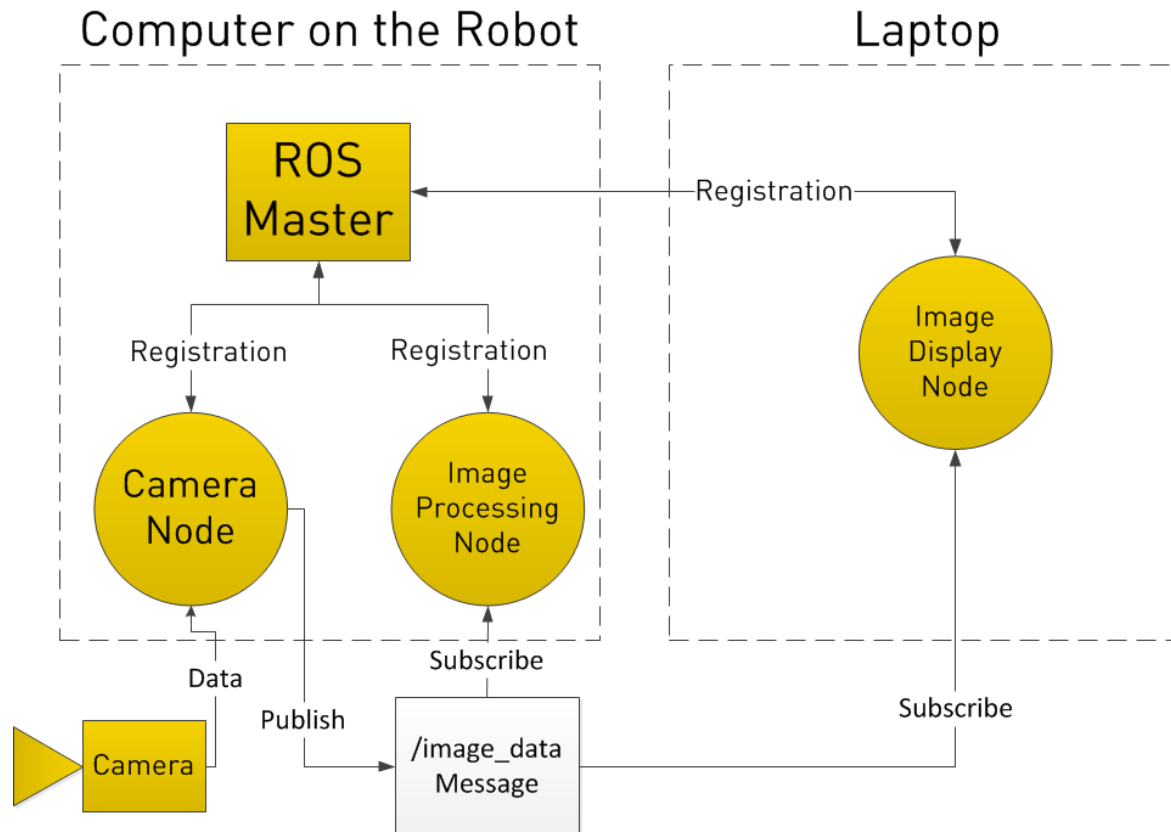
Services as Distributed Nodes:

- Publish
- Subscribe
- Messages

- Nodes & Packages

What is ROS?

- Exemplos...



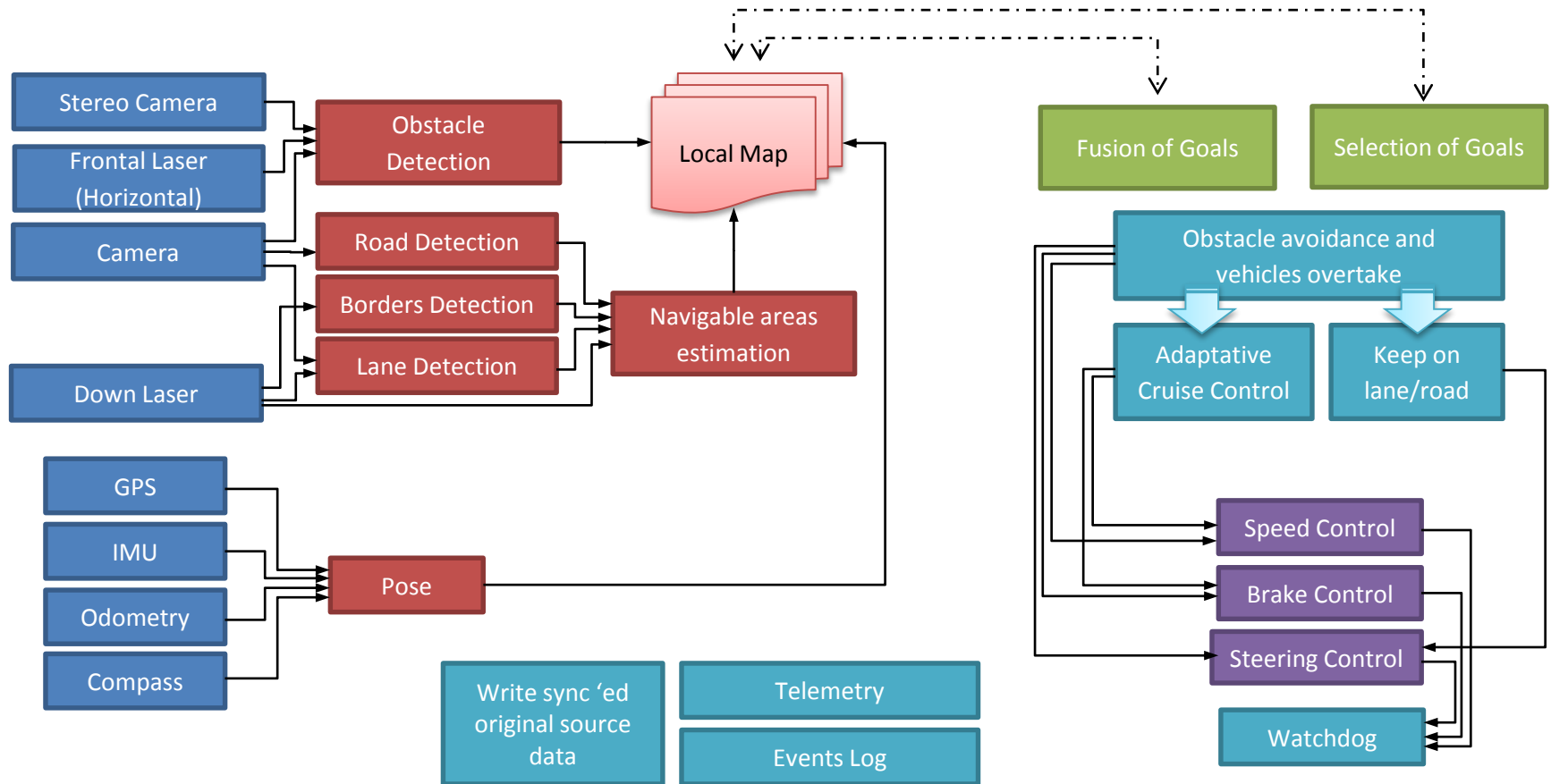
Services as
Distributed Nodes:

- Publish
- Subscribe
- Messages

- Nodes &
Packages

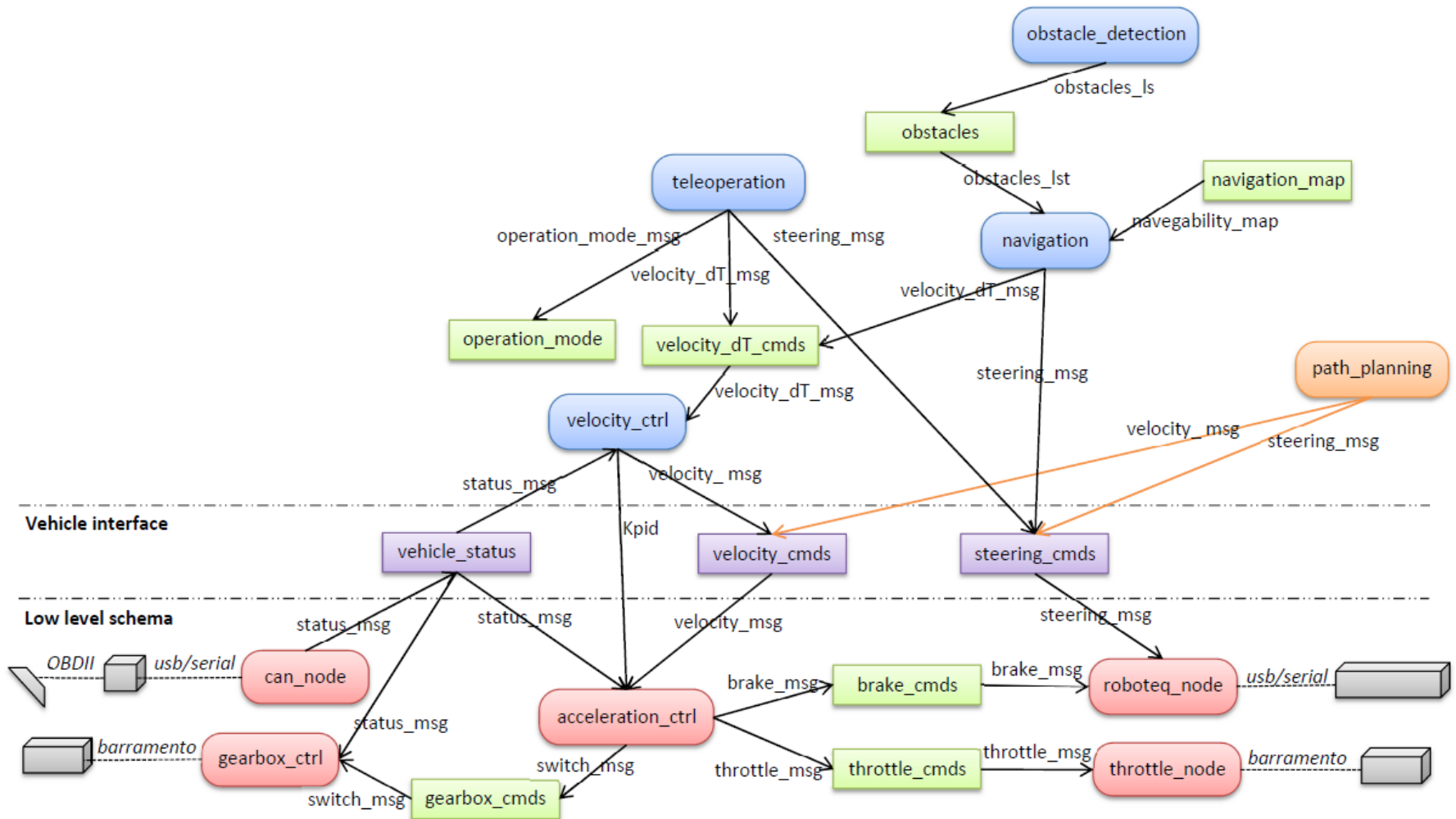
Projeto CaRINA I : R&D

Arquitetura de Software

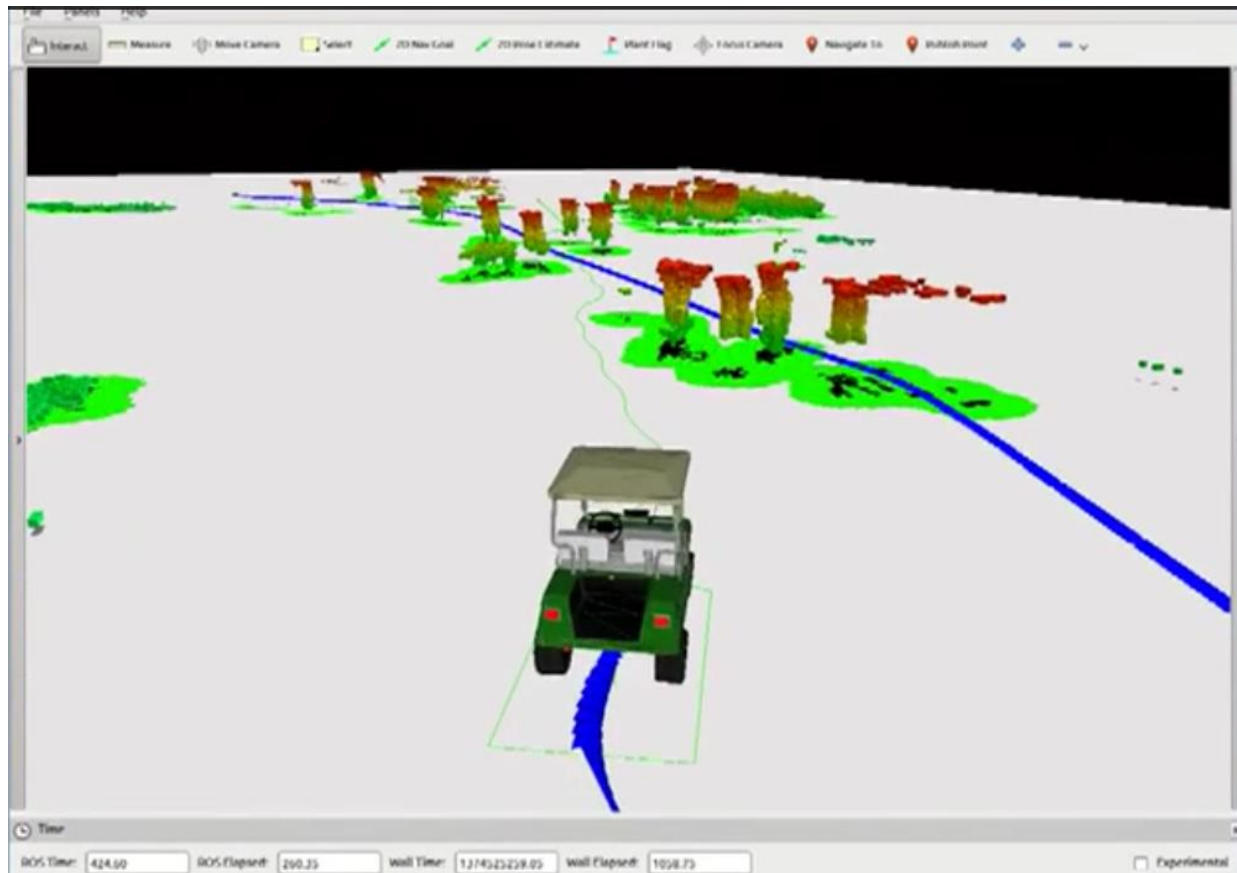


Projeto CaRINA II : R&D

Arquitetura de Software



ROS + GAZEBO Simulation



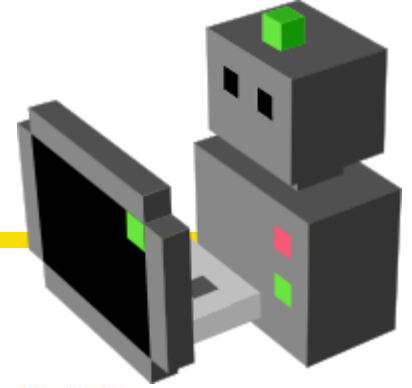
GAZEBO

<http://gazebosim.org/>

<https://www.youtube.com/user/rklaser/videos>



Software de Simulação MORSE



<https://www.openrobots.org/wiki/morse/>

MORSE »

[previous](#) | [next](#) | [modules](#) | [index](#)

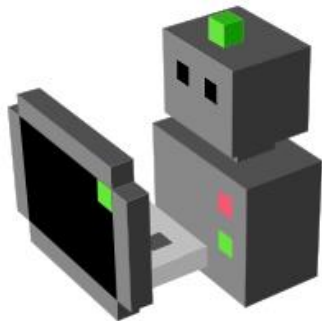


Table Of Contents

What is MORSE?

- Simulation with MORSE
 - Extending MORSE
- Integration in your workflow
- Performances
- MORSE installation
- MORSE as a software project
 - Community
 - Documentation
- Focus on academic requirements
- What else?
- MORSE limitations

[Previous topic](#)

[Advanced Blender commands](#)

What is MORSE?



MORSE is an **generic simulator for academic robotics**. It focuses on realistic 3D simulation of **small to large environments**, indoor or outdoor, with one to tenths of autonomous robots.

MORSE can be entirely controlled from the **command-line**. Simulation scenes are **generated from simple Python scripts**.

MORSE comes with a set of *standard sensors* (cameras, laser scanner, GPS, odometry,...), *actuators* (speed controllers, high-level waypoints controllers, generic joint controllers) and *robotic bases* (quadrotors, ATRV, Pioneer3DX, generic 4 wheel vehicle, PR2,...). New ones can easily be added.

MORSE rendering is based on the *Blender Game Engine*. The OpenGL-based Game Engine supports shaders, provides advanced lightning options, supports multi-texturing, and use the state-of-the-art *Bullet* library for physics simulation.

Simulation with MORSE

Robotic Tools



TOOLS:

Player-Stage (Old)



Gazebo – GazeboSim Robot Simulator

ROS – Robot Operating System



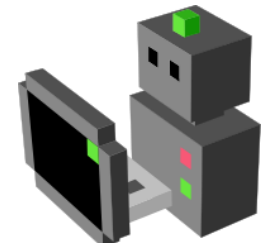
OpenCV – Open Source Computer Vision Library



PCL – Point Cloud Library



Morse - Modular OpenRobots Simulation Engine



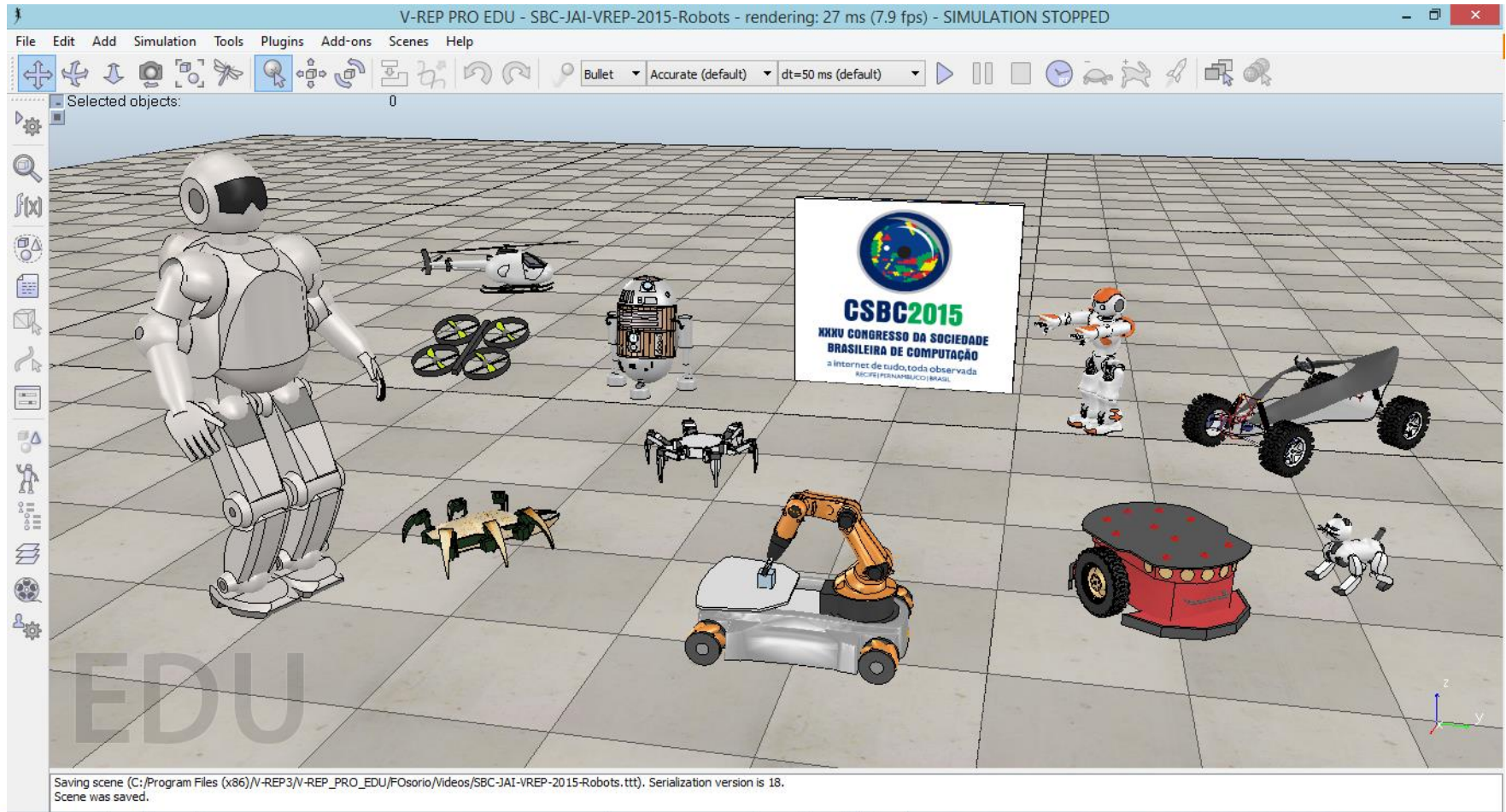
Microsoft Robotics Studio, Webots, Aria, ...

Matlab, Weka, SNNS/JavaNNS, FANN, GALib, Python, ...

Morse



Simulação com V-REP



Free Educational, Easy-of-Use, Multi-Platform
Large set of Sensors, Actuators and Robots

VREP



Simulador V-REP

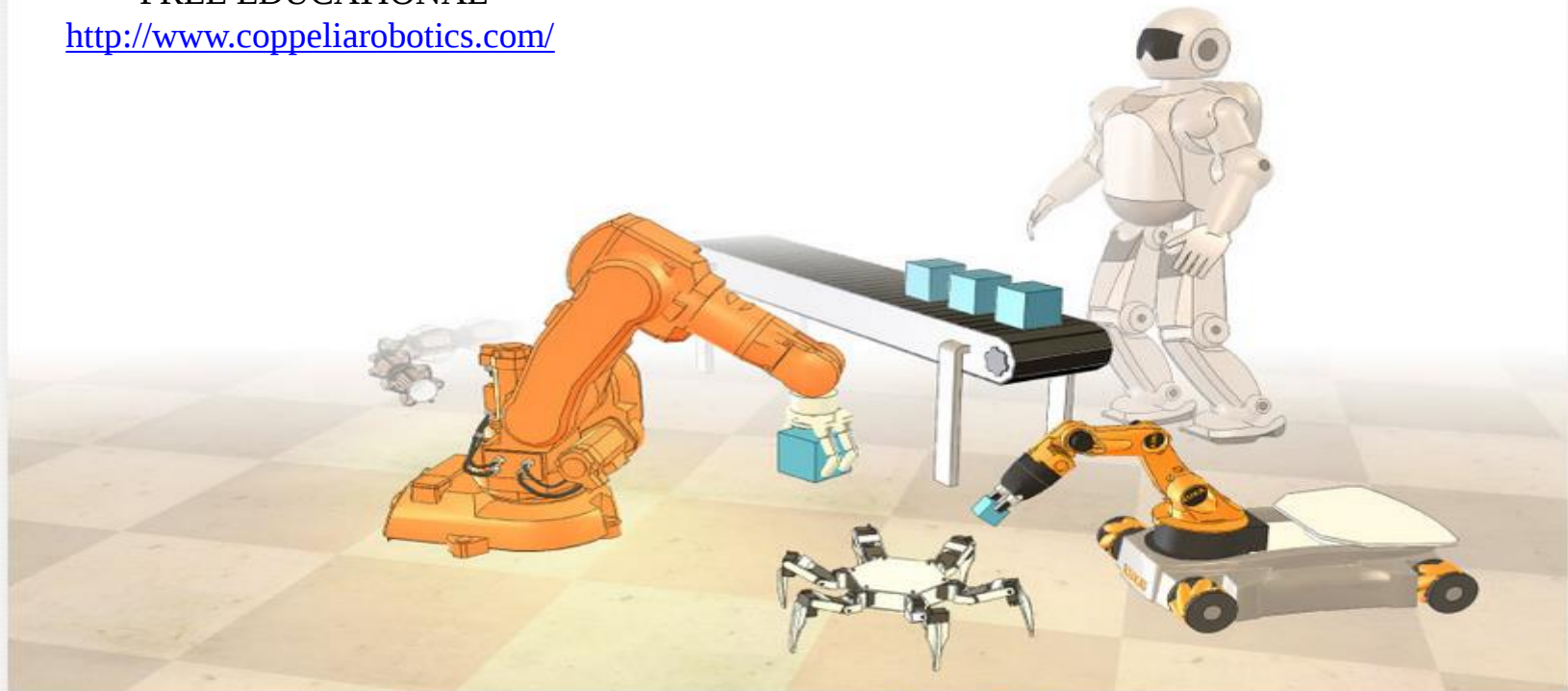


[home](#) [features](#) [videos](#) [downloads](#) [licensing](#) [resources](#) [partners](#) [contact](#)

Cross-Platform & Portable

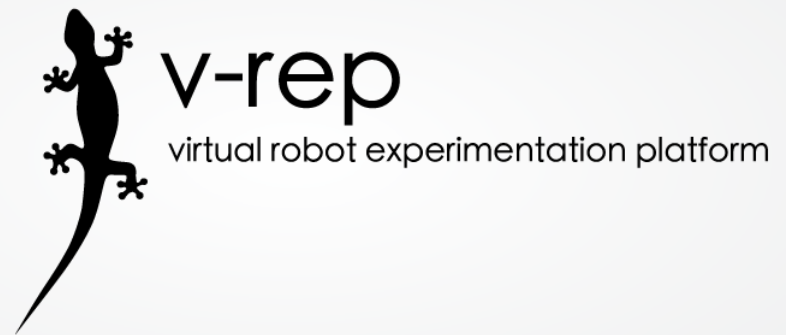


FREE EDUCATIONAL
<http://www.coppeliarobotics.com/>

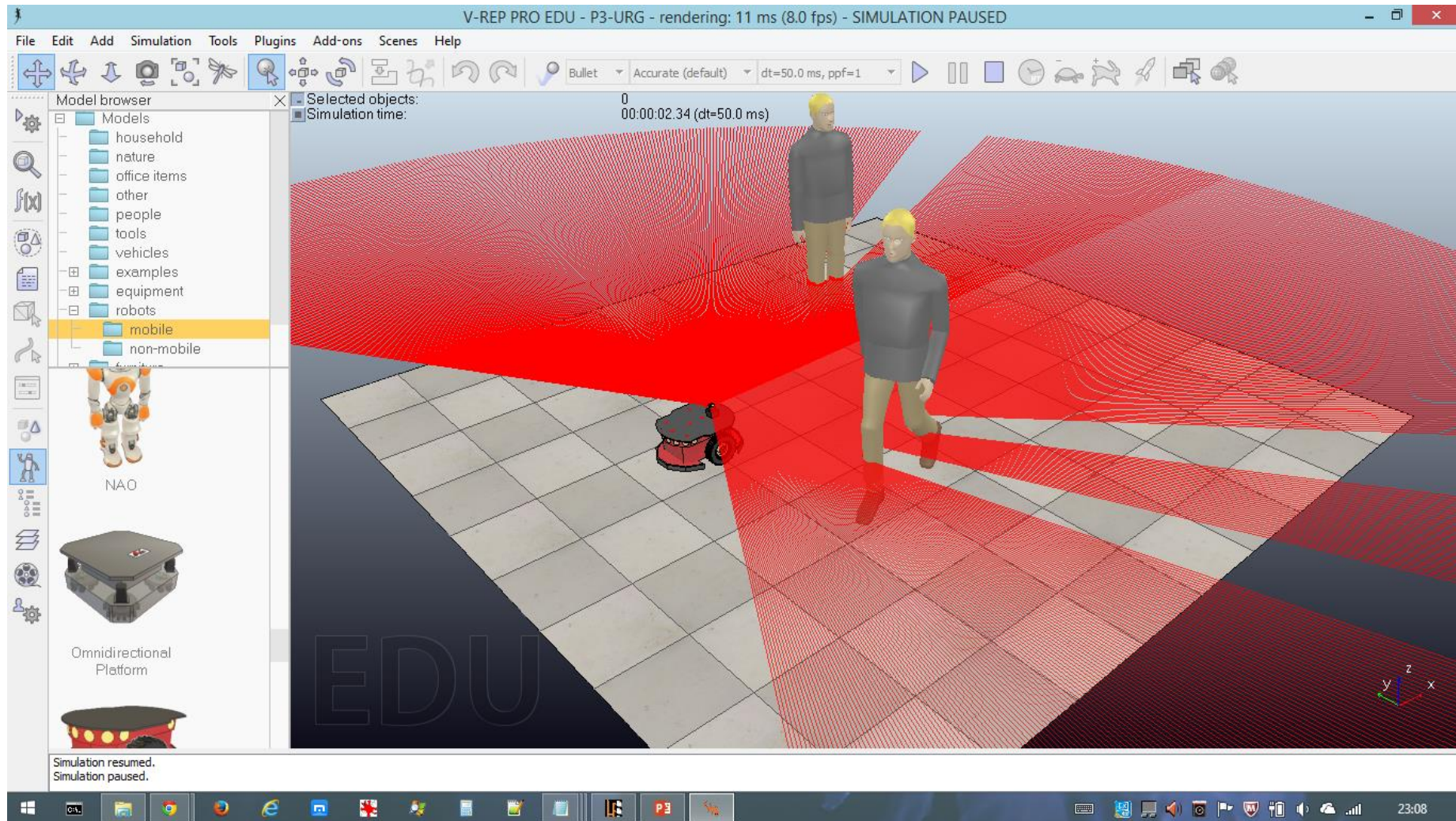


Create. Compose. Simulate. **Any Robot.**

Software V-REP



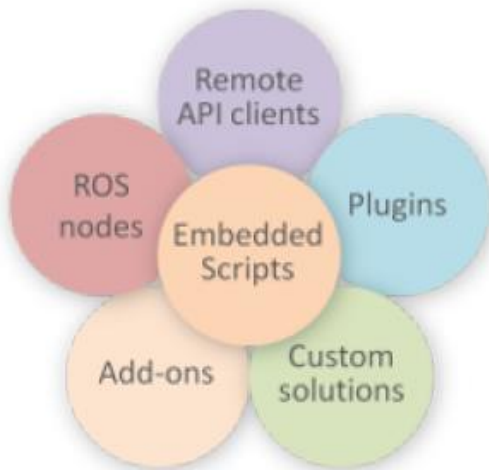
<http://www.coppeliarobotics.com/>



Simulador V-REP

VREP is cross-platform, and allows the creation of portable, scalable and easy maintainable content: a single portable file can contain a fully functional model (or scene), including control code.

6 Programming Approaches



Regular API: 400 functions (C/C++ & Lua)

Remote API: 100 functions (C/C++, Python, Java, Matlab, Octave & Urbi).

ROS interface: 100 services, 30 publisher types, & 25 subscriber types.

Remote API

LUA

C / C++

Multiple Robot Models:

Mobile Robots

Humanoids

Manipulators

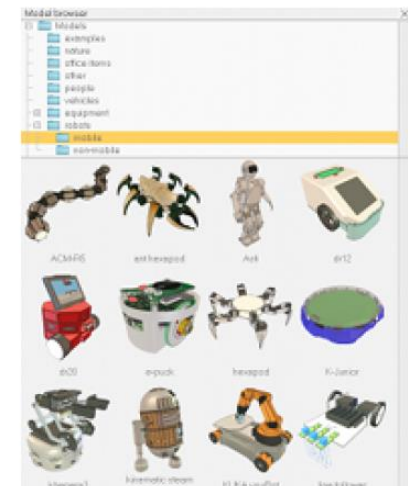
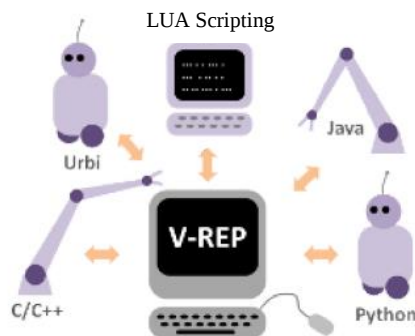
Aerial

Dynamics/Physics

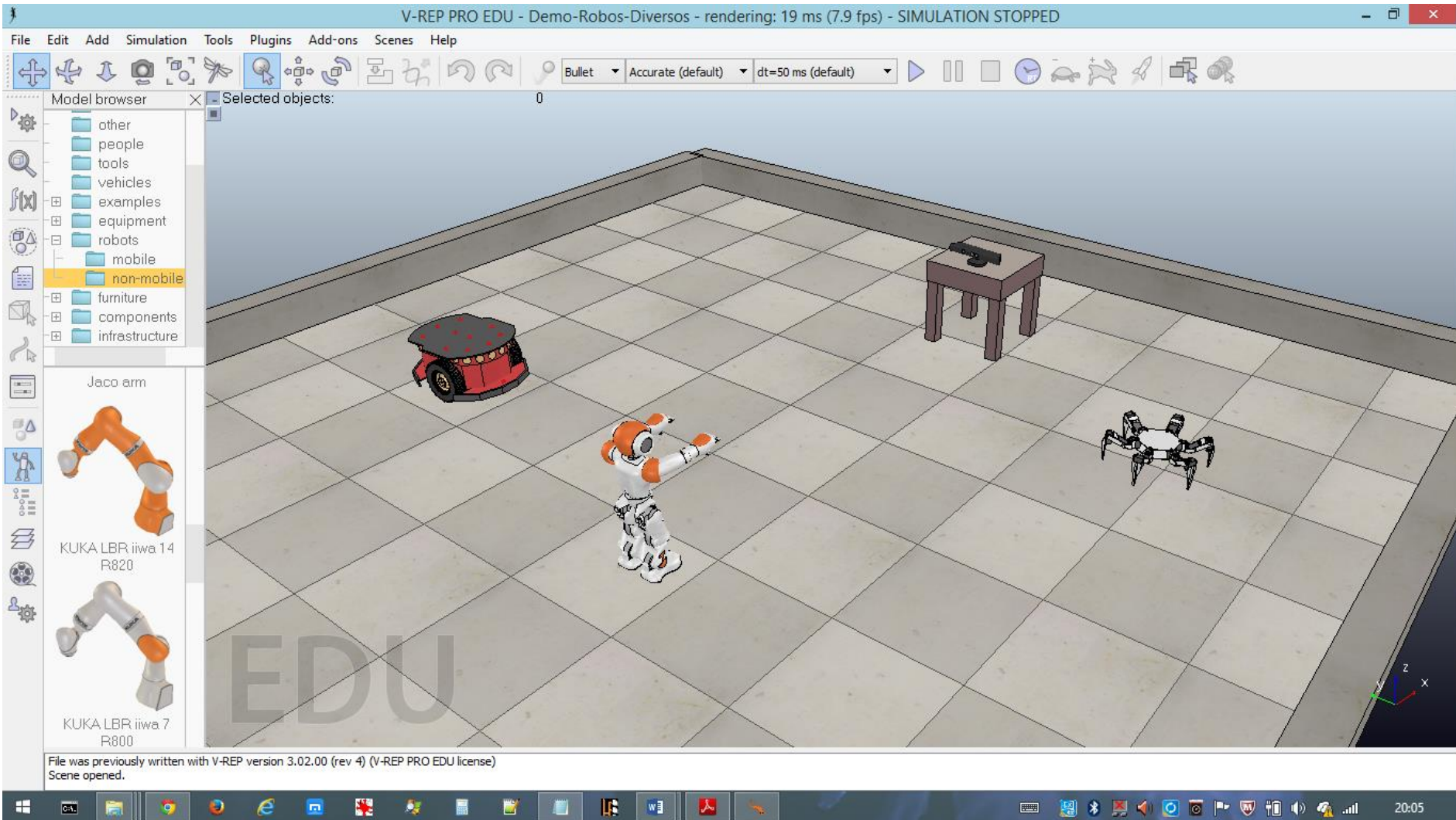
Bullet

ODE

Vortex



V-REP – Janela de aplicação



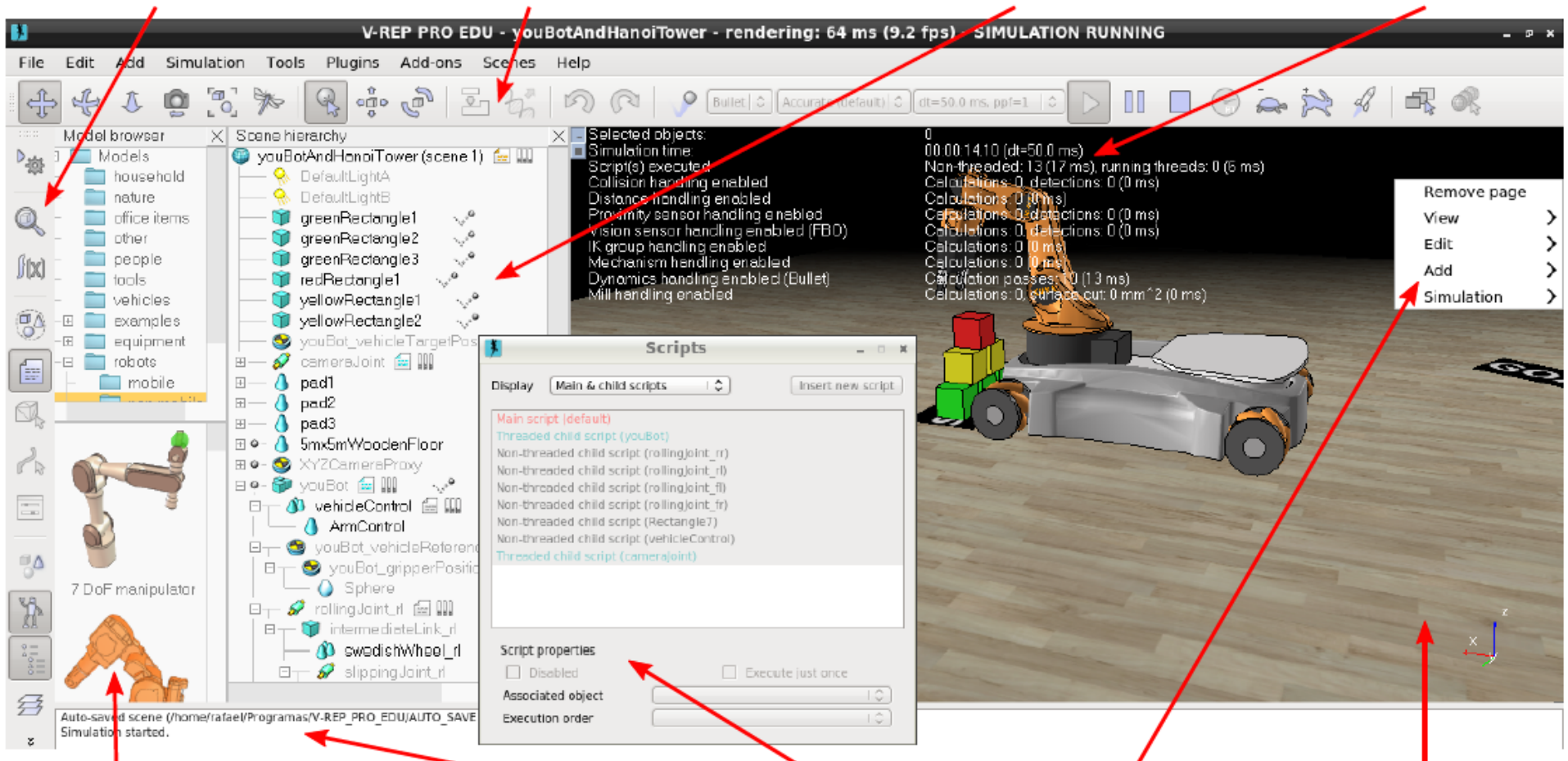
V-REP – Componentes de tela

Barra de Ferramentas 1

Barra de Ferramentas 2

Componentes Hierárquicos na Cena

Texto Informativo



Escolha de Modelos

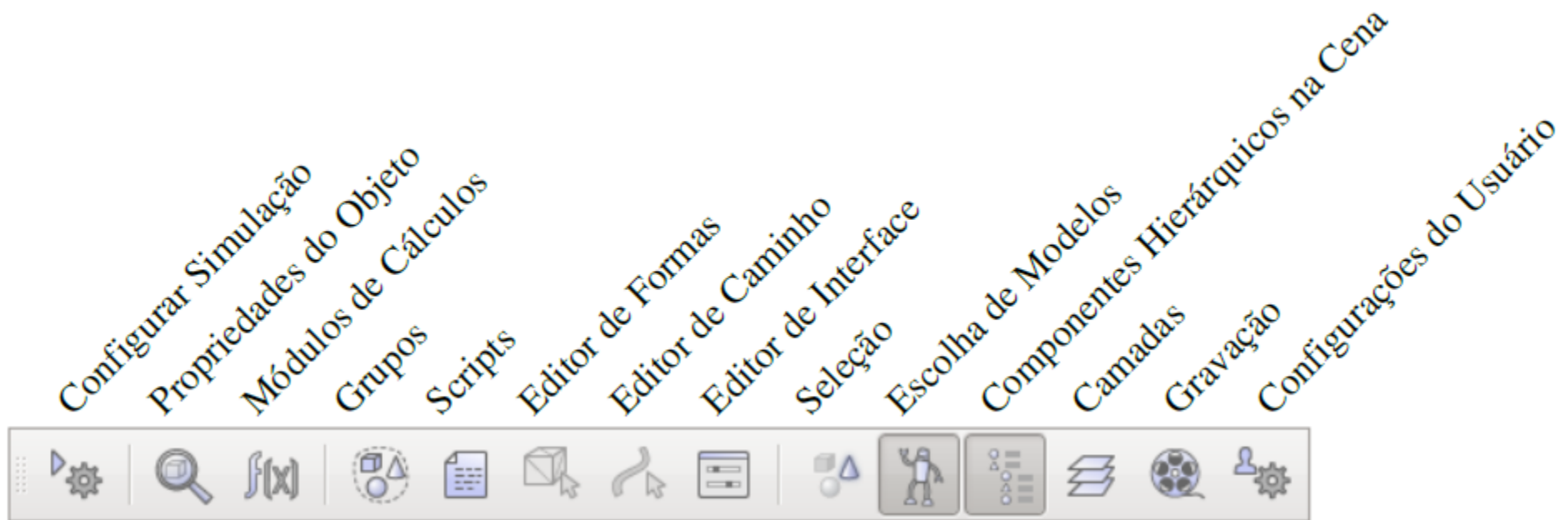
Barra de Status

Janela de Diálogo

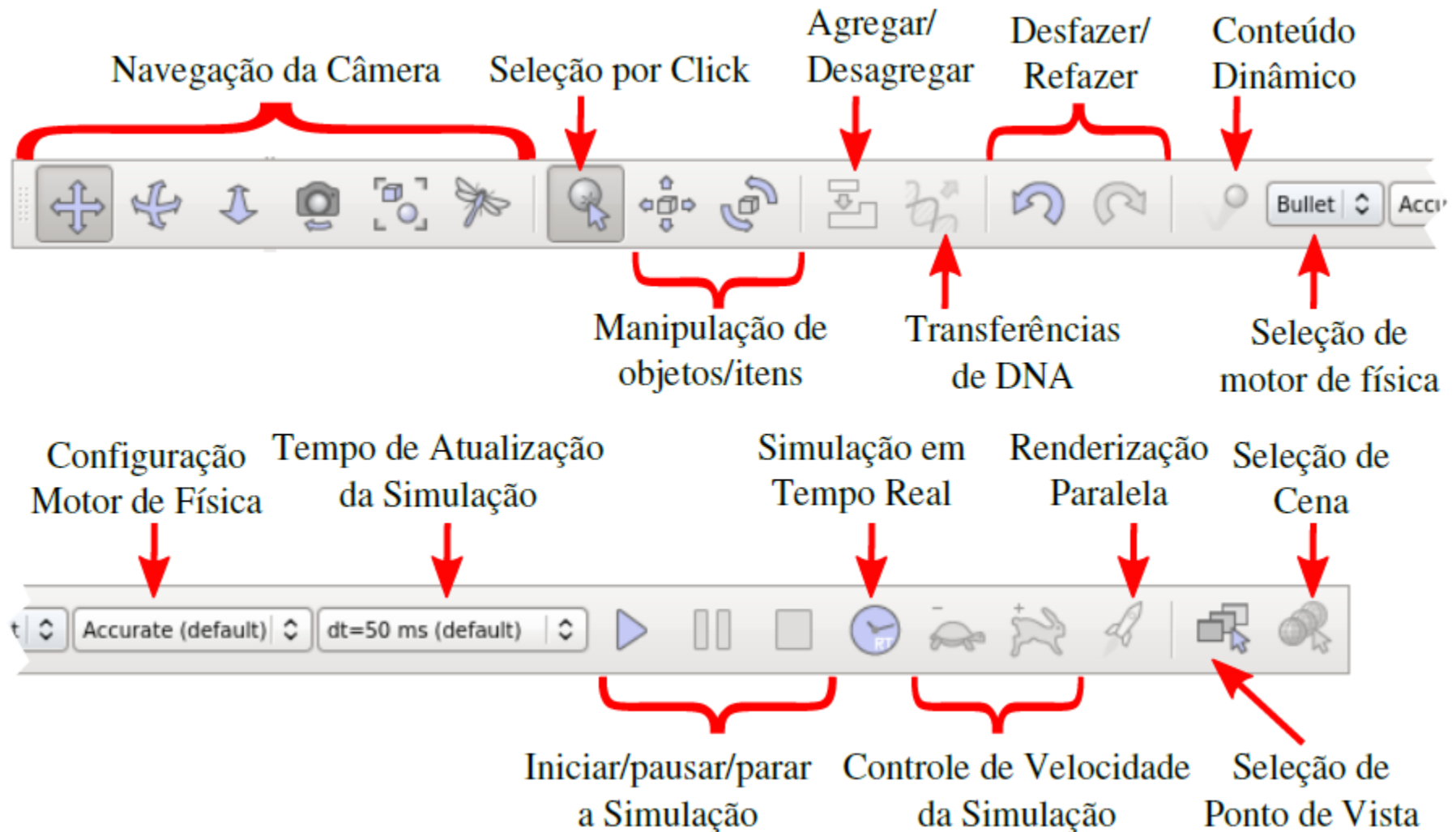
Menu Popup

Simulação (Cena)

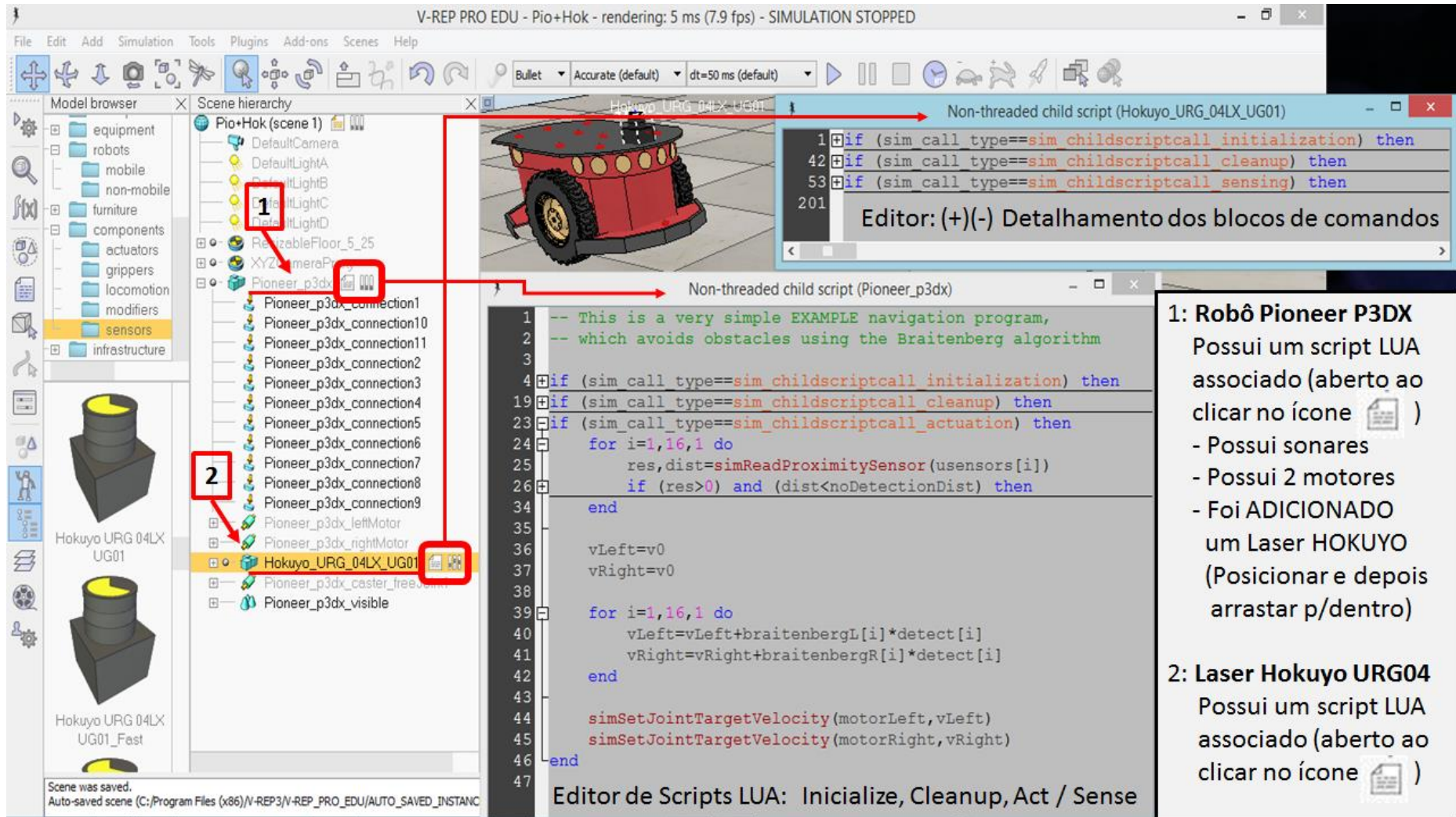
V-REP – Barra de ferramentas 1



V-REP – Barra de ferramentas 2



V-REP – Robô, Componentes, Scripts LUA



V-REP PRO EDU - Pio+Hok - rendering: 5 ms (7.9 fps) - SIMULATION STOPPED

Model browser: equipment, robots, mobile, non-mobile, furniture, components, actuators, grippers, locomotion, modifiers, sensors, infrastructure

Scene hierarchy: Pio+Hok (scene 1), DefaultCamera, DefaultLightA, DefaultLightB, DefaultLightC, DefaultLightD, ResizableFloor_5_25, XYZCameraProxy, Pioneer_p3dx, Pioneer_p3dx_connection1, Pioneer_p3dx_connection10, Pioneer_p3dx_connection11, Pioneer_p3dx_connection2, Pioneer_p3dx_connection3, Pioneer_p3dx_connection4, Pioneer_p3dx_connection5, Pioneer_p3dx_connection6, Pioneer_p3dx_connection7, Pioneer_p3dx_connection8, Pioneer_p3dx_connection9, Pioneer_p3dx_leftMotor, Pioneer_p3dx_rightMotor, **Hokuyo_URG_04LX_UG01**, Pioneer_p3dx_caster_tree, Pioneer_p3dx_visible

Hokuyo URG 04LX UG01

Hokuyo URG 04LX UG01_Fast

Scene was saved.
Auto-saved scene (C:/Program Files (x86)/V-REP3/V-REP_PRO_EDU/AUTO_SAVED_INSTANC

Non-threaded child script (Hokuyo_URG_04LX_UG01)

```
1 if (sim call type==sim_childscriptcall_initialization) then
42 if (sim call type==sim_childscriptcall_cleanup) then
53 if (sim call type==sim_childscriptcall_sensing) then
201
```

Editor: (+)(-) Detalhamento dos blocos de comandos

Non-threaded child script (Pioneer_p3dx)

```
1 -- This is a very simple EXAMPLE navigation program,
2 -- which avoids obstacles using the Braitenberg algorithm
3
4 if (sim call type==sim_childscriptcall_initialization) then
19 if (sim call type==sim_childscriptcall_cleanup) then
23 if (sim call type==sim_childscriptcall_actuation) then
24   for i=1,16,1 do
25     res,dist=simReadProximitySensor(usensors[i])
26     if (res>0) and (dist<noDetectionDist) then
34   end
35
36   vLeft=v0
37   vRight=v0
38
39   for i=1,16,1 do
40     vLeft=vLeft+braitenbergL[i]*detect[i]
41     vRight=vRight+braitenbergR[i]*detect[i]
42   end
43
44   simSetJointTargetVelocity(motorLeft,vLeft)
45   simSetJointTargetVelocity(motorRight,vRight)
46 end
47
```

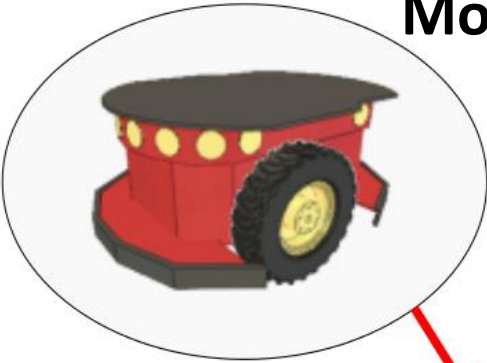
Editor de Scripts LUA: Inicialize, Cleanup, Act / Sense

1: Robô Pioneer P3DX
Possui um script LUA associado (aberto ao clicar no ícone )
- Possui sonares
- Possui 2 motores
- Foi ADICIONADO um Laser HOKUYO (Posicionar e depois arrastar p/dentro)

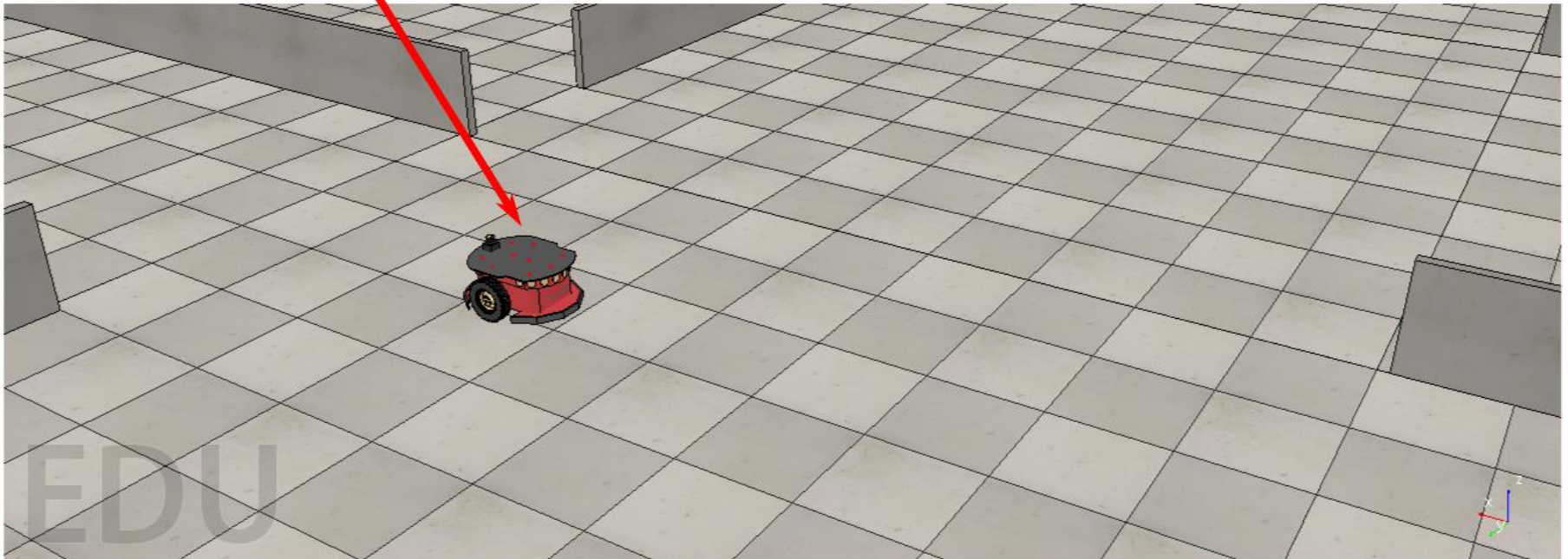
2: Laser Hokuyo URG04
Possui um script LUA associado (aberto ao clicar no ícone )

V-REP – Principais elementos

Modelo - subelemento da cena (extensão “ttm”).



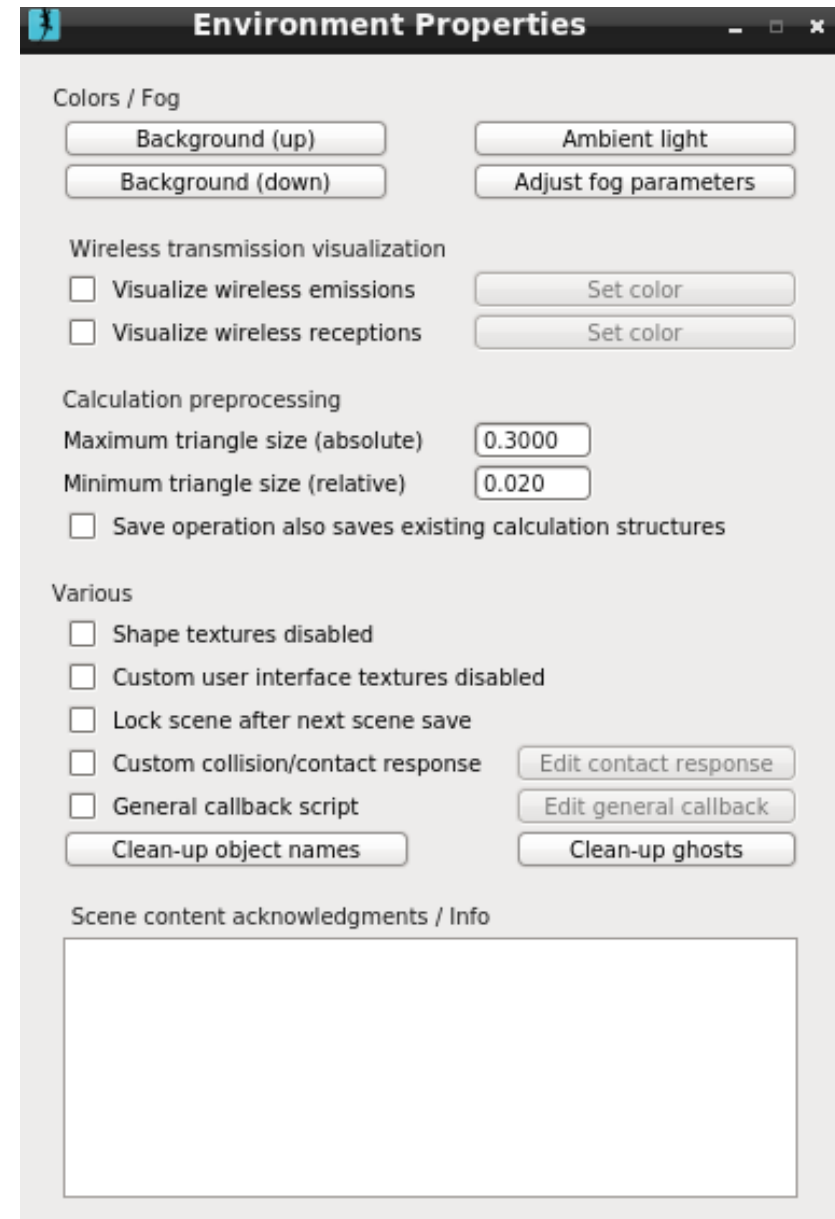
Cena - contém toda a informação necessária para a reabertura e simulação (todo o ambiente, script principal e pontos de vistas em um “ttt”).



EDU

V-REP – Ambiente

- Alguns parâmetros:
 - cores de fundo,
 - parâmetros para inclusão de nevoeiro,
 - luz ambiente,
 - informação para a criação da cena,
 - etc.
- Via menu: Tools / Environment



V-REP – Entidades

- Tipos: objeto ou uma coleção de objetos;
- Propriedades especiais de Objetos e Coleções:
 - Collidable,
 - Measurable (distância entre objetos),
 - Detectable (sensores de proximidade),
 - Cuttable,
 - Renderable (visto por sensores de visão).
- Propriedades especiais somente de objetos:
 - Viewable (objetos transparentes ou exibem imagem, ex: TV).
- Acessível em Tools / Scene object properties.

V-REP – Tipos de objetos

Tipos de Objetos:

Shape



Vision sensor



Camera



Graph



Joint



Force/torque sensor



Light



Path



Proximity sensor



Mill



Dummy

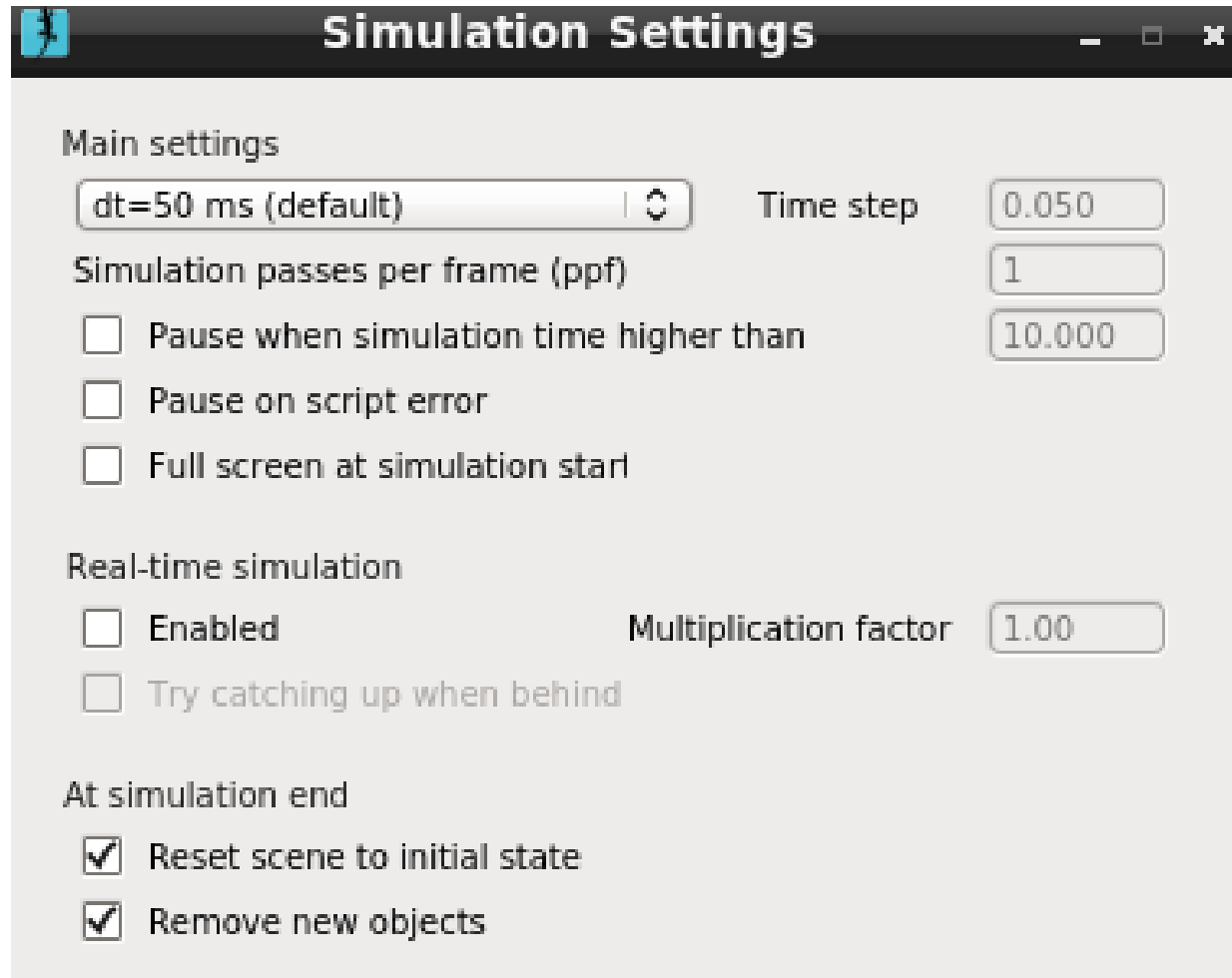


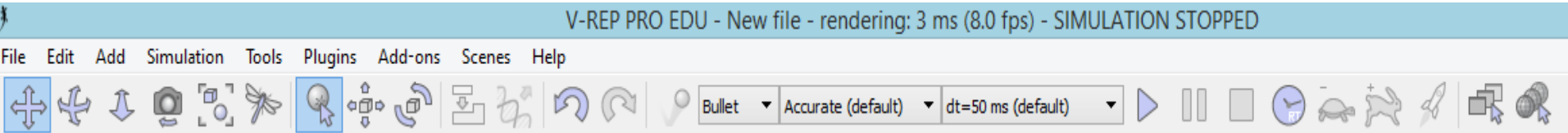
Mirror



V-REP – Configuração da Simulação

- Disponível em Simulation / Simulation Settings.





Botões de controle da Câmera Virtual 3D: Posicionamento de Observação da Cena

Principais opções: Deslocar (Pan), Girar (Rotação ao Redor dos Elementos), Zoom (Avança/Recua)



Botões de controle do Objeto Virtual

Principais opções:

Selecionar Objeto, Mover o Objeto (pode indicar/selecionar os eixos: X, Y ou Z de deslocamento),

Girar o objeto (pode selecionar os eixos: X, Y ou Z de rotação)



Botões de controle da Simulação Virtual:

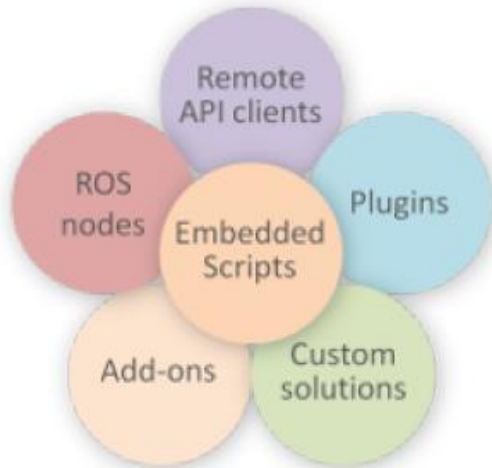
Play, Pause, Stop => Início/Fim Simulação

Acelerar ou ir mais devagar no “passo” de simulação

Simulador V-REP

VREP is cross-platform, and allows the creation of portable, scalable and easy maintainable content: a single portable file can contain a fully functional model (or scene), including control code.

6 Programming Approaches



Regular API: 400 functions (C/C++ & Lua)

Remote API: 100 functions (C/C++, Python, Java, Matlab, Octave & Urbi).

ROS interface: 100 services, 30 publisher types, & 25 subscriber types.

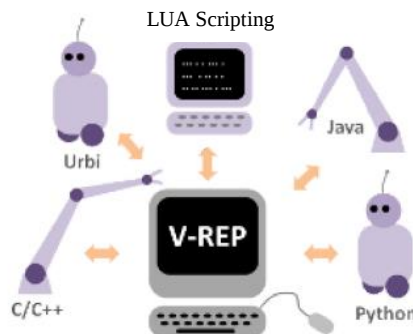
Remote API

LUA

C / C++

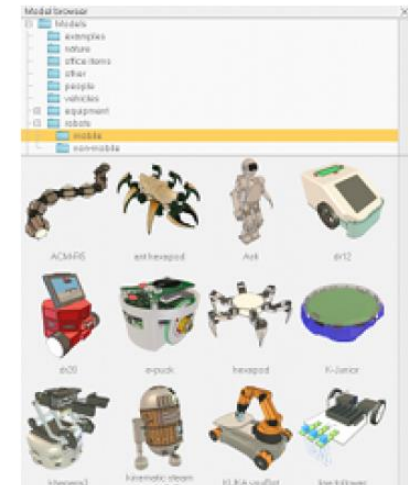


Multiple Robot Models:
Mobile Robots
Humanoids
Manipulators
Aerial

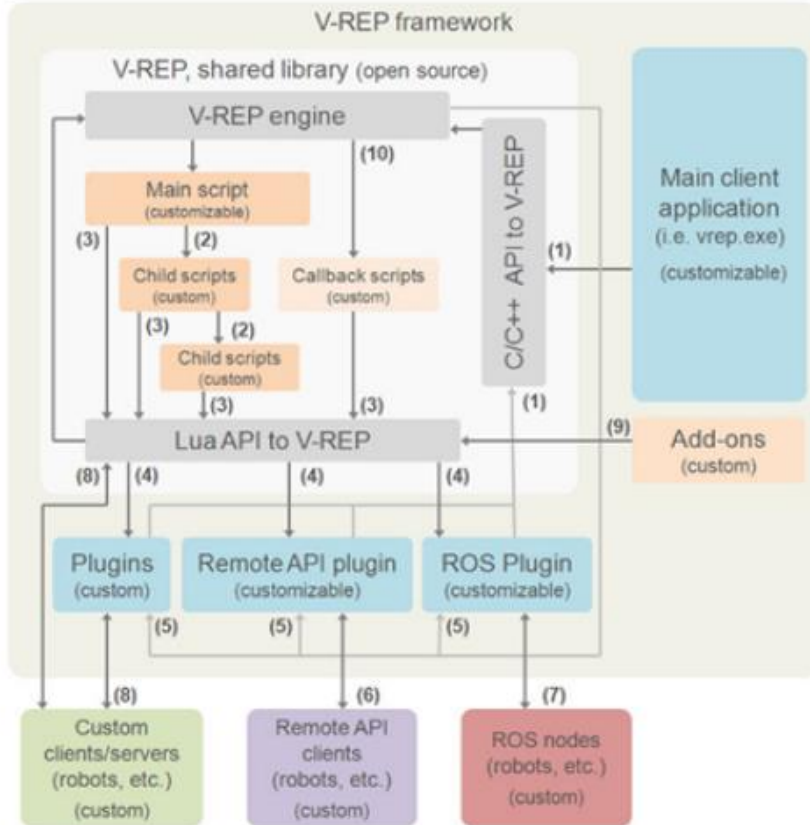


Dynamics/Physics

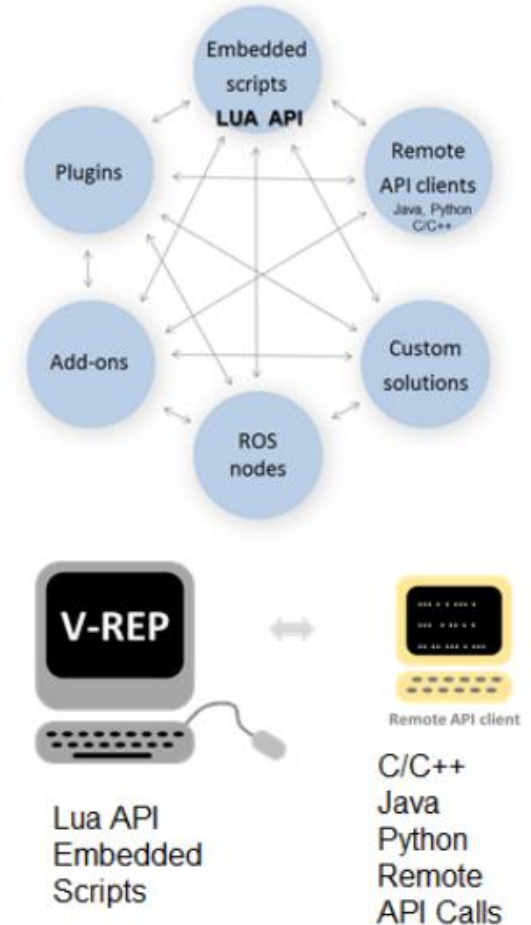
Bullet
ODE
Vortex



Visão Geral da Arquitetura do V-REP



- (1) Chamadas da API em C/C++
- (2) Execução em cascata *child script*
- (3) Chamadas da API Lua
- (4) Callbacks customizados API Lua
- (5) V-REP callbacks de eventos
- (6) Chamadas remotas das fçs da API
- (7) Interface com nodos ROS
- (8) Comunicação customizada (socket, serial, pipes, etc.)
- (9) Chamadas Add-on para API Lua
- (10) Chamadas de callback scripts



Nativo: LUA Scripts

API: C/C++, Python, Java

Client/Server, ROS, MATLAB

PROGRAMAÇÃO USANDO O V-REP



Contato:

Prof. Fernando Osório

Prof. Denis Wolf

<http://www.icmc.usp.br/~fosorio>

E-mail: { fosorio, denis } @icmc.usp.br

Laboratório de Robótica Móvel – ICMC/USP

Site: <http://www.lrm.icmc.usp.br/>

Vídeos: <http://youtube.com/lrmicmc>

<https://www.youtube.com/user/lrmicmc/videos>