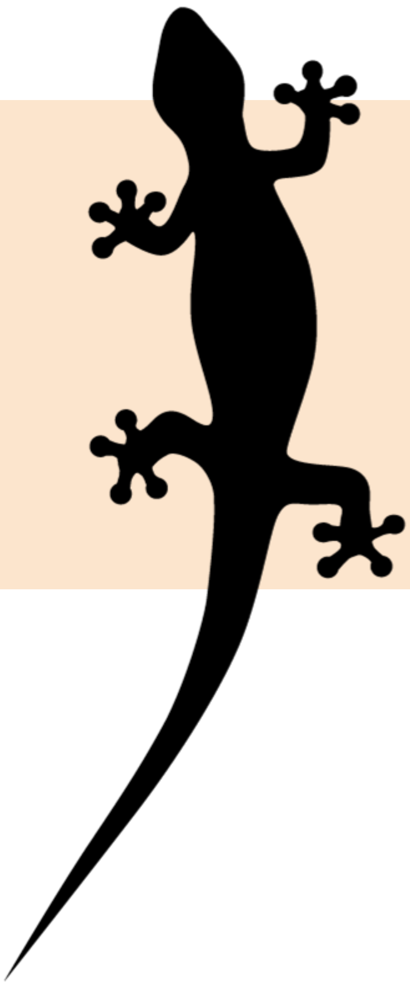


Introduction to V-REP

virtuel robot experimentation platform

by Mathias Thor

This presentation is inspired by the official V-REP presentation and the official website

Content

- ❑ V-REP Overview
- ❑ Scene Objects
- ❑ Calculation Modules
- ❑ Control Mechanisms
- ❑ Extra
- ❑ Starting your own project in GoRobots
- ❑ Questions and Getting Started

V-REP Overview

What is it?

General purpose robot simulator with integrated development environment

V-REP Overview

What is it?

General purpose robot simulator with integrated development environment

What can it do ?

Sensors, mechanisms, robots and whole systems can be modelled and simulated in various ways

V-REP Overview

What is it?

General purpose robot simulator with integrated development environment

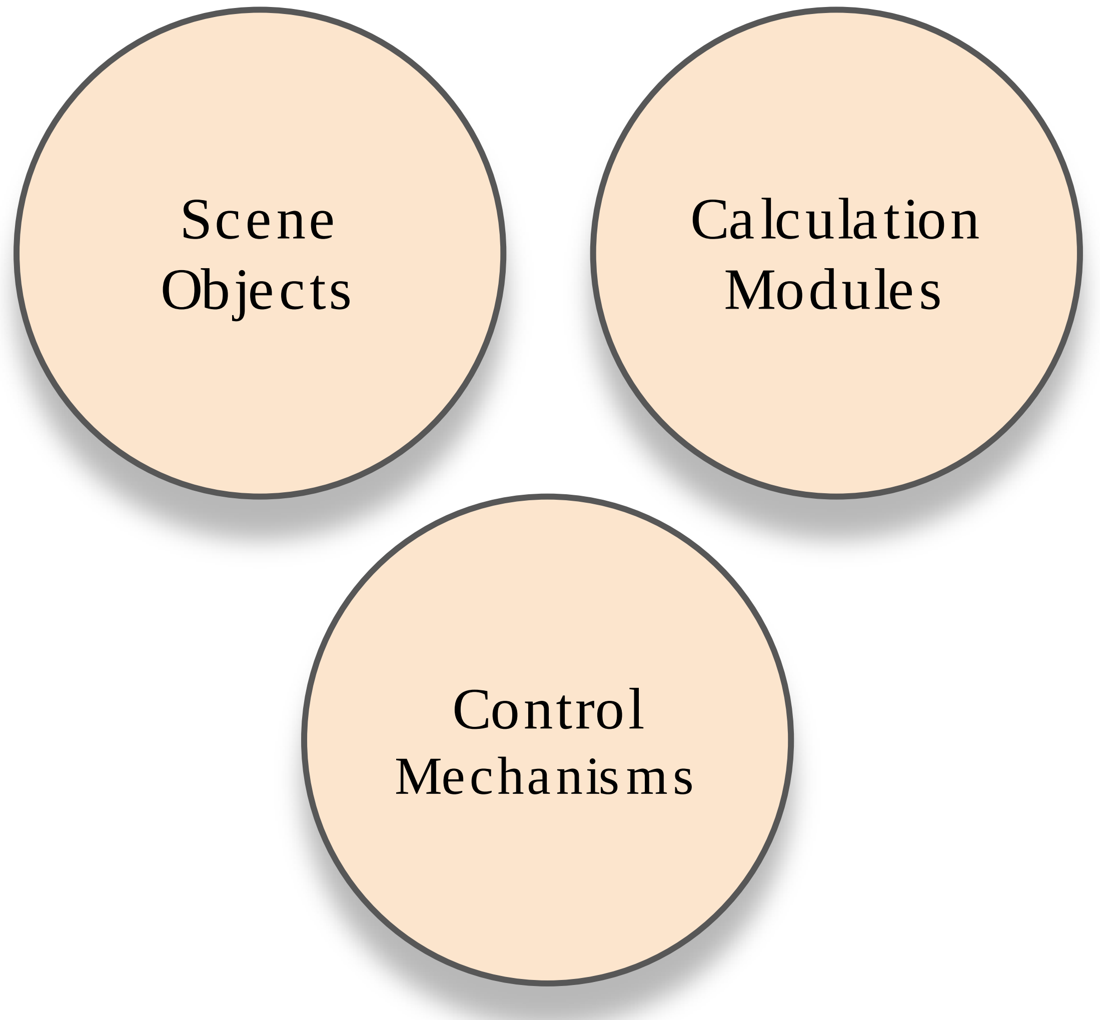
What can it do ?

Sensors, mechanisms, robots and whole systems can be modelled and simulated in various ways

Typical applications ?

Fast prototyping and verification
Controller development
Hardware control
Simulation of factory automation systems
Safety monitoring
Product presentation
etc.

Three Central Elements



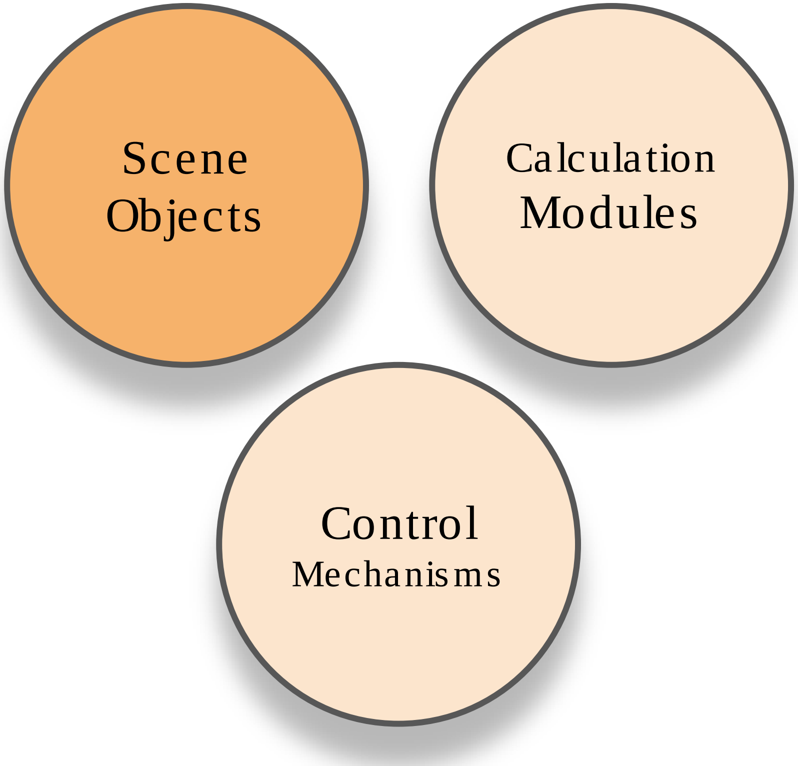
Scene Objects

Basic building blocks

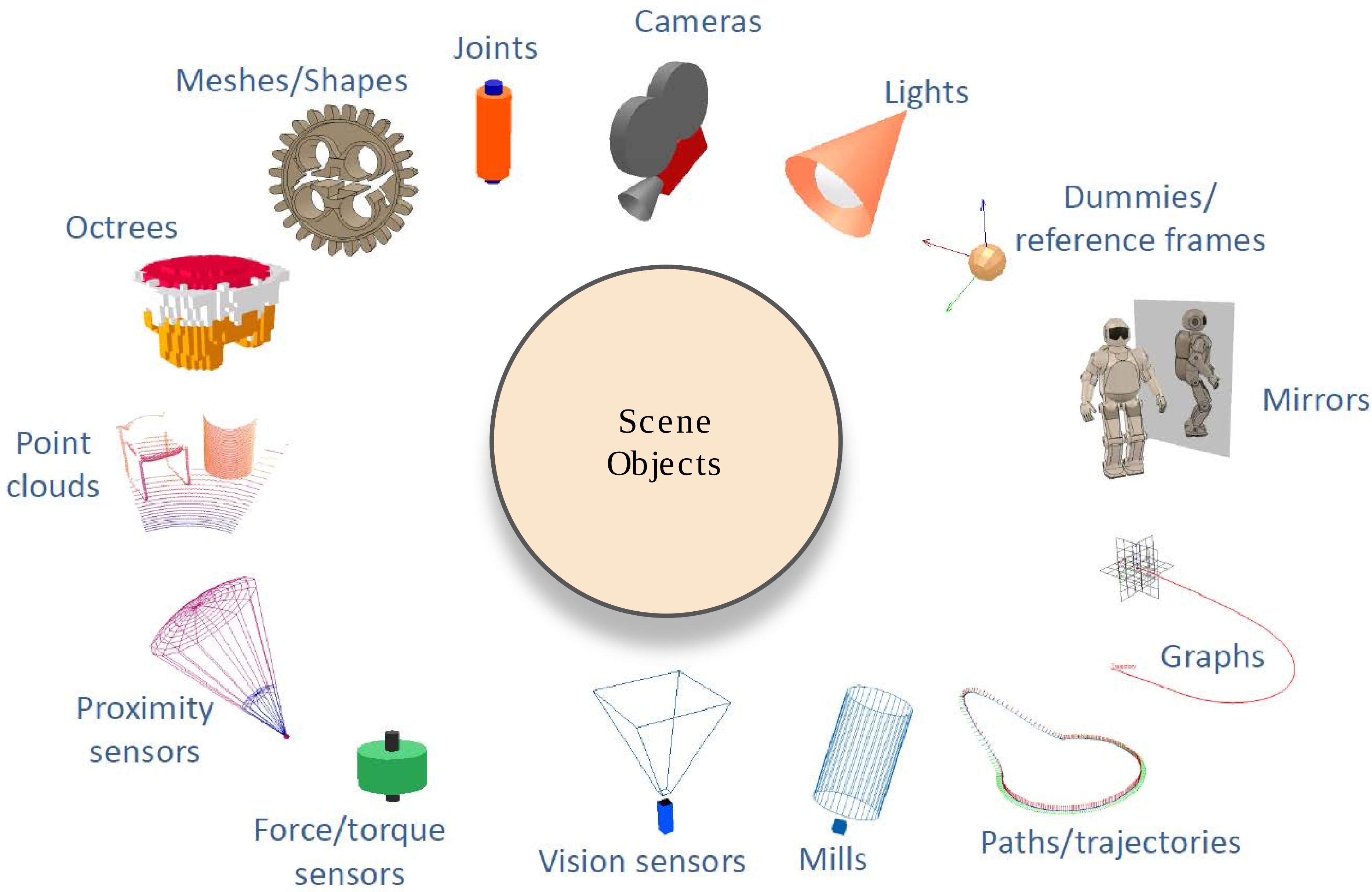
14 different types

Can be combined with each other

Can form complex systems together with calculation modules and control mechanisms



Scene Objects



Shapes and Dummies

Shapes

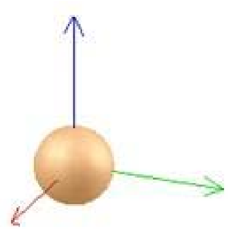
Random mesh, convex mesh, primitive mesh, or heightfield mesh

Can be grouped/ungrouped *(also merged)*

Optimized for fast calculations

Dummies

Auxiliary reference frame & helper object



Joints and Force/Torque Sensors

Joints

Revolute-type • Prismatic-type • Screw-type • Spherical-type

Velocity, position, spring/damper or force controlled

Behavior is controlled by physical engine

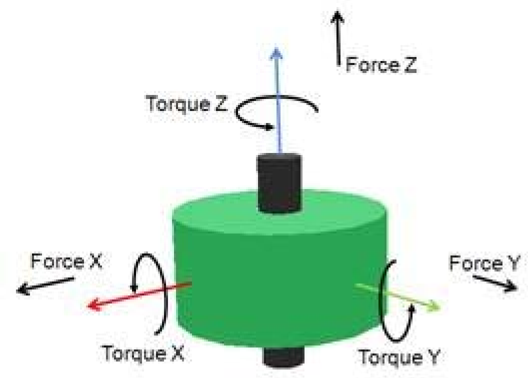
Easy to do inverse kinematics

Force/Torque Sensors

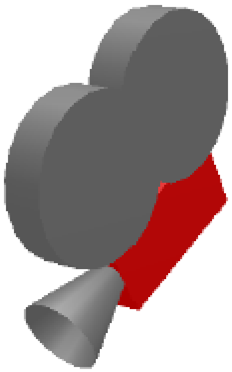
Measures force and torque

Can conditionally break apart

Used to ‘glue’ rigid parts together
(not as strong as grouping, but keeps individual dynamics)



Cameras, Lights, and Mirrors



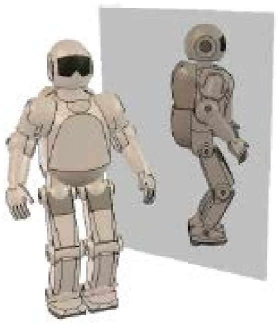
Cameras

Perspective / orthographic projection

Tracking & automatic view-fitting function

Lights

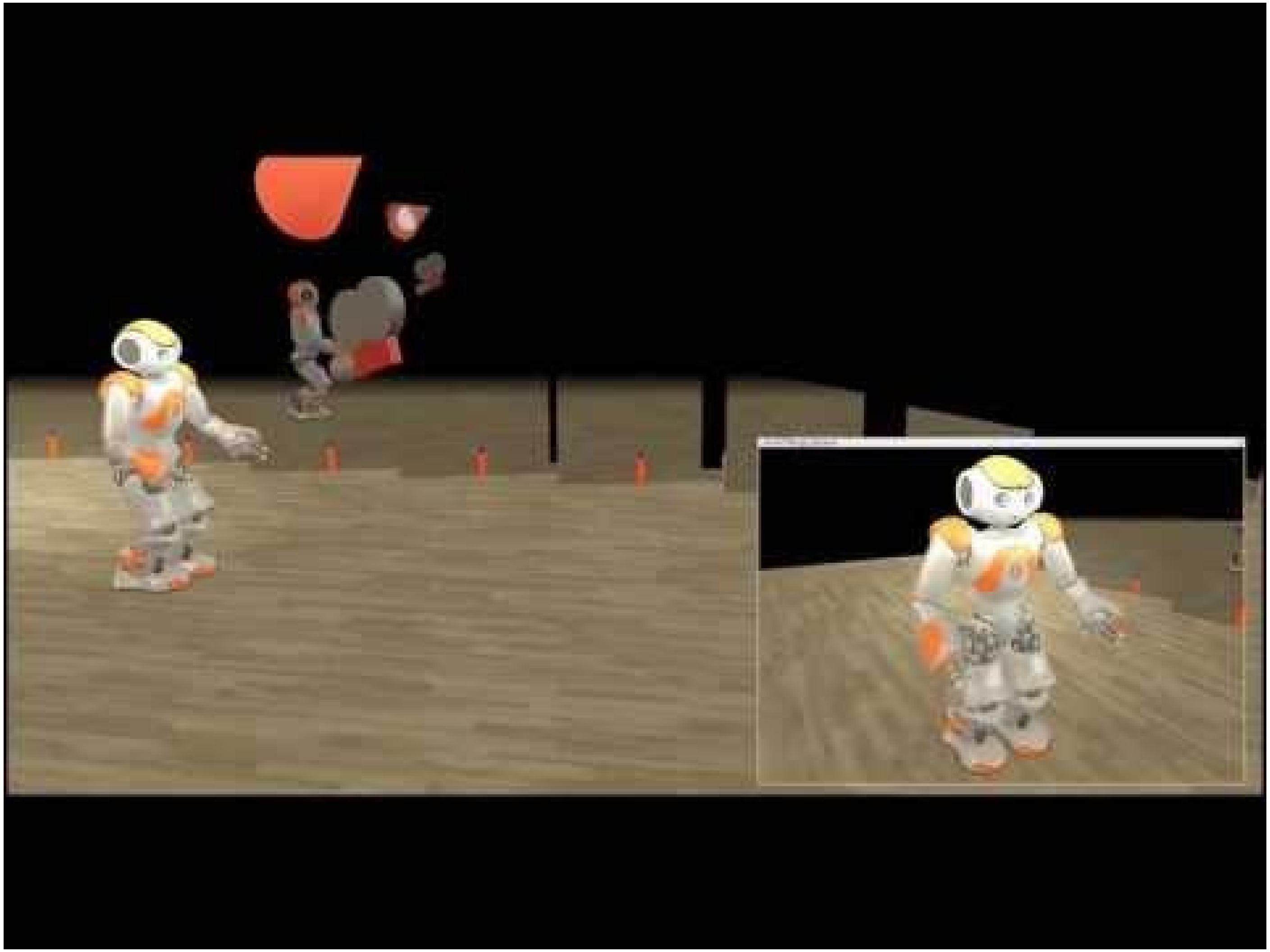
Spotlight / directional / omnidirectional



Mirrors

Mirror or scene / object clipping function

Cameras, Lights, and Mirrors

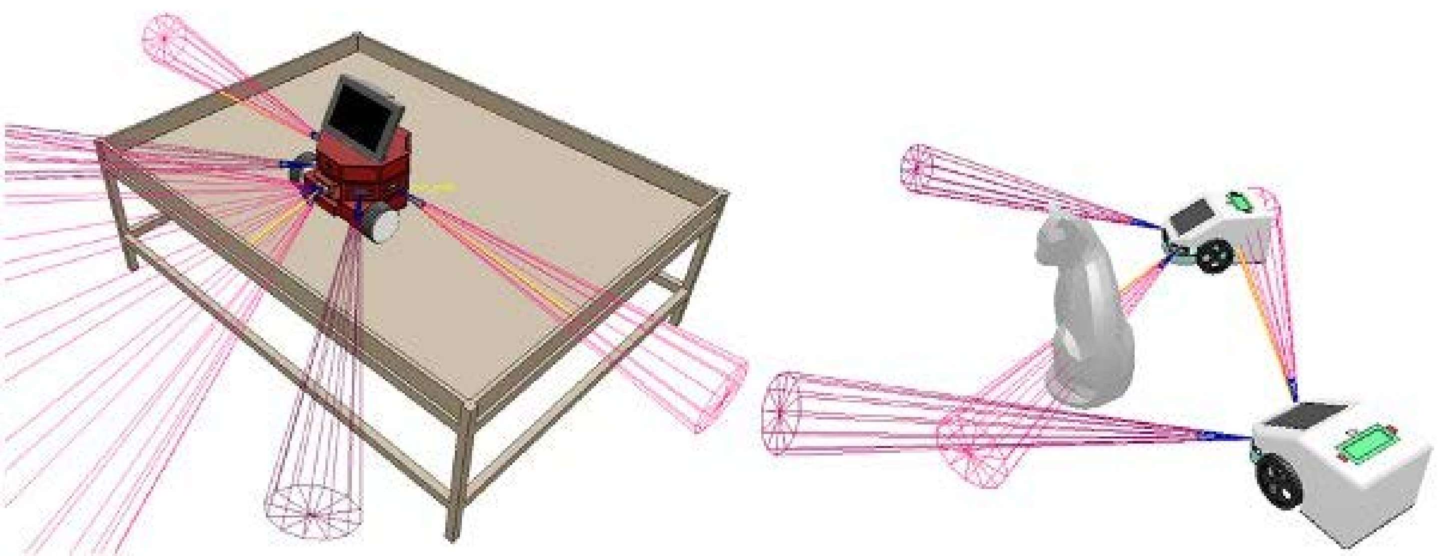


Proximity Sensors

More than simple ray-type detection

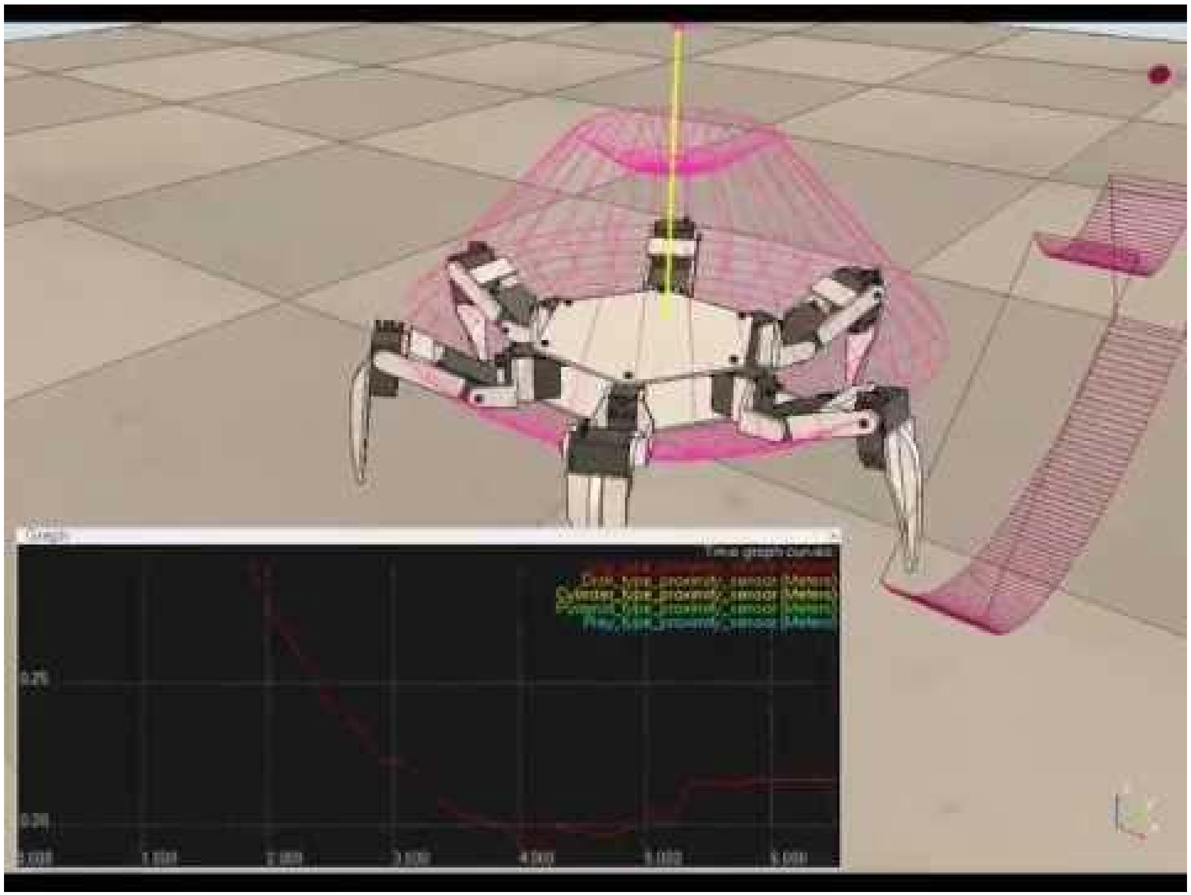
Configurable detection volume

Can be used to model almost any type of proximity sensor, from ultrasonic to infrared, and so on.



Graphs

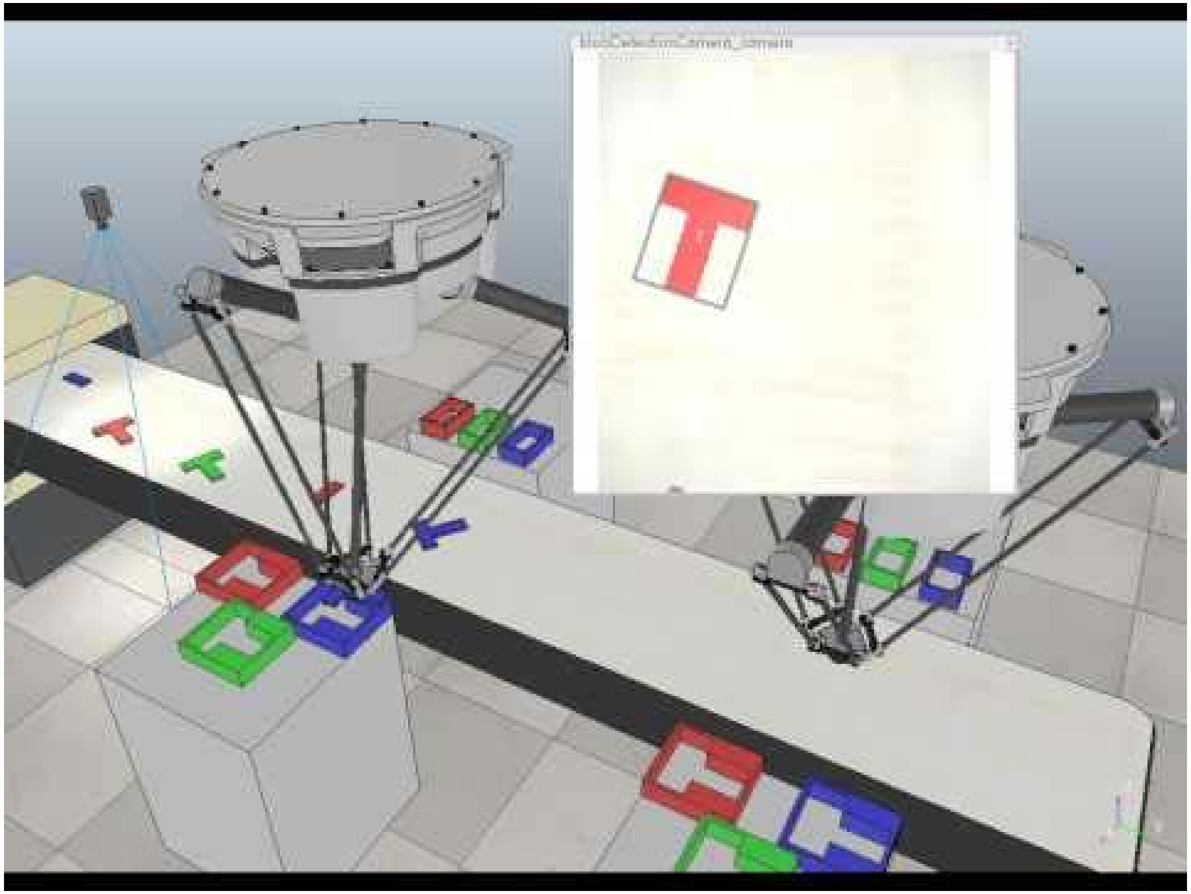
Time graphs • X/Y graphs • 3D curves • Can be exported



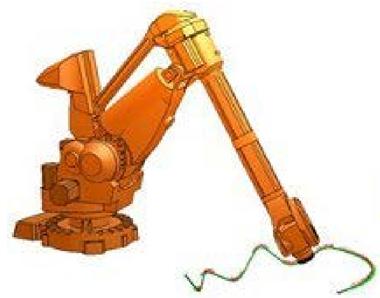
Useful debugging tool!

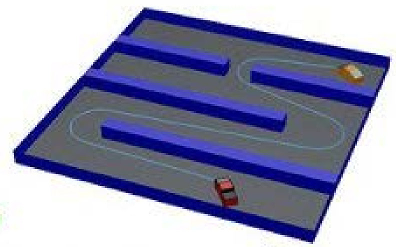
Vision Sensors

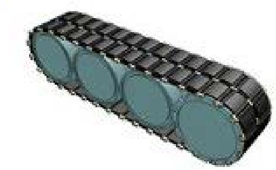
Integrated image processing • Ray-traced rendering also available



Paths and Mills



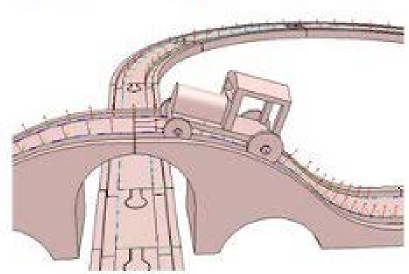


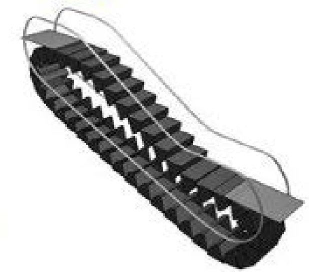


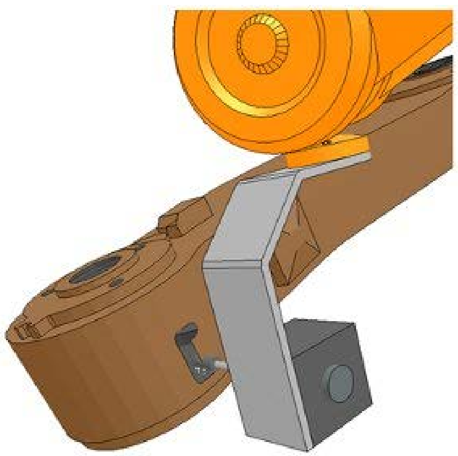
Paths

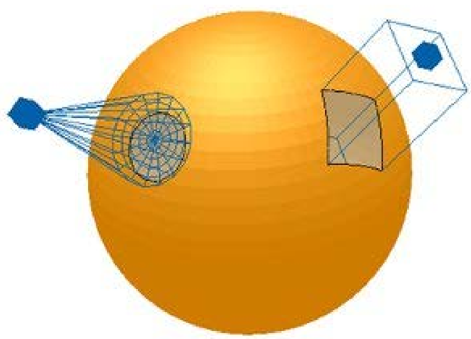
6 dim. trajectory definition

Generate a path from the edge of a shape.







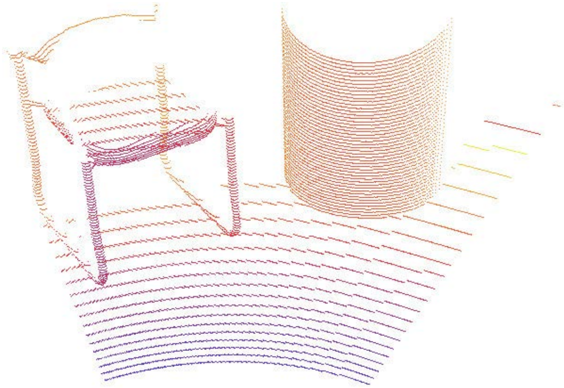


Mills

Customizable cutting volume

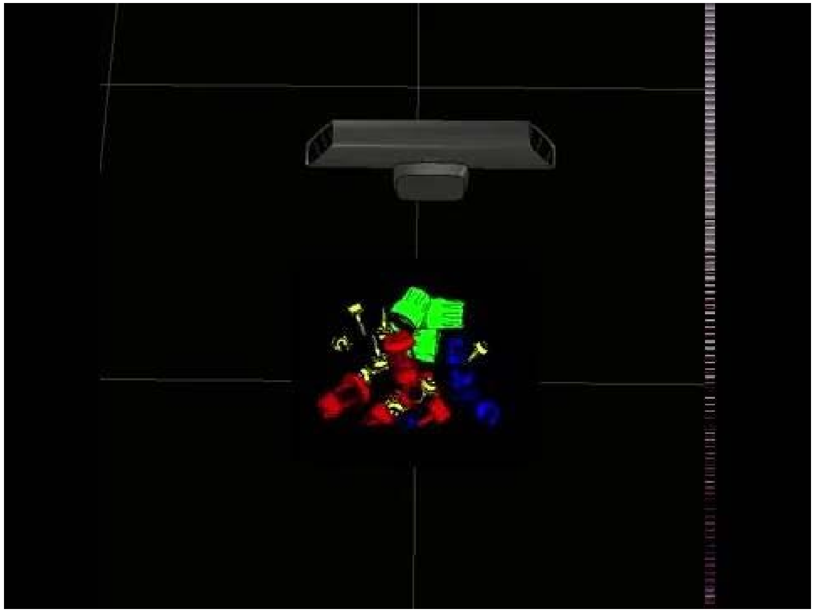
Cuts shapes (i.e. meshes)

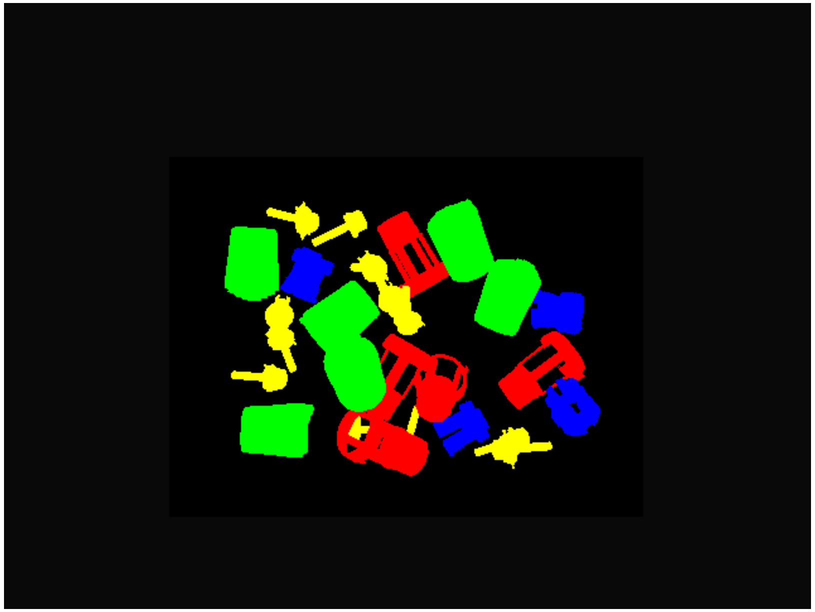
Octrees and Point Clouds



Point Clouds

Point container

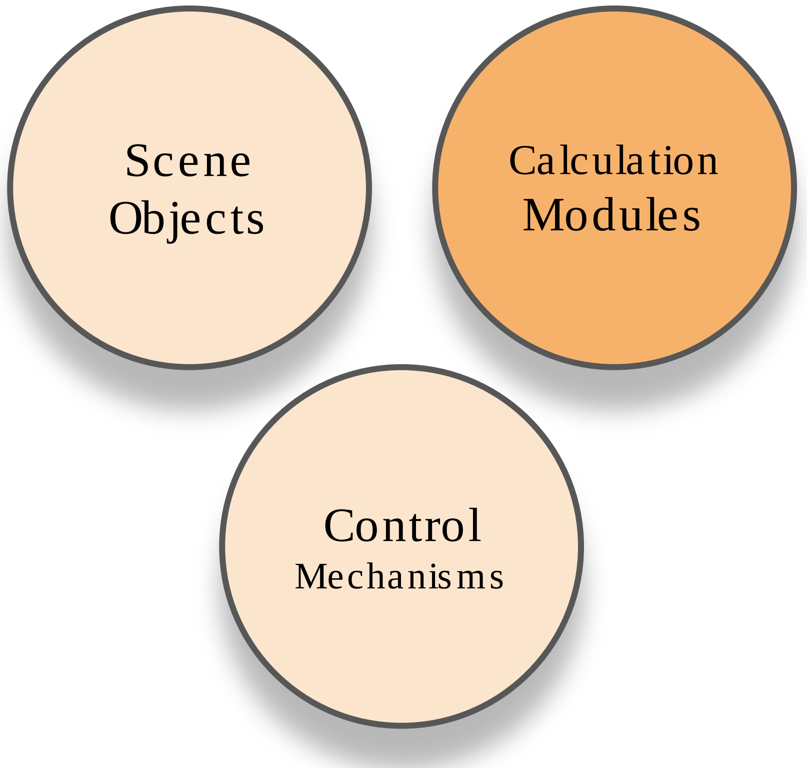




Calculation modules

5 basic algorithms

Can be combined with each other

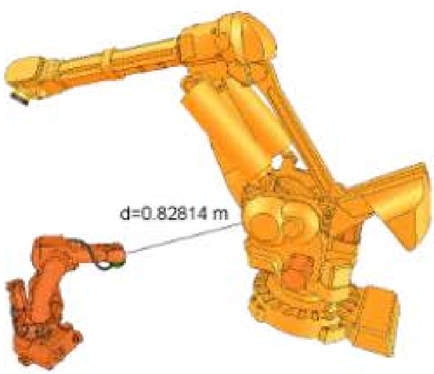


Calculation modules

Collision detection

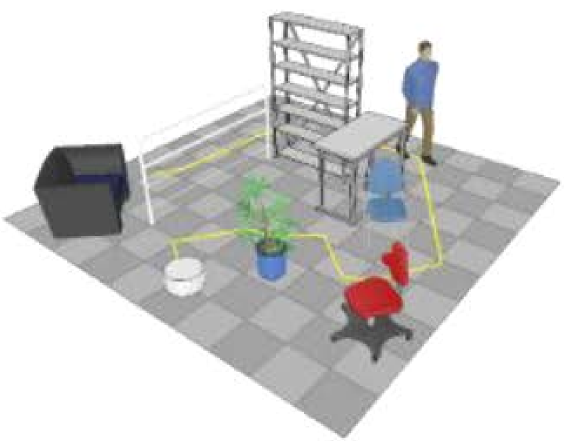
Calculation
Modules

Physics / Dynamics



Minimum distance calculation

Forward / Inverse kinematics



Path / motion planning

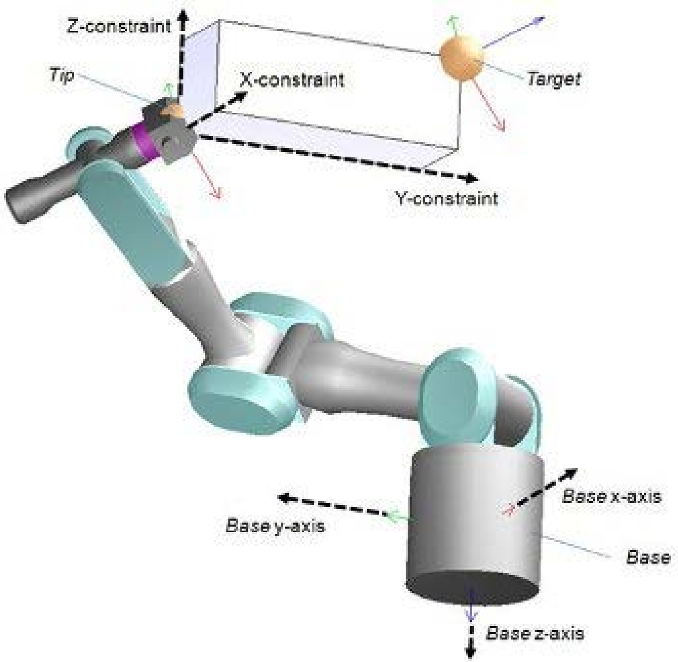
Inverse/forward kinematics & Minimum Distance Calculations

Inverse/forward kinematics

Any mechanism: redundant, branched, closed, etc.
Damped / undamped resolution

Weighted resolution • Conditional resolution

Obstacle avoidance



Minimum Distance Calculations

Any mesh • Any point cloud • Any individual point

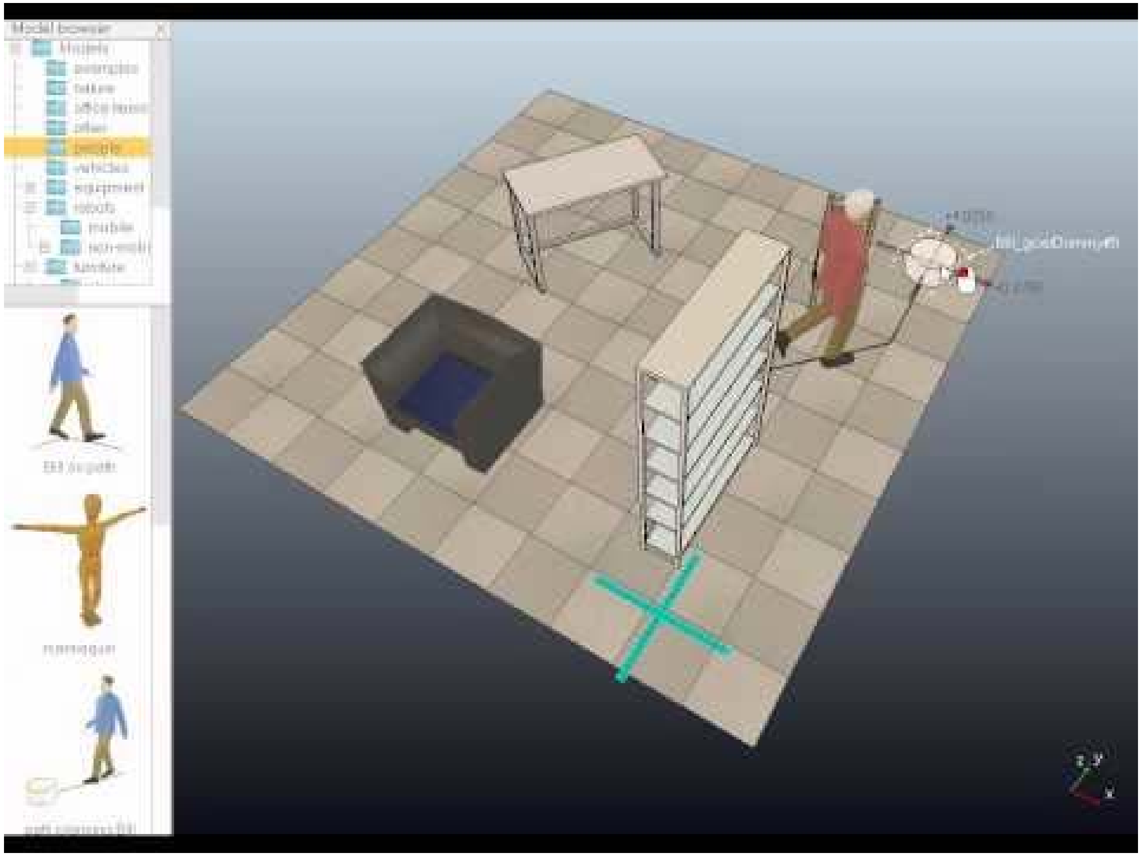
Collision Detection and Path Planning

Collision Detection

Any mesh • Any point cloud • Any individual point

Path Planning

Planning tasks in 3D-space, and in 2D-space for vehicles with non-holonomic motion constraints.



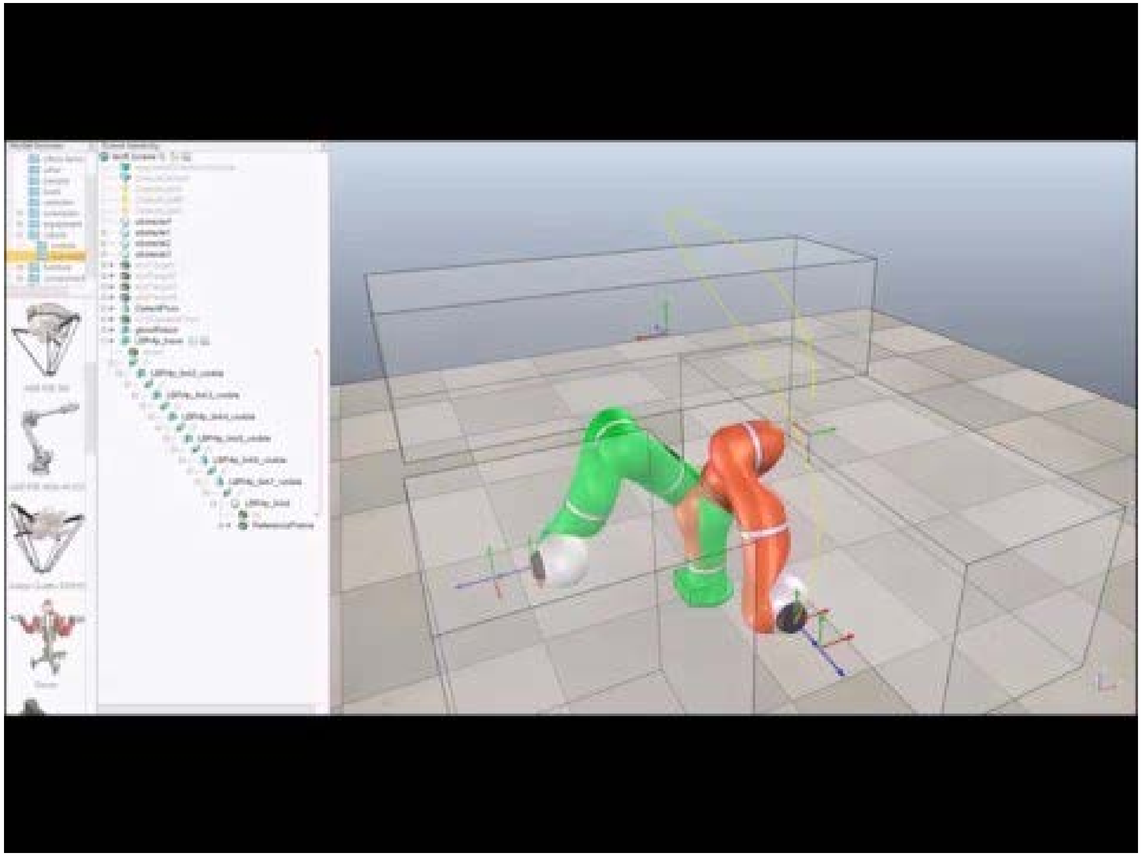
Collision Detection and Path Planning

Collision Detection

Any mesh • Any point cloud • Any individual point

Path Planning

Planning tasks in 3D-space, and in 2D-space for vehicles with non-holonomic motion constraints.



Dynamics / Physics

4 physics engines:

- ➢ Bullet Physics
- ➢ Open Dynamics Engine (ODE)
- ➢ Vortex Dynamics
- ➢ Newton Dynamics

Simple switching

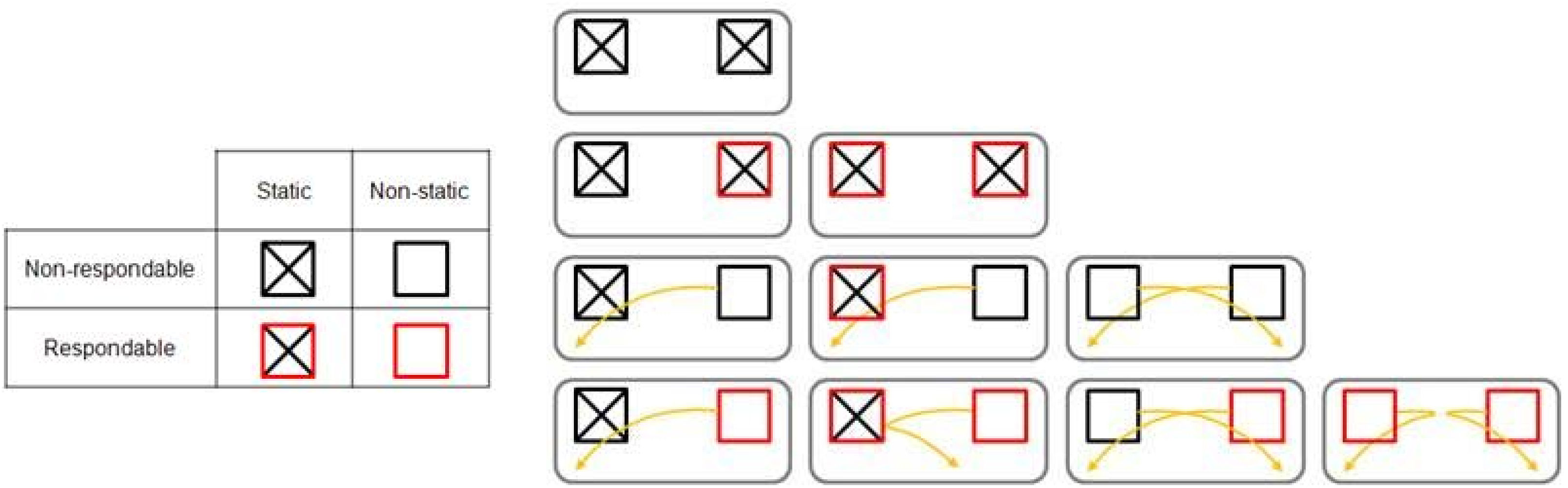
Dynamic particles to simulate air or water jets

Physics Engines Properties - Joints

Property	Value
Apply all properties to selected joints	Apply
Bullet properties	
Normal CFM	0.0000e+00
Stop ERP	2.0000e-01
Stop CFM	0.0000e+00
ODE properties	
Normal CFM	1.0000e-05
Stop ERP	6.0000e-01
Stop CFM	1.0000e-05
Bounce	0.0000e+00
Fudge factor	1.0000e+00
Vortex properties	
Joint axis friction (off)	
Joint axis limits	
Joint dependency	
X axis position (relaxation: off, friction: off)	
Y axis position (relaxation: off, friction: off)	
Z axis position (relaxation: off, friction: off)	
X axis orientation (relaxation: off, friction: off)	
Y axis orientation (relaxation: off, friction: off)	
Z axis orientation (n/a)	
Newton properties	
Joint dependency	

Dynamics

Static/non-static shapes
respondable/non-respondable shapes *(masks)*

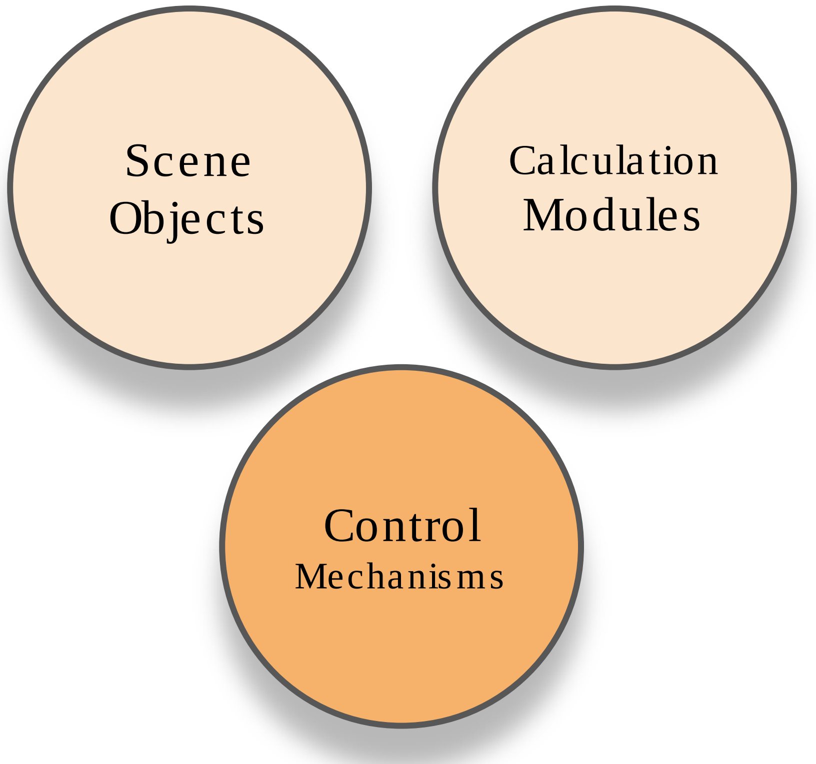


Control Mechanisms

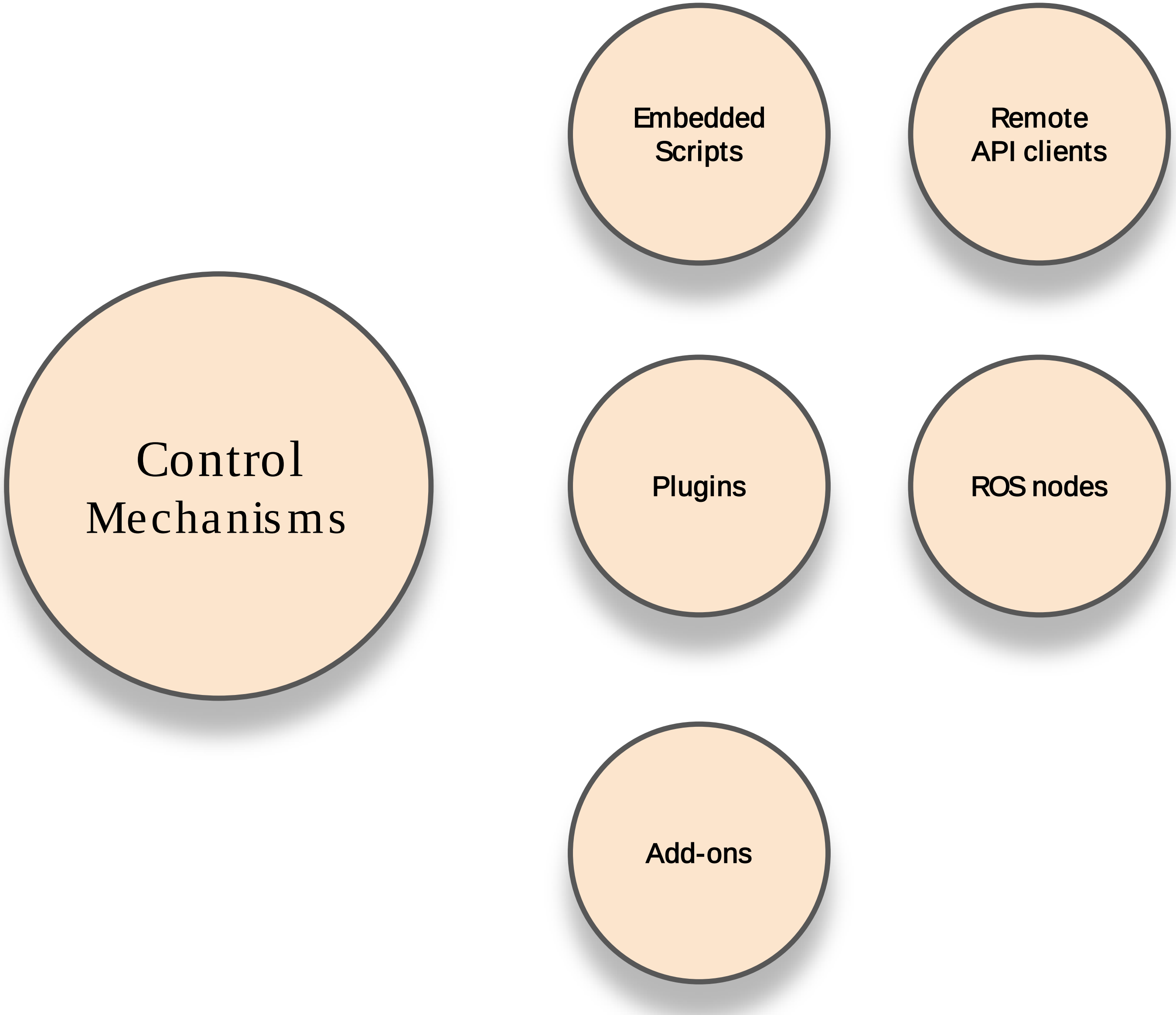
6 methods or interfaces

7 languages

All methods can be used at the same time,
and even work hand-in-hand



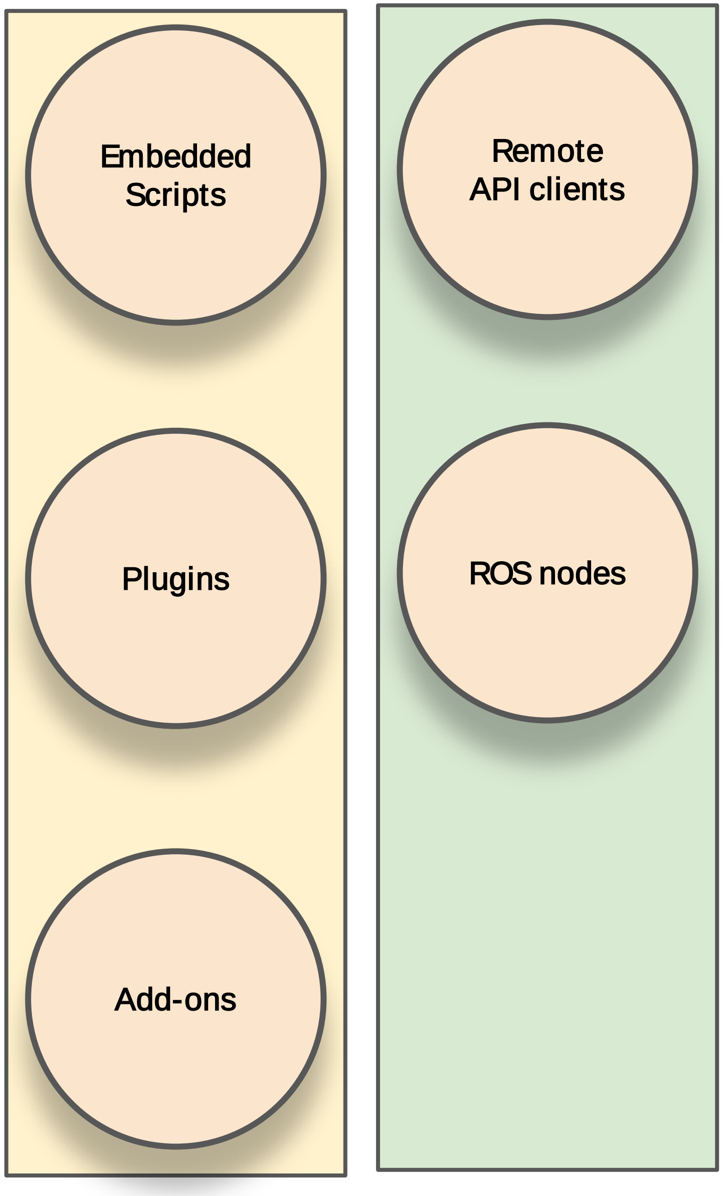
Control Mechanisms



Control Mechanisms

Local Interface

Same process



Remote Interface

Different process / hardware

Control Mechanisms

Embedded Scripts

Plugins

Add-ons

Over 500 API functions
(Extendable)

Can be attached to any scene object

Many Lua extension libraries available

Threaded or non-threaded

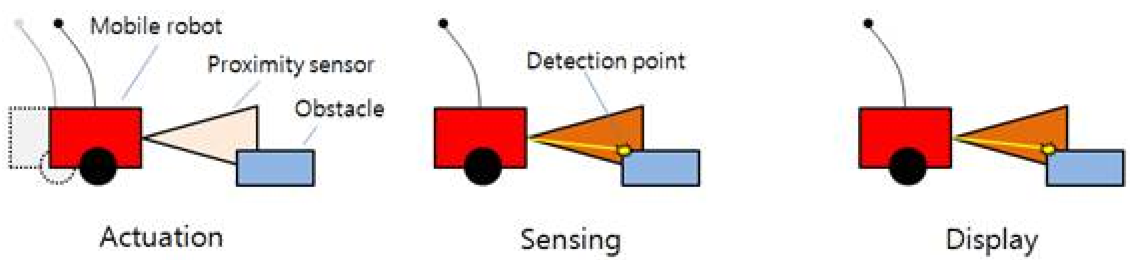
Various types: main script, child scripts, callback scripts
(e.g. custom joint controllers)

Lua interface

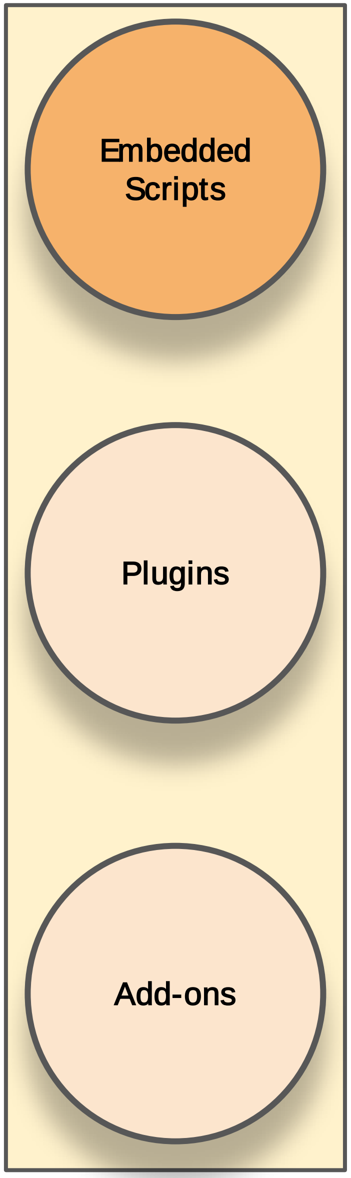
Lightweight and easy to program (child scripts)

→ Initialization • Actuation • Sensing • shut down

Extremely portable solution



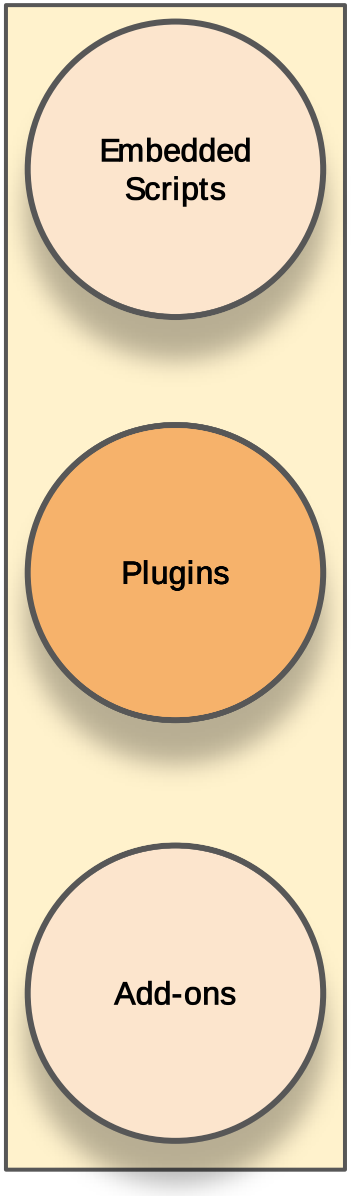
Control Mechanisms



Non-threaded child script

```
function sysCall_init()
    sensorHandleFront=sim.getObjectHandle("DoorSensorFront")
    sensorHandleBack=sim.getObjectHandle("DoorSensorBack")
    motorHandle=sim.getObjectHandle("DoorMotor")
end

function sysCall_actuation()
    resF=sim.readProximitySensor(sensorHandleFront)
    resB=sim.readProximitySensor(sensorHandleBack)
    if ((resF>0)or(resB>0)) then
        sim.setJointTargetVelocity(motorHandle,-0.2)
    else
        sim.setJointTargetVelocity(motorHandle,0.2)
    end
end

function sysCall_sensing()

end

function sysCall_cleanup()
    -- Put some restoration code here
end
```

Control Mechanisms



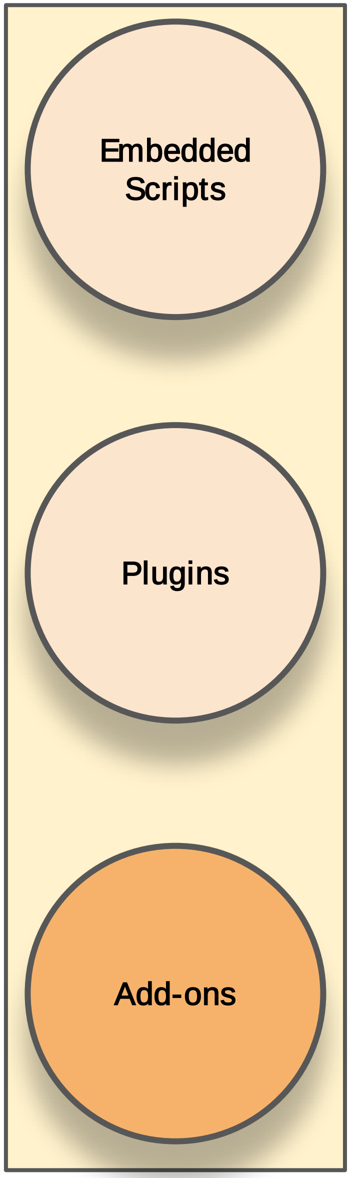
Over 500 API functions. Extendable

C/C++ interface

Used to provide V-REP with a special functionality requiring either fast calculation capability (scripts are most of the time slower than compiled languages), a specific interface to a hardware device (e.g. a real robot), or a special communication interface with the outside world

The ROS functionality is placed in a plugin

Control Mechanisms



Over 400 API functions. Extendable

Lua interface

Can customize the simulator

Lightweight and easy to set-up

Can start automatically and run in the background, or they can be called as functions

Control Mechanisms

Remote API clients

ROS nodes

Over 100 API functions
(Extendable)

Client (your program) - Server (v-rep plugin) relationship

C/ C++, Python, Java, Matlab, Octave, Lua & Urbi interfaces

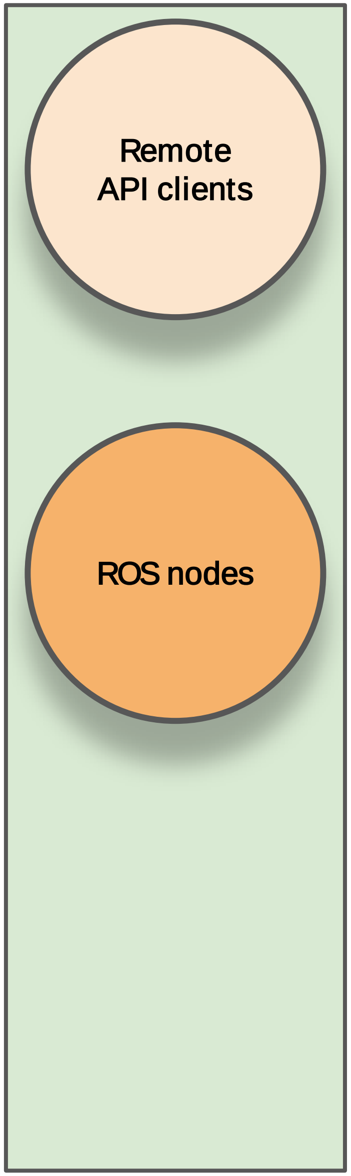
Lightweight and easy to use

Python Example

```python
import vrep

# close any open connections
vrep.simxFinish(-1)
# Connect to the V-REP continuous server
clientID = vrep.simxStart('127.0.0.1', 19997, True, True, 500, 5)

if clientID != -1: # if we connected successfully
    print ('Connected to remote API server')
```

```python
# ... calculate u ...

for ii,joint_handle in enumerate(joint_handles):
    # get the current joint torque
    _, torque = \
        vrep.simxGetJointForce(clientID,
                    joint_handle,
                    vrep.simx_opmode_blocking)
    if _ !=0 : raise Exception()
```

Control Mechanisms



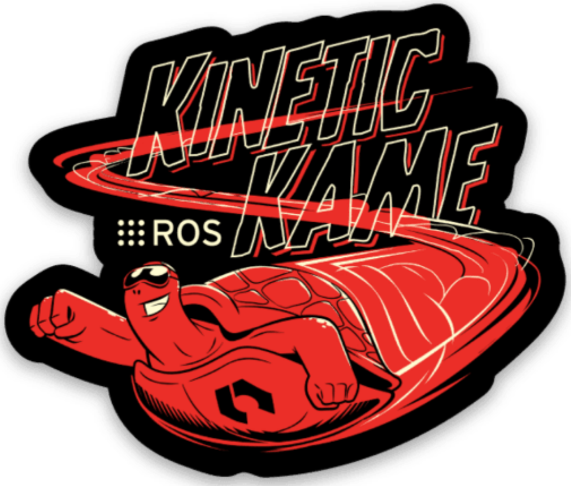
Plugin-based

It acts as a ROS node that other nodes can communicate with via ROS services, ROS publishers and ROS subscribers

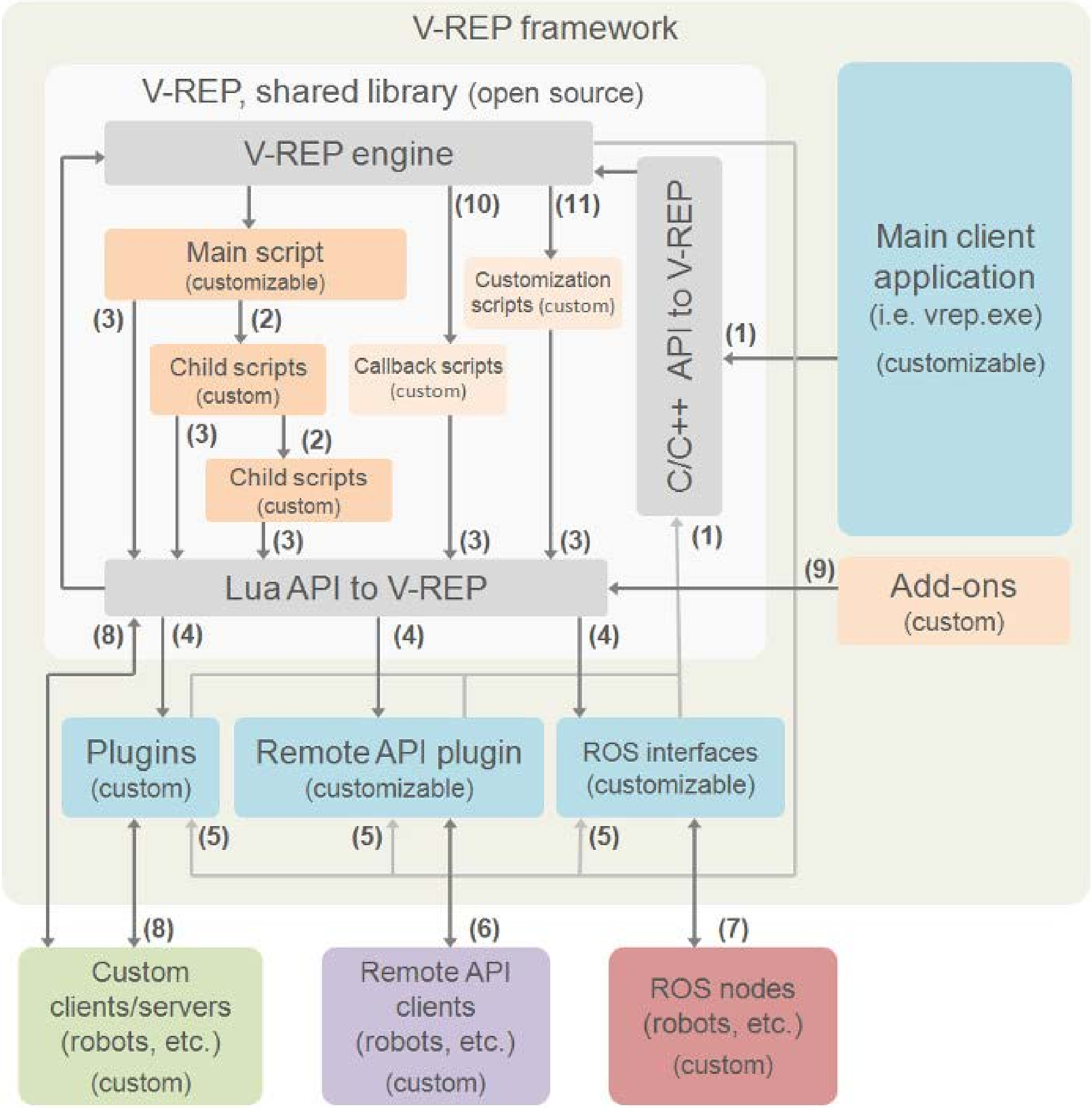
Supports all standard messages and is extendable

Generic (e.g. easy to change program from communicating with the simulation to communicating with a real robot)

I successfully compiled it with ROS Kinetic in ubuntu 16.04



Control Mechanisms



1) C/C++ API Calls
2) Cascaded Child script execution
3) LUA API calls
4) Custom LUA API callbacks
5) V-REP event callbacks
6) remote API function calls
7) ROS transit
8) custom communication
   a) socket, serial, pipes, etc.
9) Add-on calls to LUA API
10) Script callback calls

Control Mechanisms

	Embedded script	Add-on	Plugin	Remote API client	ROS node	Custom client/server
Control entity is external (i.e. can be located on a robot, different machine, etc.)	No	No	No	Yes	Yes	Yes
Difficulty to implement	Easiest	Easiest	Relatively easy	Easy	Relatively difficult	Relatively difficult
Supported programming language	Lua	Lua	C/C++	C/C++, Python, Java, Matlab, Octave, Lua, Urbi	Any [1]	Any
Simulator functionality access (available API functions)	500+ functions, extendable	500+ functions, extendable	500+ functions	>100 functions, extendable	Depends on the selected ROS interface	custom implementation
The control entity can control the simulation and simulation objects (models, robots, etc.)	Yes	Yes	Yes	Yes	Yes	Yes
The control entity can start, stop, pause and step a simulation	Start, stop, pause	Start, stop, pause	Start, stop, pause, step	Start, stop, pause, step	Start, stop, pause, step	Start, stop, pause, step
The control entity can customize the simulator	Yes	Yes	Yes	No	No	No
Code execution speed	Relativ. slow [2] (fast with JiT compiler)	Relativ. slow [2] (fast with JiT compiler)	Fast	Depends on programming language	Depends on programming language	Depends on programming language
Communication lag	None	None	None	Yes, reduced [3]	Yes, reduced	Yes, can be reduced
Control entity is fully contained in a scene or model, and is highly portable	Yes	No	No	No	No	No
API mechanism	Regular API	Regular API	Regular API	Remote API	ROS	Custom communication + regular API
API can be extended	Yes, with custom Lua functions	Yes, with custom Lua functions	Yes, V-REP is open source	Yes, Remote API is open source	Yes, ROS plugin is open source	N/A
Control entity relies on	V-REP	V-REP	V-REP	Sockets + Remote API plugin	Sockets + ROS plugin + ROS framework	Custom communication + script/plugin
Synchronous operation [4]	Yes, inherent. No delays	Yes, inherent. No delays	Yes, inherent. No delays	Yes. Slower due to comm. Lag	Yes. Slower due to comm. Lag	Yes. Slower due to comm. Lag
Asynchronous operation [4]	Yes, via threaded scripts	No	No (threads available, but API access forbidden)	Yes, default operation mode	Yes, default operation mode	Yes

1) Depends on what ROS currently supports

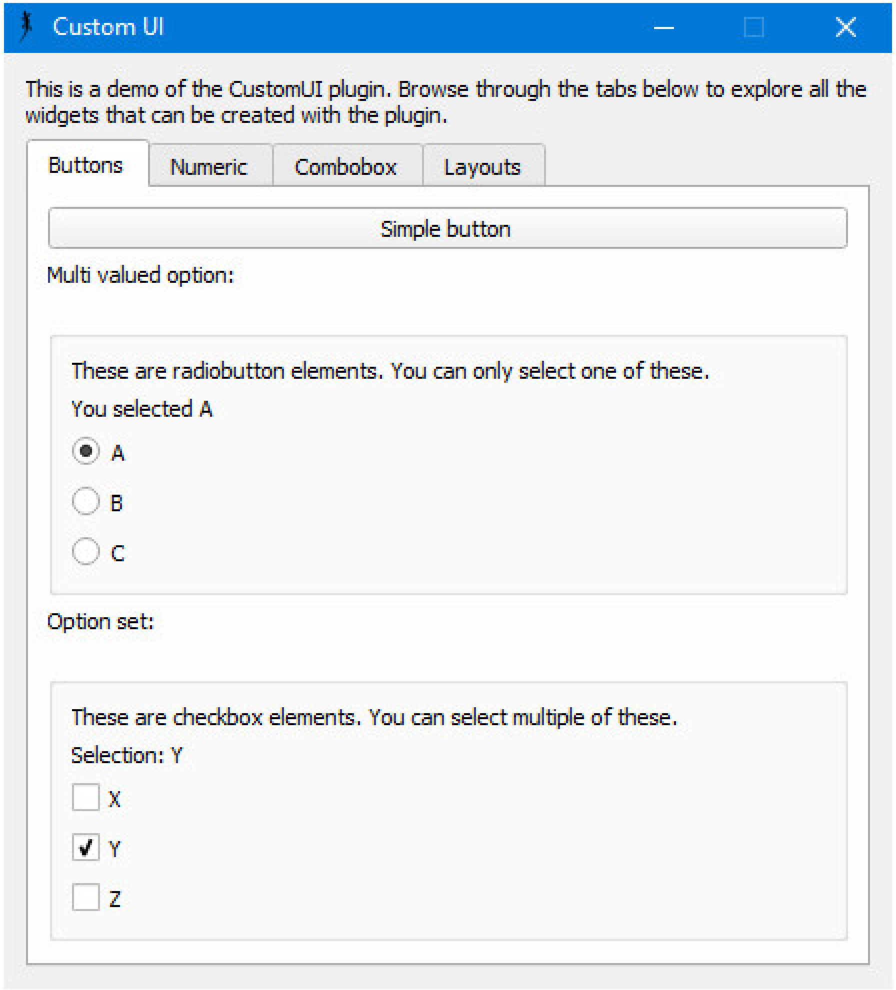
2) The execution of API functions is however very fast. Additionally, there is an optional JiT (Just in Time) compiler option that can be activated

3) Lag reduced via streaming and data partitioning modes

4) *Synchronous* in the sense that each simulation pass runs synchronously with the control entity, i.e. simulation step by step

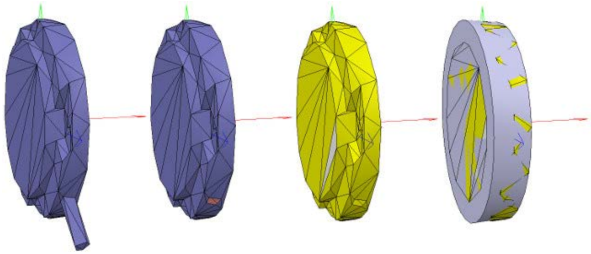
If in doubt, use their website!
http://www.coppeliarobotics.com

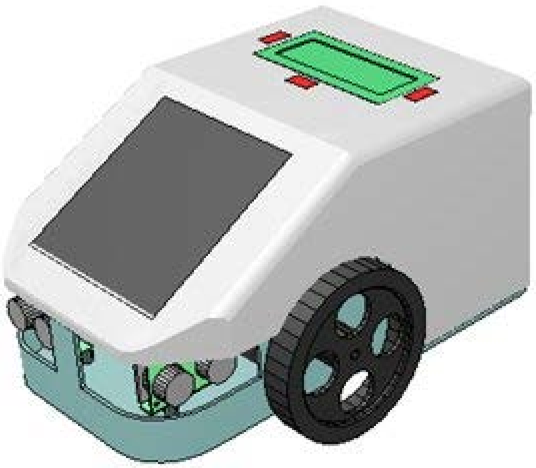
Other features - *Custom User Interfaces (QT-Based)*

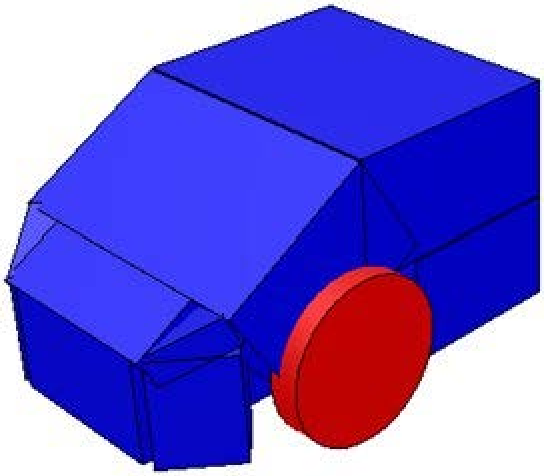


Other features - *Mesh Edit Modes*

- Triangle, vertex or edge edit mode
- Modify meshes (adjust vertices, add/remove triangles)
- Semi-automatic primitive shape extraction function
- Triangle, vertex or edge extraction
- Mesh decomposition
- Convex decomposition
- Convex hull extraction
- Mesh decimation



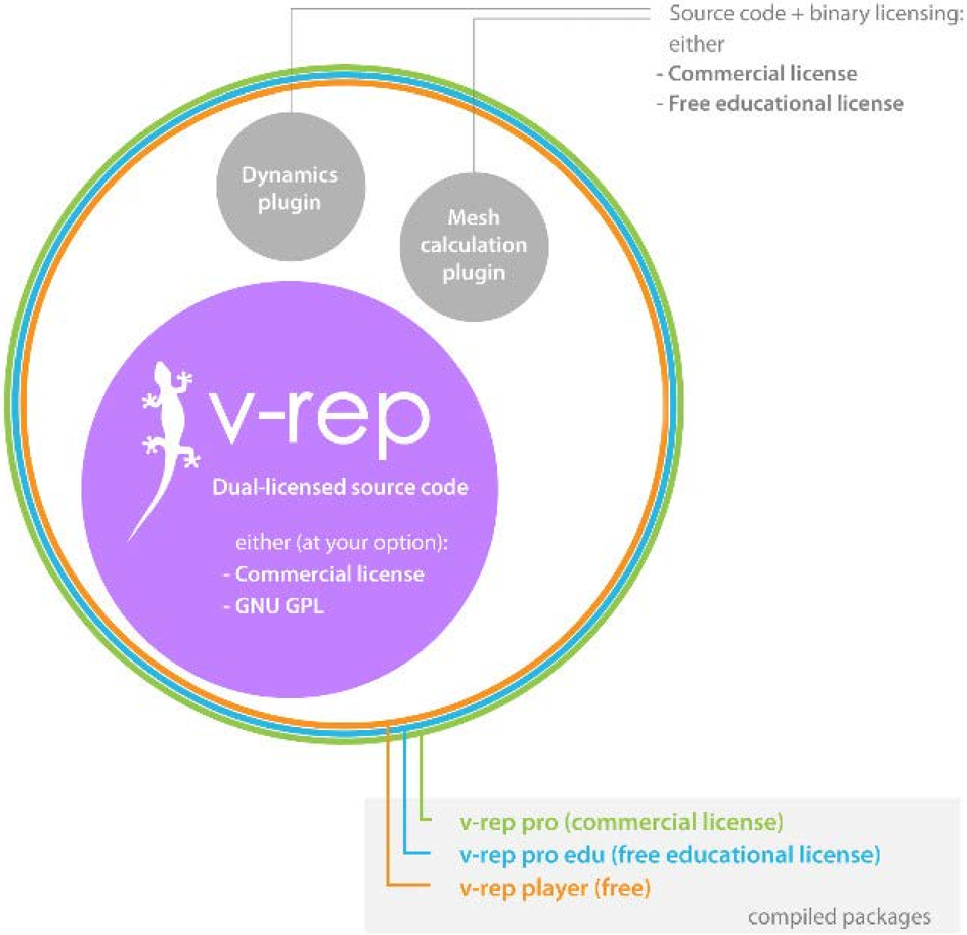




Other features

- Headless mode support (i.e. via command line)
- Import formats: OBJ, STL, 3DS, DXF, COLLADA & URDF
- Model browser and scene hierarchy
- Multilevel undo / redo
- Movie recorder *(w. ray tracing)*
- Simulation of paint or welding seams
- Static & dynamic textures
- Exhaustive documentation
- Etc.

V-REP Source Code Licensing



PLUGIN educational license
(where 'PLUGIN' may refer to 'DYNAMICS PLUGIN' or 'MESH CALCULATION PLUGIN'):

The PLUGIN educational license applies ONLY to EDUCATIONAL ENTITIES composed by following people and institutions:

1. Hobbyists, students, teachers and professors
2. Schools and universities

EDUCATIONAL ENTITIES do NOT include companies, research institutions, non-profit organisations, foundations, etc.

An EDUCATIONAL ENTITY may use, modify, compile and distribute the modified/unmodified PLUGIN under following conditions:

1. Distribution should be free of charge.
2. Distribution should be to EDUCATIONAL ENTITIES only.
3. Usage should be non-commercial.
4. Altered source versions must be plainly marked as such and distributed along with any compiled code.
5. When using the PLUGIN in conjunction with V-REP, the "EDU" watermark in the V-REP scene view should not be removed.
6. The origin of the PLUGIN must not be misrepresented. you must not claim that you wrote the original software.

The PLUGIN is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. In no event will the original author be held liable for any damages arising from the use of this software.

Resources

V-REP website: www.coppeliarobotics.com

V-REP user manual: www.coppeliarobotics.com/helpFiles/

V-REP forum: www.forum.coppeliarobotics.com

V-REP YouTube channel: VirtualRobotPlatform

V-REP Twitter account: coppeliaRobotic

Starting your own project in GoRobots

Starting your own project in GoRobots

1. Ask for membership in ENS group on GitLab (Koh, Jan-Matthias, or Me)

Starting your own project in GoRobots

1. Ask for membership in ENS group on GitLab (Koh, Jan-Matthias, or Me)
2. Fork your version of GoRobots

If you forked later than 08.47 this morning do this!

1) Configure a remote that points to the upstream repository in Git

    $ git remote add upstream git@gitlab.com:ens_sdu/gorobots.git

2) Fetch the branches and their respective commits from the upstream repository

    $ git fetch upstream

3) Check out your fork's local master branch.

    $ git checkout master

4) Merge the changes from upstream/master into your local master branch (Sync.)

    $ git merge upstream/master

5) Check for merge conflicts with (you may need to install meld)

    $ meld .

No Important Changes (Check with meld)?→ $ git reset --hard

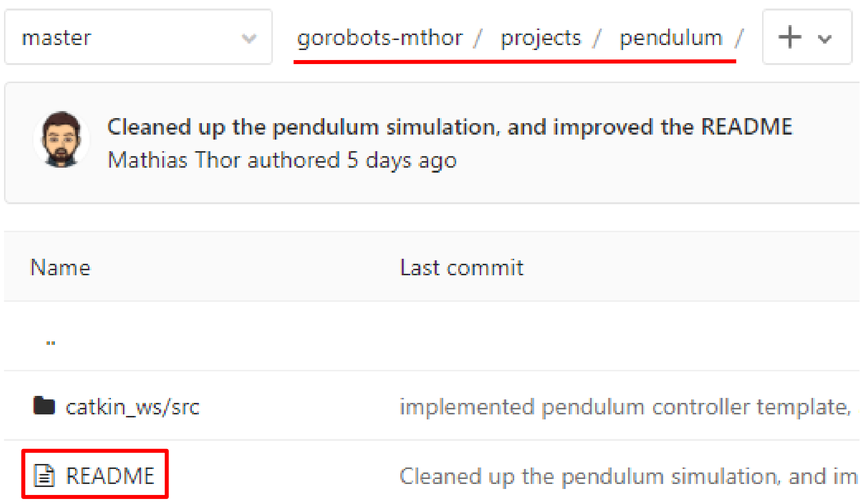
Starting your own project in GoRobots

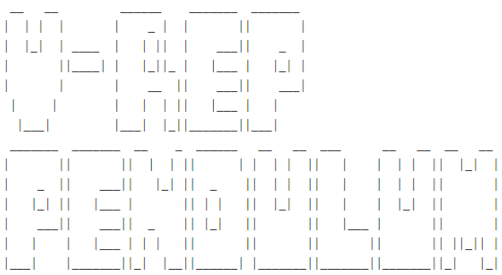
1. Ask for membership in ENS group on GitLab (Koh, Jan-Matthias, or Me)
2. Fork your version of GoRobots
3. Clone your fork to your Pc

git clone "this address"

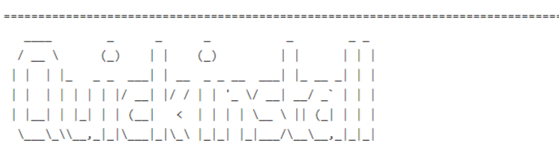
Starting your own project in GoRobots

1. Ask for membership in ENS group on GitLab (Koh, Jan-Matthias, or Me)
2. Fork your version of GoRobots
3. Clone your fork to your Pc
4. Get pendulum simulation up and running by following the guide in the readme file





By Mathias Thor



```
(tested on ubuntu 16.04 LTS)
================================================================================

 - Run following commands (install ros kinentic and more):
   $sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros
   $sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01
   $sudo apt-get update
   $sudo apt-get install -y git ros-kinetic-desktop-full git cmake python-tempita python-catkin-tools python-lxml x

 - Download: V-REP_PRO_EDU_V3_5_0_Linux.tar.gz from http://www.coppeliarobotics.com/downloads.html

 - Extract it in you home folder (e.g. /home/{username}/)

 - Run the following command in the V-REP root folder to launch the program (i.e. /home/mat/V-REP_PRO_EDU_V3_5_0_Li
   $./vrep.sh

 - Go to the catkin folder:
   $cd /home/{username}/workspace/gorobots/projects/pendulum/catkin_ws

 - write the following commands
   $source /opt/ros/kinetic/setup.bash
   $export VREP_ROOT="/home/{username}/V-REP_PRO_EDU_V3_5_0_Linux/"
   $export VREP_DIR="/home/{username}/V-REP_PRO_EDU_V3_5_0_Linux/"
   $catkin clean
   $catkin build
   $catkin build

 - Add the following lines to your .bashrc file in your home folder
   export VREP_ROOT="/home/{username}/V-REP_PRO_EDU_V3_5_0_Linux/"
   export VREP_DIR="/home/{username}/V-REP_PRO_EDU_V3_5_0_Linux/"
   export VREP_ROOT_DIR="/home/{username}/V-REP_PRO_EDU_V3_5_0_Linux/"
```

Please write me on mathias@mmmi.sdu.dk if you find any errors in the guide

Starting your own project in GoRobots

1. Ask for membership in ENS group on GitLab (Koh, Jan-Matthias, or Me)
2. Fork your version of GoRobots
3. Clone your fork to your Pc
4. Get pendulum simulation up and running by following the guide in the readme file
5. Start your own project and use the following structure

Your controller here →
e.g. /controllers/pendulum/pendulumController.cpp

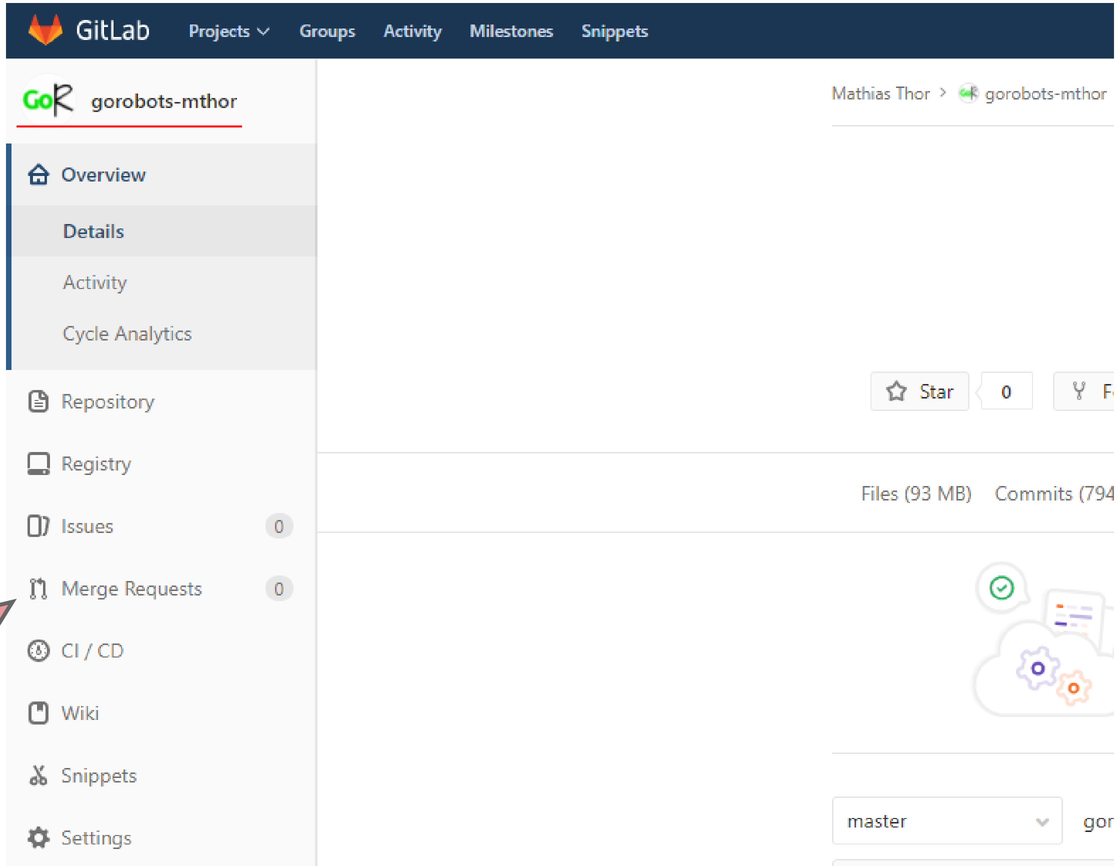
Your makelistcatkin and child script here →
e.g. /projects/pendulum/catkin_ws/src/...

Your v-rep sim (.ttt) in v-rep_simulations in here →
e.g. /utils/v-rep_simulations/pendulum/ pendulum.ttt

Name
archive
controllers
docs
examples
practices
projects
tests
utils
.gitignore
Makefile
README
simulation.makefile

Starting your own project in GoRobots

1. Ask for membership in ENS group on GitLab (Koh, Jan-Matthias, or Me)
2. Fork your version of GoRobots
3. Clone your fork to your Pc
4. Get pendulum simulation up and running by following the guide in the readme file
5. Start your own project and use the following structure
6. End your project with a merge request



Press this

Questions and Getting started

Any Question?

Try getting the pendulum simulation to run on your PC!

Installing VORTEX Physical engine

1) Go to: https://www.cm-labs.com/licenses/
2) Create free account
3) Go to: My Account > Downloads (the download page)
4) Press: Optional Download > Vortex Studio (Linux)
5) Then go to: My Account > Licenses
    a) Then request a license for Vortex Studio Essentials
6) Extract the downloaded folder
    a) Go to "Vortex_Studio…/bin" (in the terminal)
    b) Then run "./VortexLicenseManager --activate *KEY*"
7) Restart V-REP


6 programming approaches: embedded scripts, plugins
add-ons, remote API clients, ROS nodes & custom solutions