

Virtual Robot Experimentation Platform

V-REP

www.coppeliarobotics.com

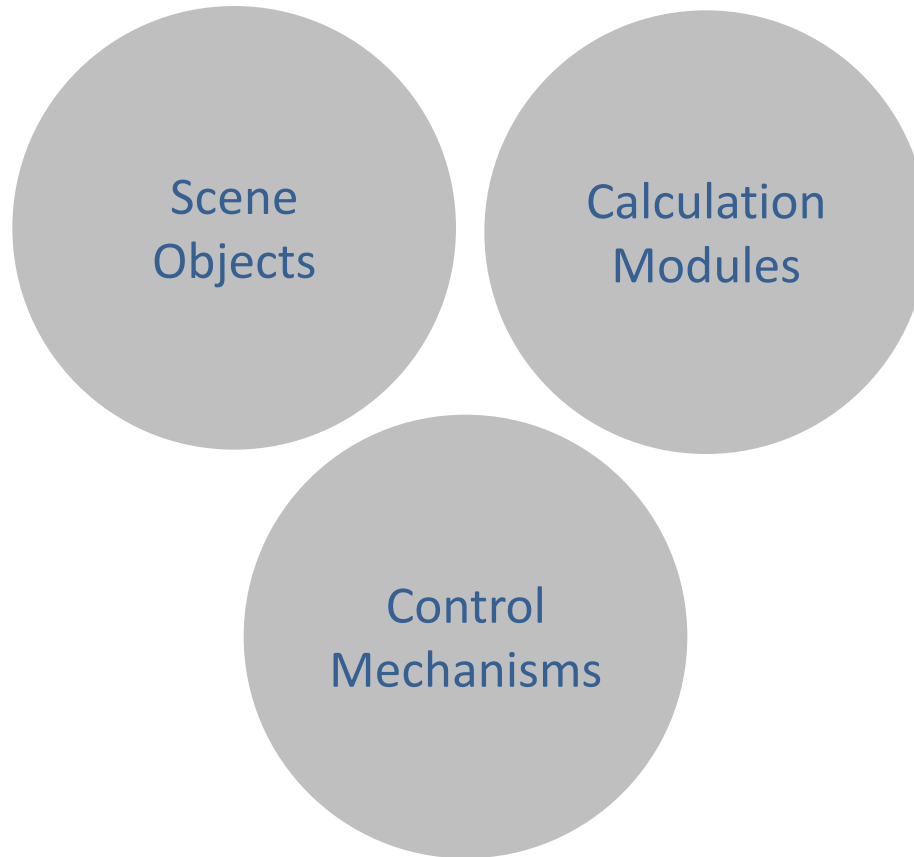
What is it ? General purpose robot simulator with integrated development environment

What can it do ? Sensors, mechanisms, robots and whole systems can be modelled and simulated in various ways >> [Play overview video](#)

Typical applications ?

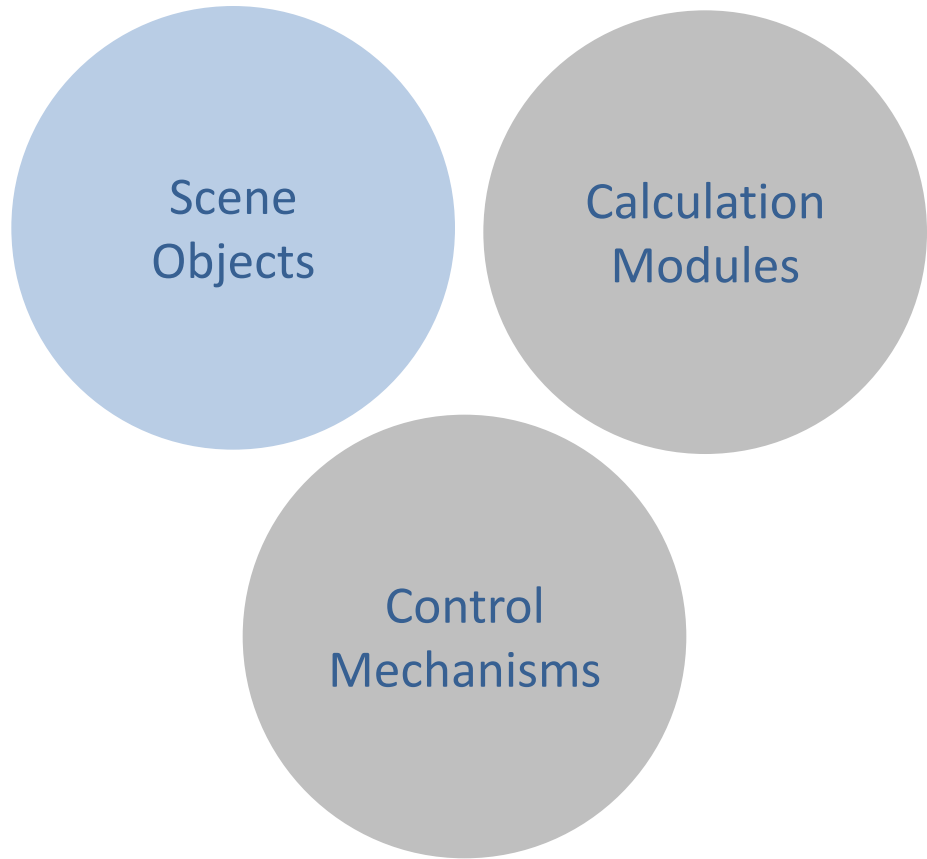
- Fast prototyping and verification
- Fast algorithm development
- Robotics related education
- Remote monitoring
- Hardware control
- Simulation of factory automation systems
- Safety monitoring
- Product presentation
- etc.

3 Central Elements

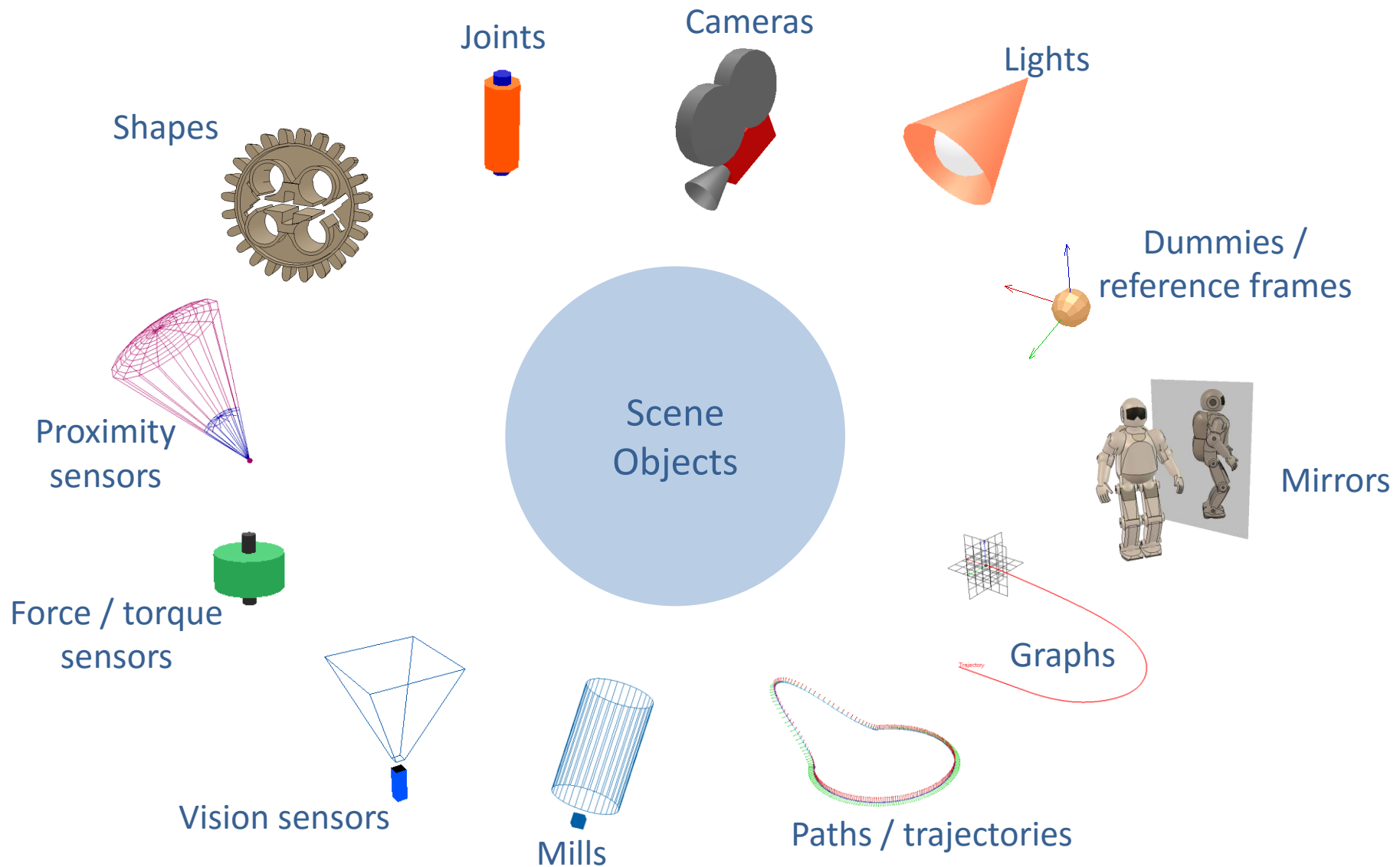


Scene Objects

- Basic building blocks
- 12 different types
- Can be combined with each other
- Can form complex systems together with calculation modules and control mechanisms

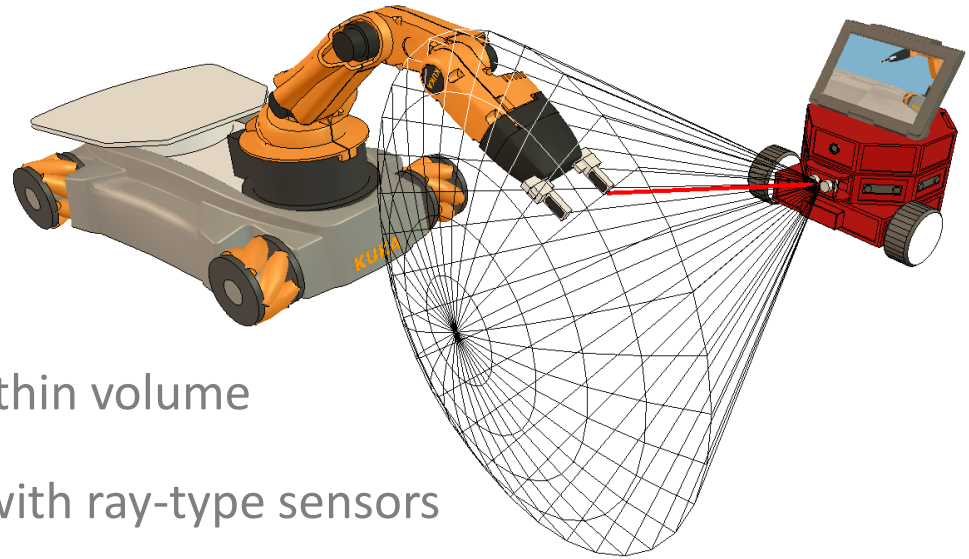


Scene Objects



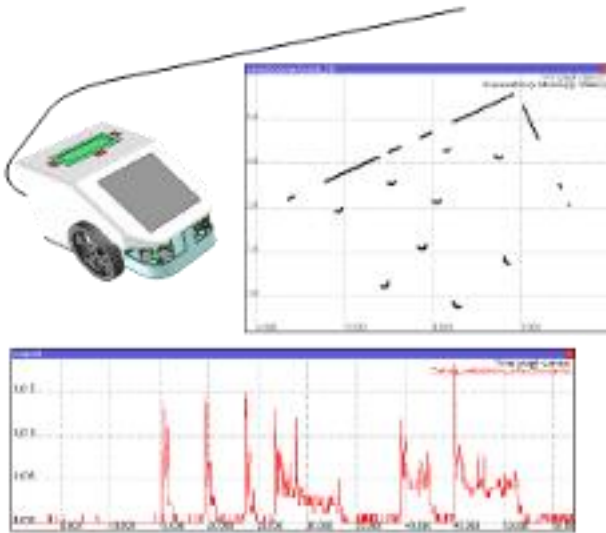
Proximity Sensors

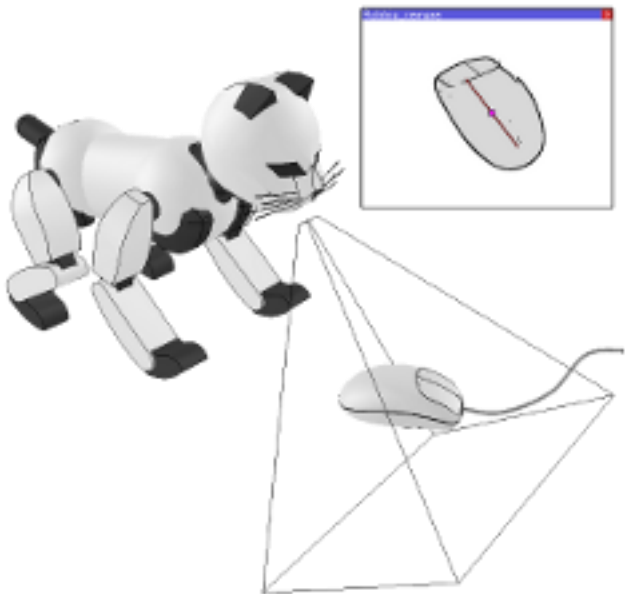
- More than simple ray-type detection
- Configurable detection volume
- Fast minimum distance calculation within volume
- Much more realistic simulation than with ray-type sensors



Graphs

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





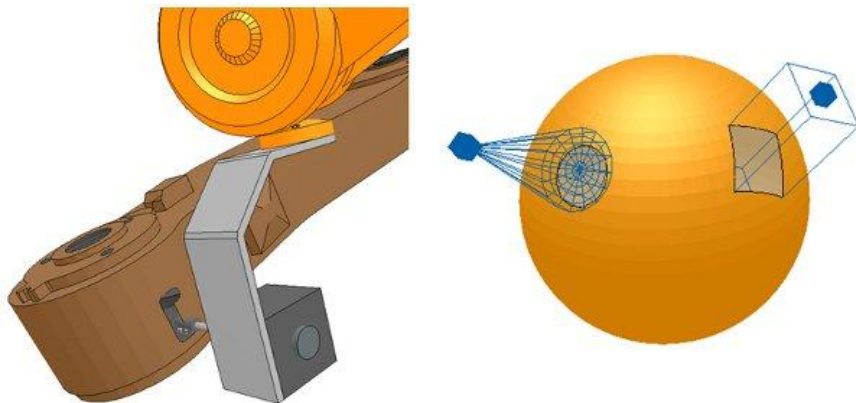
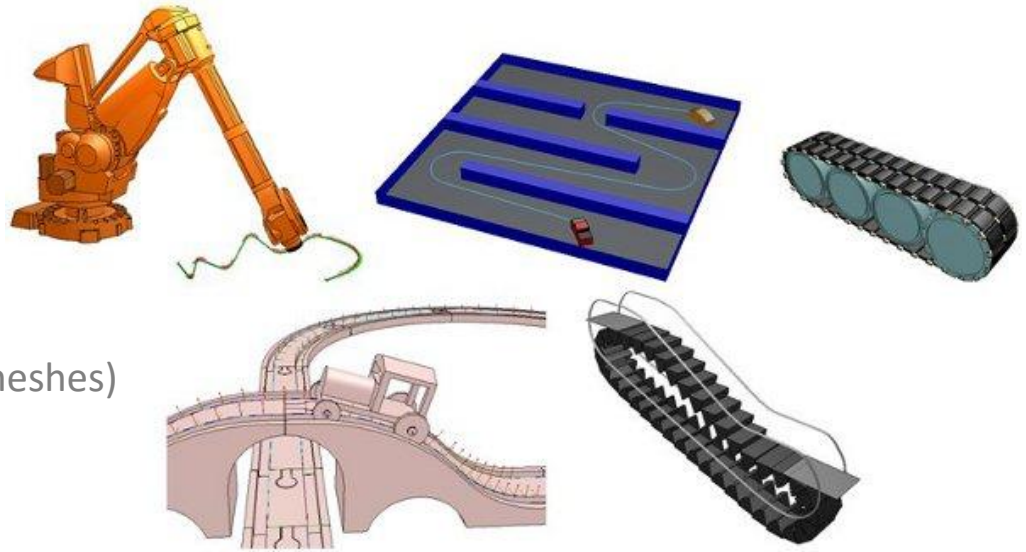
Vision Sensors

- Integrated image processing
- Extendable via plugin mechanism
- Ray-traced rendering also available

Paths and Mills

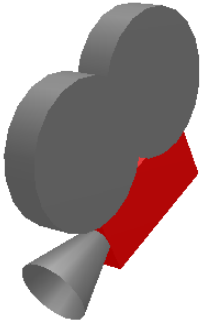
Paths

- 6 dim. trajectory definition
- Path can be shaped
(i.e. automatically generate extruded meshes)



Mills

- Customizable cutting volume
- Cuts shapes (i.e. meshes)



Cameras

- Perspective / orthographic projection
- Tracking & automatic view-fitting function

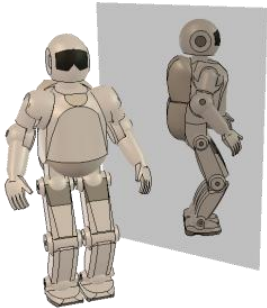
Lights

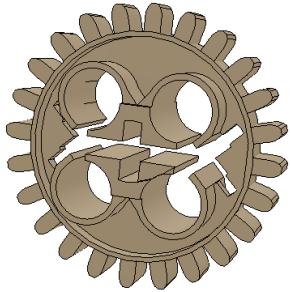
Spotlight / directional / omnidirectional



Mirrors

Mirror or scene / object clipping function





Shapes

- Random mesh, convex mesh, primitive mesh or heightfield mesh
- Can be grouped/ungrouped
- Optimized for fast calculations



Force/Torque Sensors

- Measures force and torque
- Can conditionally break apart

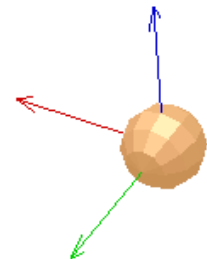
Joints

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

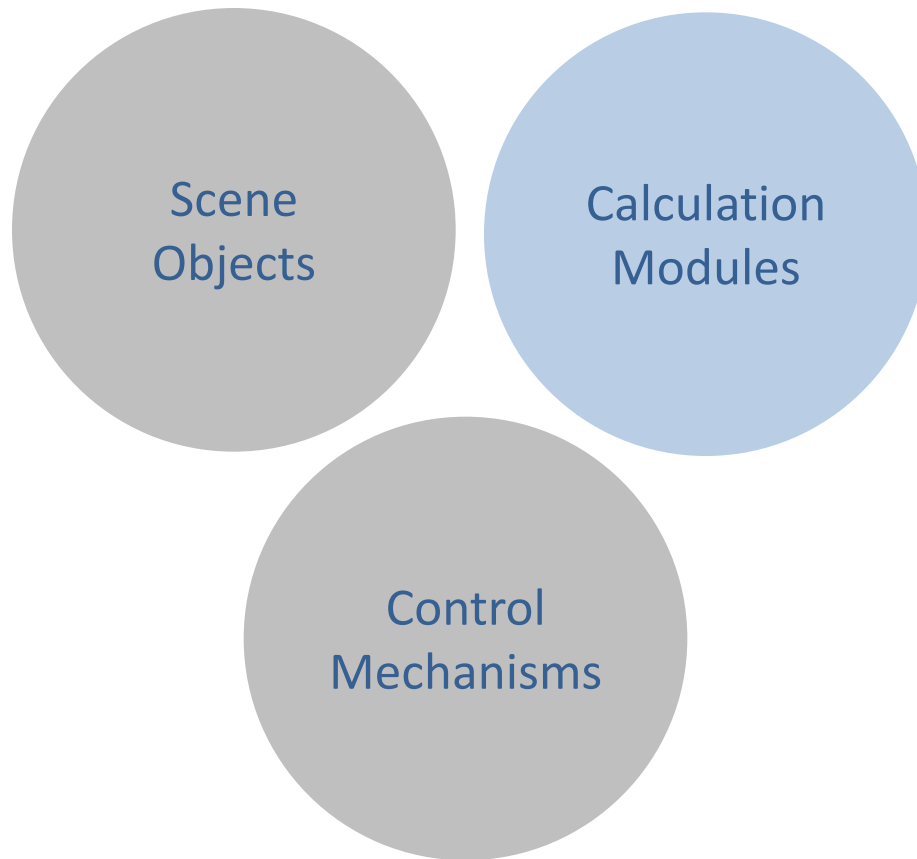


Dummies

Auxiliary reference frame & helper object



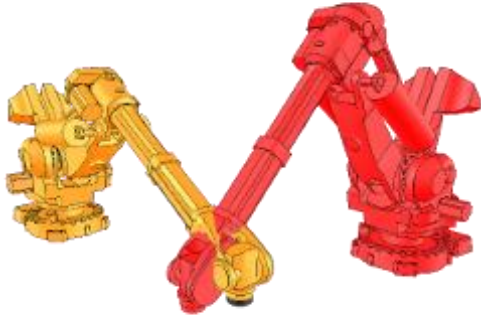
3 Central Elements



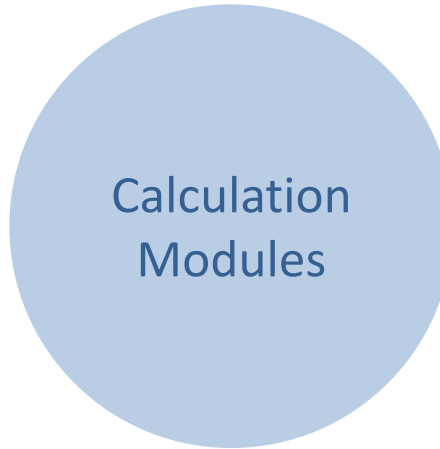
Calculation modules

- 5 basic algorithms
- Can be combined with each other
- Can form complex systems together with scene objects and control mechanisms

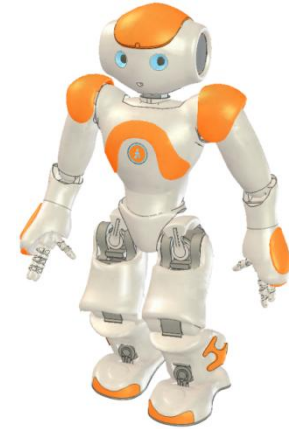
Calculation Modules



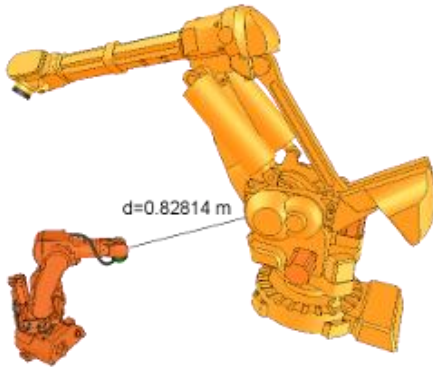
Collision detection



Calculation
Modules



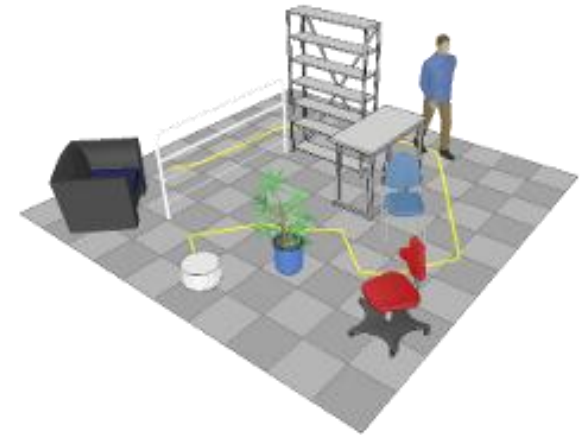
Physics / Dynamics



Minimum distance calculation



Forward / Inverse kinematics



Path / motion planning

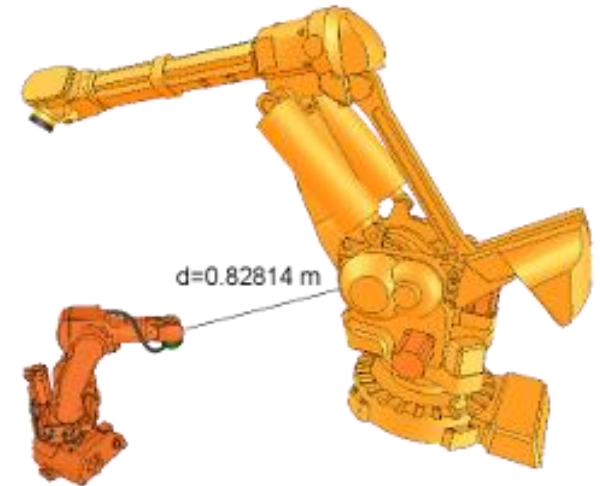
Inverse / forward Kinematics

- Any mechanism: redundant, branched, closed, etc.
- Damped / undamped resolution
- Weighted resolution
- Conditional resolution
- Obstacle avoidance



Minimum Distance Calculation

Any mesh (also open / concave / polygon soups)



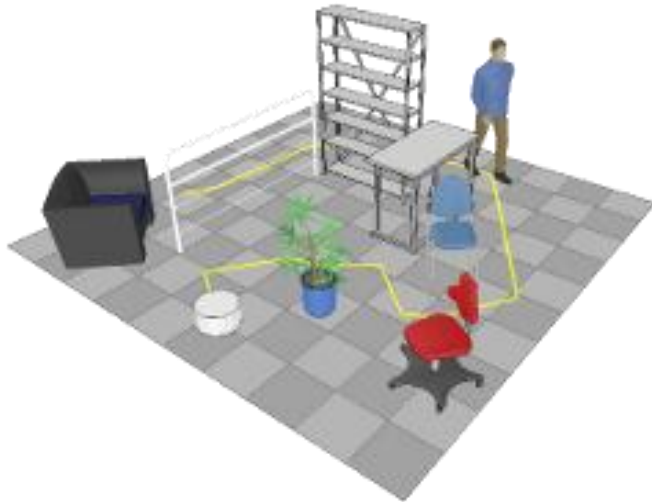
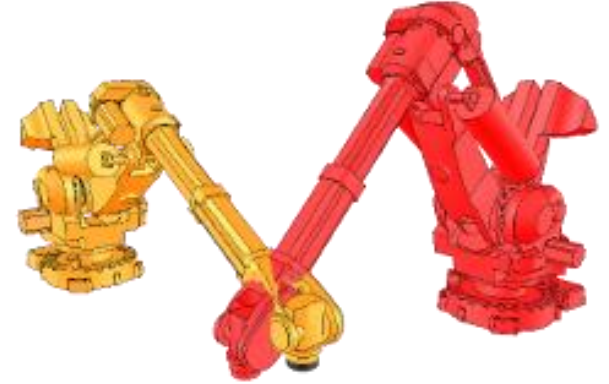


Dynamics / Physics

- 4 physics engines: [Bullet Physics](#)
[Open Dynamics Engine](#)
[Vortex Dynamics](#)
[Newton Dynamics](#)
- Simple mouse click to switch
- Dynamic particles to simulate air or water jets
- Can work hand-in-hand with kinematics module

Collision Detection

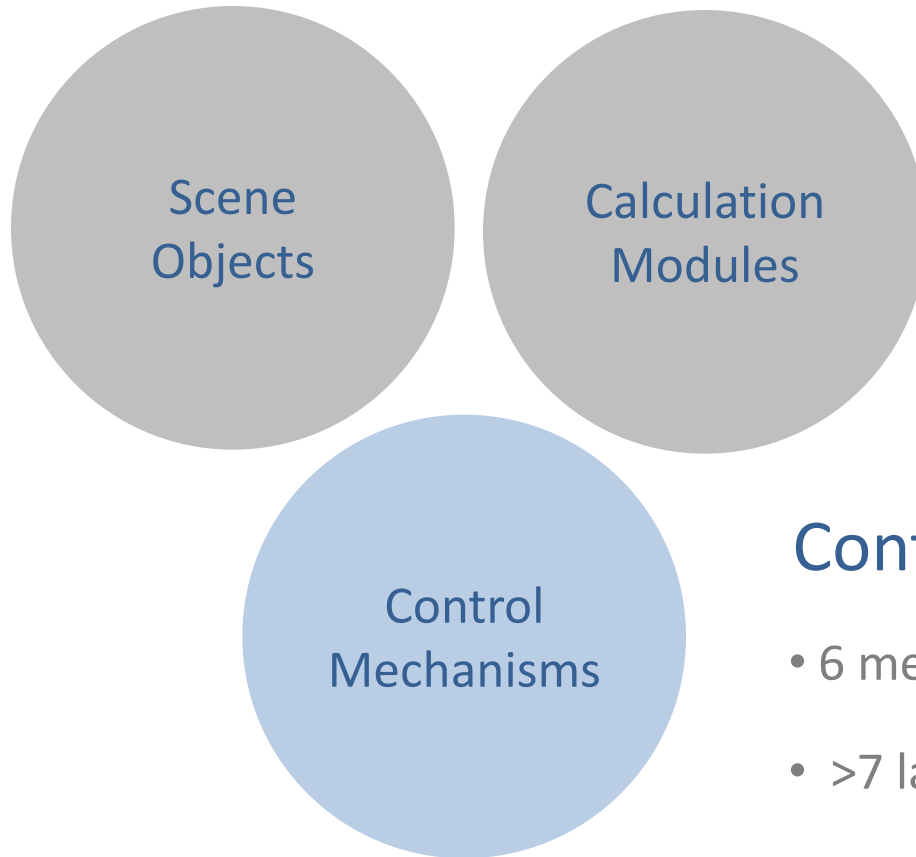
Any mesh (also open / concave / polygon soups)



Path / Motion Planning

Supported via an OMPL plugin for V-REP

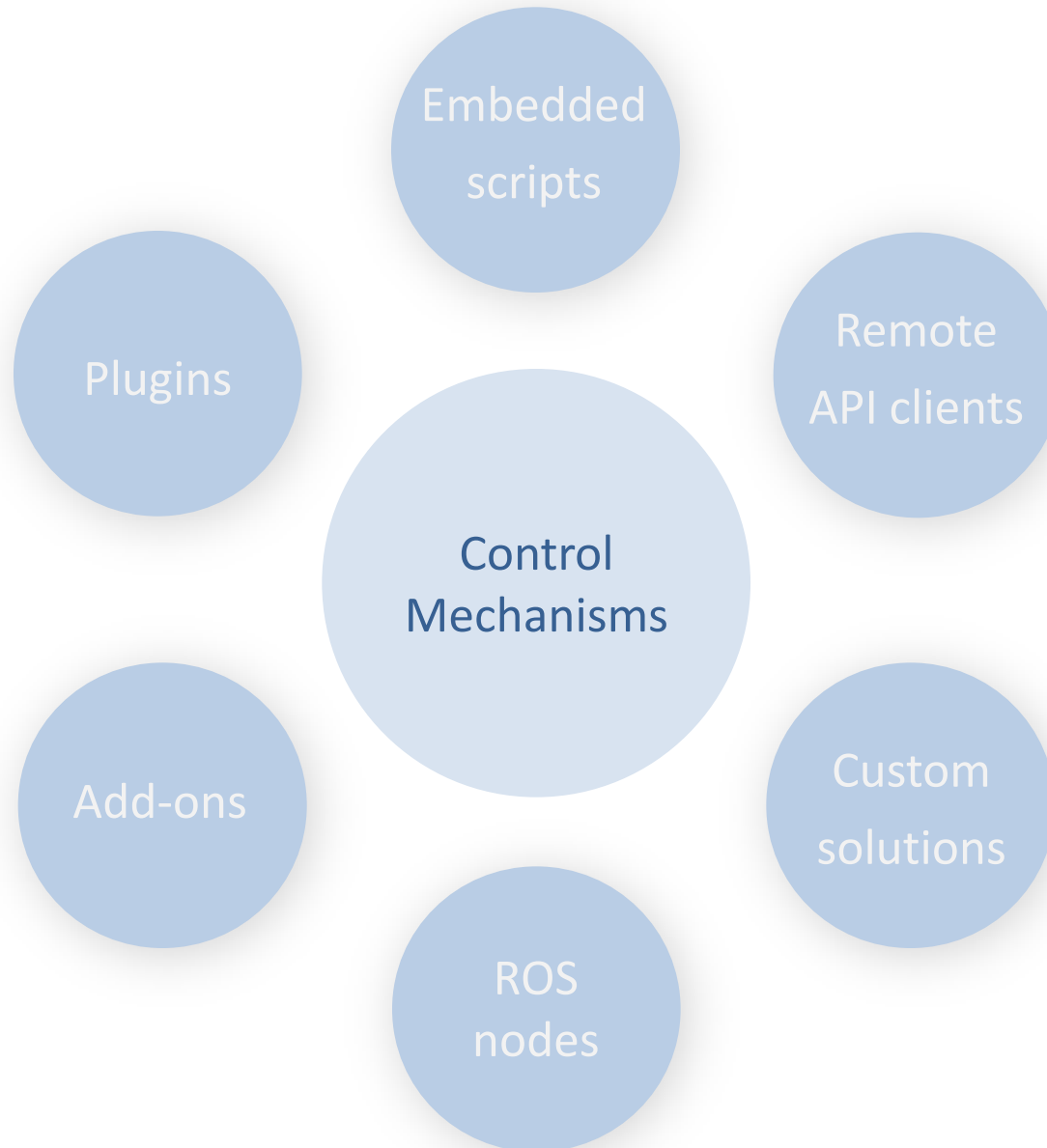
3 Central Elements



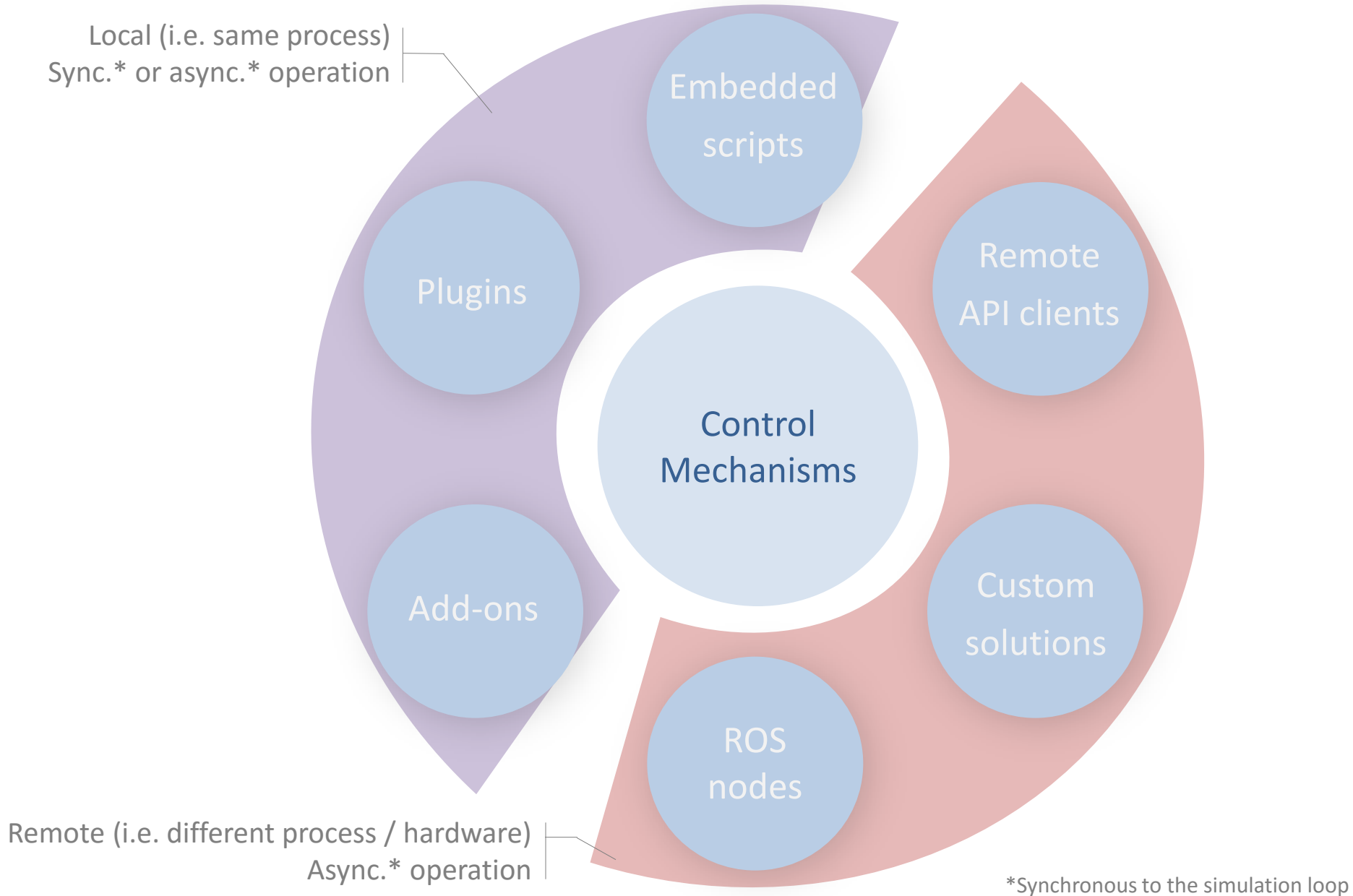
Control Mechanisms

- 6 methods or interfaces
- >7 languages
- 6 methods can be used at the same time, and even work hand-in-hand

Control Mechanisms



Local and Remote Interfaces



Plugins

Plugins

- > 400 API functions. Extendable
- C/C++ interface
- Can customize the simulator
- Can register new embedded script commands

Embedded scripts

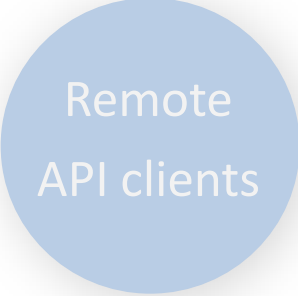
Embedded Scripts

- > 300 API functions. Extendable
- Can be attached to any scene object
- Many Lua extension libraries available
- Threaded or non-threaded. Threads can be synchronized easily
- Various types: main script, child scripts, callback scripts (e.g. custom joint controllers)
- Lua interface
- Lightweight and easy to program
- Extremely portable solution

Add-ons

Add-ons

- > 300 API functions. Extendable
- Lua interface
- Can customize the simulator
- Lightweight and easy to set-up
- Many Lua extension libraries available



Remote API clients

Remote API

- > 100 API functions. Extendable
- C/C++, Python, Java, Matlab, Octave, Lua & Urbi interfaces
- Data streaming and partitioning modes
- Lightweight and easy to use



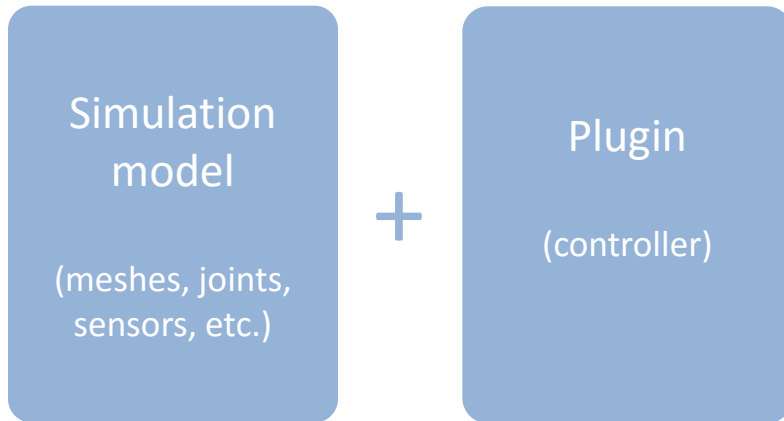
ROS nodes

ROS Interface

- > 100 services, >30 publisher types, >25 subscriber types. Extendable
- Publishers/subscribers can be enabled via service calls, or via an embedded script function call

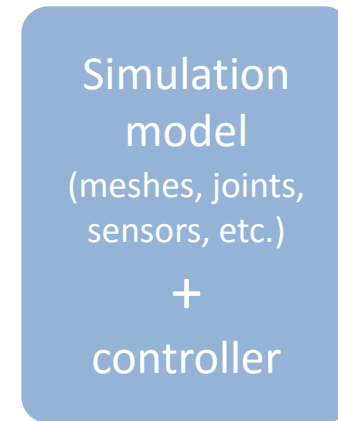
Controller Integration

Plugins



2 items

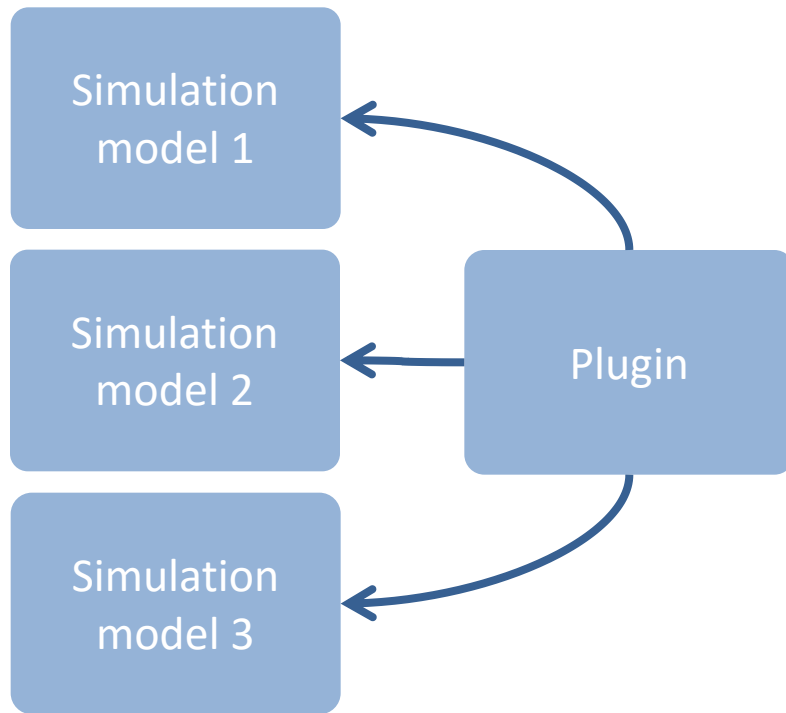
Embedded scripts



1 item

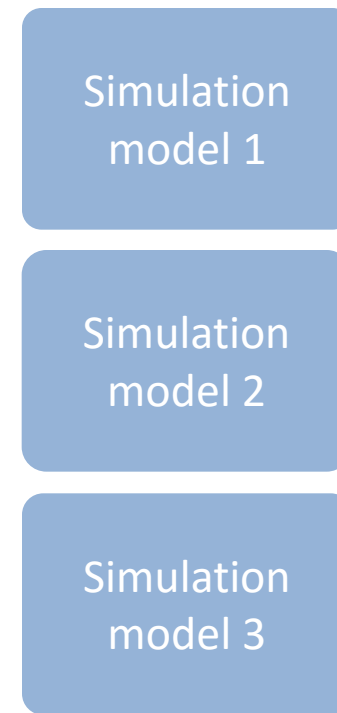
Scalability

Plugins



Plugin has to manage instances

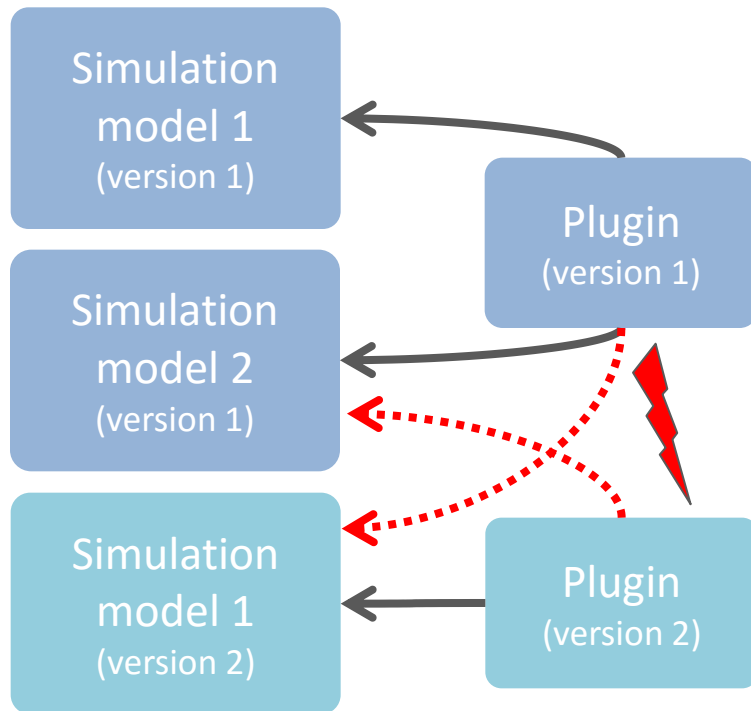
Embedded scripts



Scalability is inherent

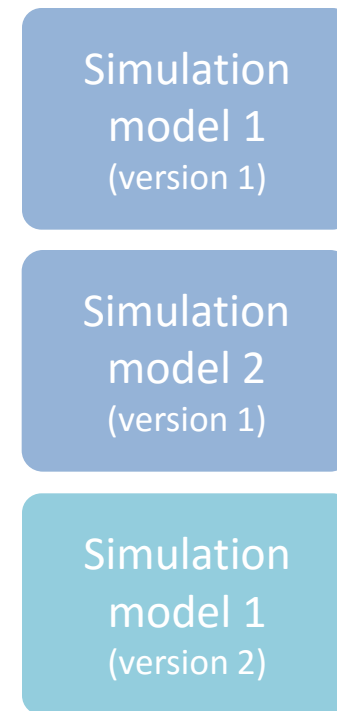
Version Conflicts

Plugins



High chances for conflicts

Embedded scripts



No chances for conflicts

Embedded Script Advantages 4/6

Portability

Plugins

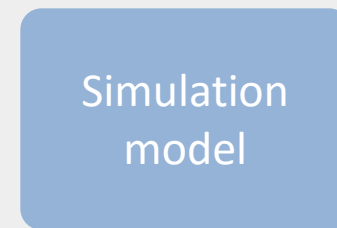
Embedded scripts

Same OS



2 files

Same OS



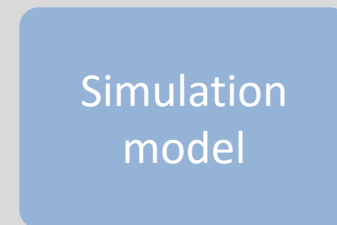
1 file

Different OS



Many files
Compilation required
OS-specific problems

Different OS



1 file
No compilation required
No OS-specific problems

Other Considerations

Plugins

Creation, compilation and installation difficulty:

High

Model modification difficulty:

High

Maintenance over the years:

OS-dependent

Compiler-dependent

Framework-dependant

Embedded scripts

Creation, compilation and installation difficulty:

Low

Model modification difficulty:

Low

Maintenance over the years:

OS-independent

Compiler-independent

Framework-independant

Synchronization with Simulation Loop

Plugins

Embedded scripts

Non-threaded

Control routine called at
each simulation pass

Easy

Non-threaded

Control routine called at
each simulation pass

Easy

Threaded

Complex synchronization
mechanism required

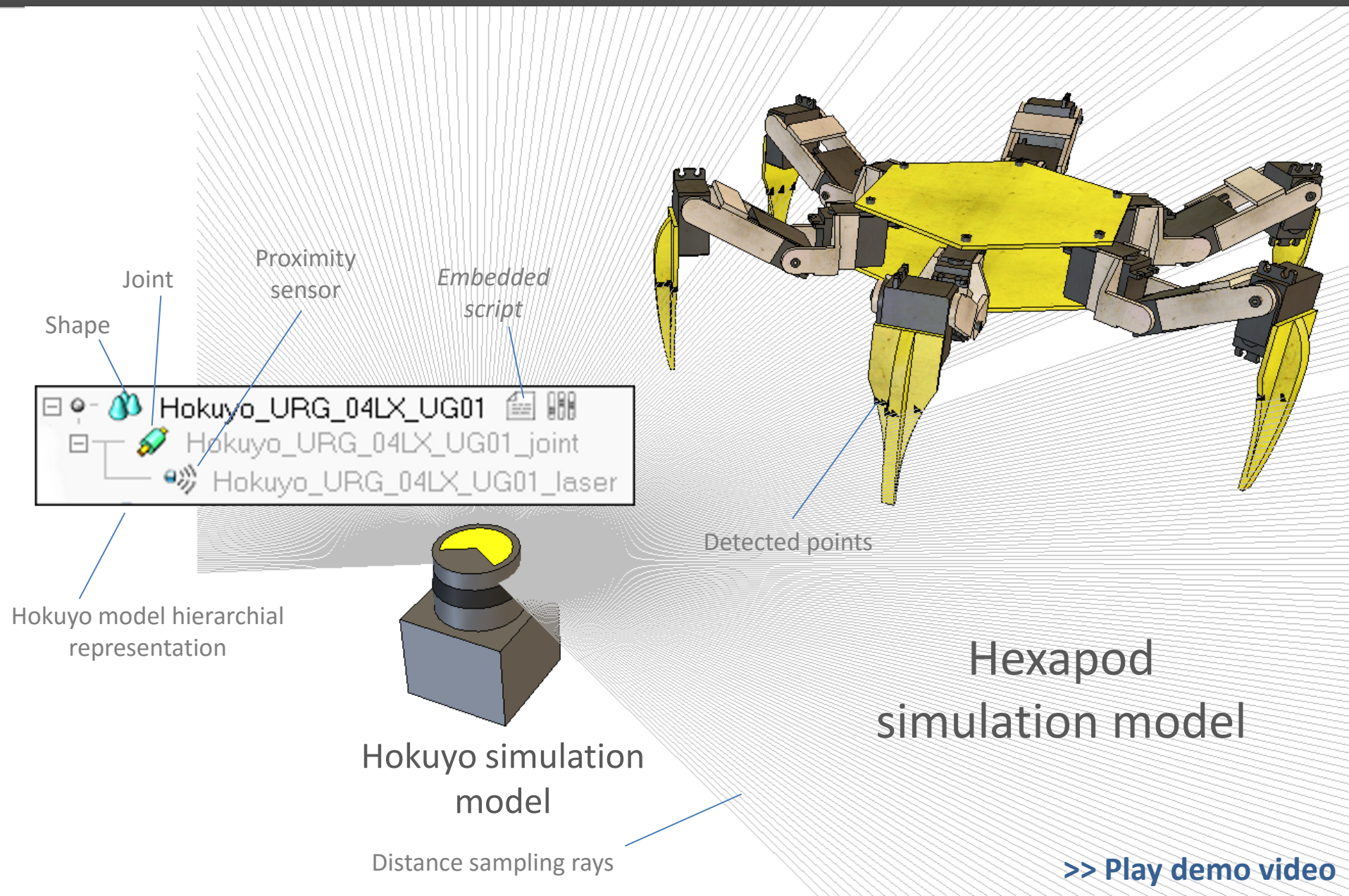
Difficult

Threaded

Control routine thread can
behave as a coroutine
e.g. `simSwitchThread()`
`simSetThreadSwitchTiming(delay)`
`simSetThreadIsFree(isFree)`
`simSetThreadResumeLocation(location,order)`

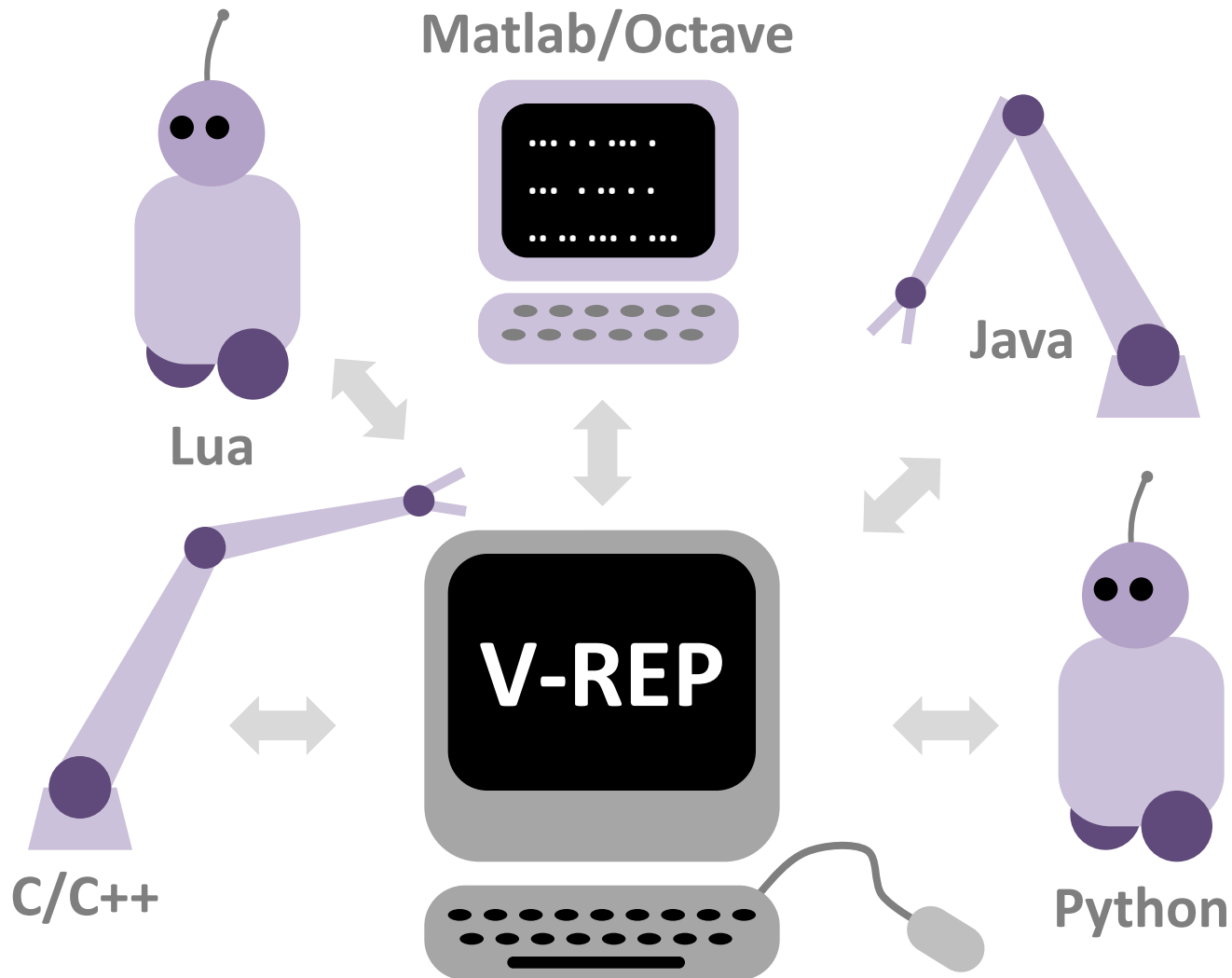
Easy

Embedded Scripts – Simple Example



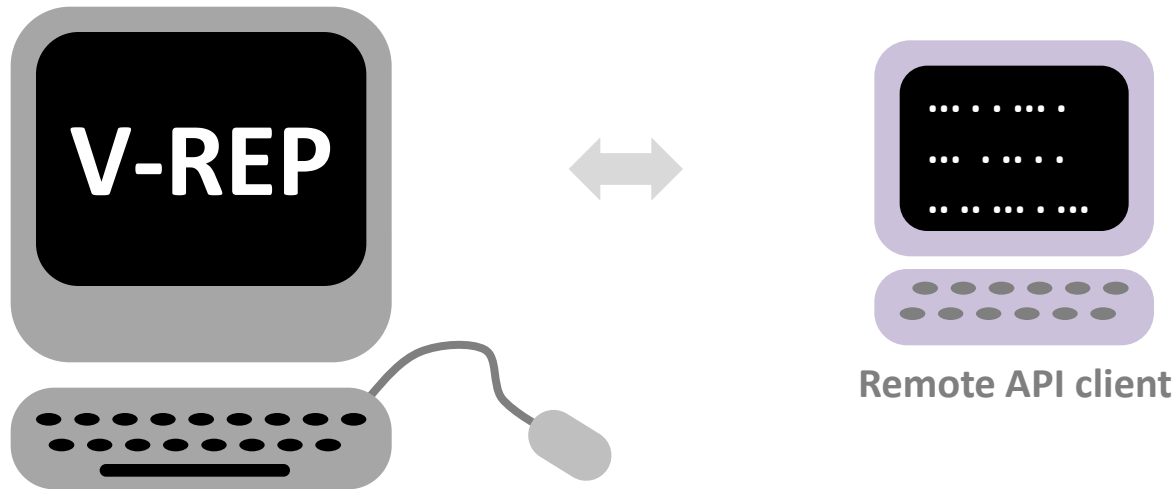
Remote API Advantages 1/2

Runs on any hardware, lightweight, several languages



Remote API Advantages 2/2

Very easy to use, almost like a *regular API*



Remote API function

Regular arguments

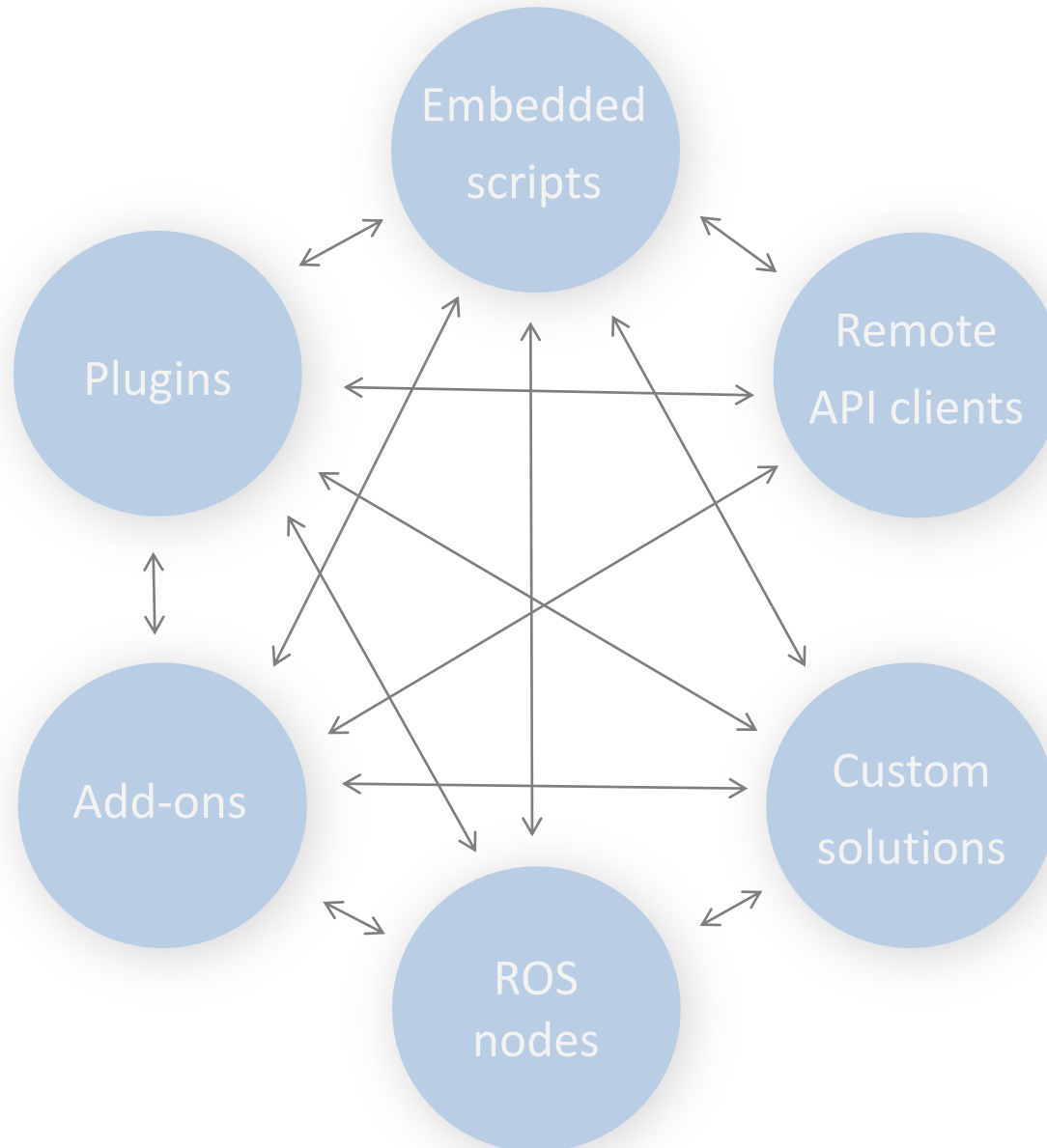
```
int returnCode=simxGetJointPosition(jointHandle,&position,operationMode);
```

- `simx_return_ok`
- `simx_return_timeout_flag`
- `simx_return_novalue_flag`
- `simx_return_remote_error_flag`
- `simx_return_local_error_flag`
- etc.

Blocking mode

- `simx_opmode_oneshot`
- `simx_opmode_oneshot_wait`
- `simx_opmode_streaming`
- `simx_opmode_discontinue`
- `simx_opmode_buffer`
- etc.

Collaborative Control Mechanisms



Example of Collaborative Mechanism 1 / 3

Remotely call a script function

```
float coords[3]={0.1f,0.2f,0.3f};
int retIntCnt;
int* retInts;
simxCallScriptFunction(clientID,"remoteApiCommandServer",sim_scripttype_childs
cript,"createDummy_function",0,NULL,3,coords,1,"MyDummyName",0,NULL,&retIntCnt
,&retInts,NULL,NULL,NULL,NULL,NULL,simx_opmode_blocking);
printf("Dummy handle: %i\n",retInts[0]);
```

Remote
API client



Creates a dummy, renames it and positions it according to the received parameters

```
createDummy_function=function(inInts,inFloats,inStrings,inBuffer)
-- Create a dummy object with specific name and coordinates
if #inStrings>=1 and #inFloats>=3 then
    local dummyHandle=simCreateDummy(0.05)
    local position={inInts[2],inInts[3],inInts[4]}
    simSetObjectName(dummyHandle,inStrings[1])
    simSetObjectPosition(dummyHandle,-1,inFloats)
    return {dummyHandle},{},{},' -- return the handle of the dummy
end
end
```

Embedded
script

Example of Collaborative Mechanism 2 / 3

Plugin

```
void SCRIPT_DO_SOME_MAGIC_CALLBACK(SScriptCallBack* callbackStruct)
{ // gets called when a script calls „simExt_doSomeMagic“
  int inputOutputArgumentStack=callbackStruct->stackID;
  ...
}

// Initialization phase of plugin: register new script commands:
simRegisterScriptCallbackFunction(SCRIPT_DO_SOME_MAGIC,"number value1,
    number value2=simExt_doSomeMagic(number index,table inVals)",
    SCRIPT_DO_SOME_MAGIC_CALLBACK);
```

Registers and handles the custom script API function „simExt_doSomeMagic“

Calls the custom API function „simExt_doSomeMagic“

```
returnData1,returnData2=simExt_doSomeMagic(arg1,arg2)
```

Embedded
script

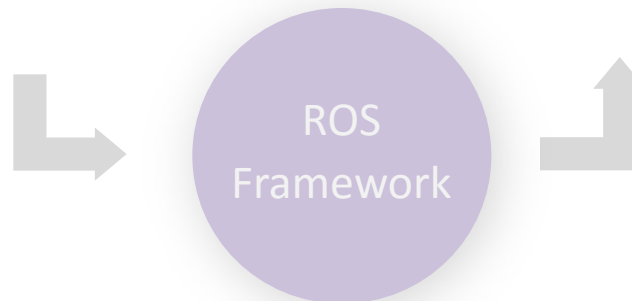
Example of Collaborative Mechanism 3 / 3

Enable image streaming to ROS

```
-- Following in the script initialization phase (executed just once):  
  
-- Retrieve the handle of the vision sensor we wish to stream:  
visionSensorHandle=simGetObjectHandle('Vision_sensor')  
  
-- Now enable topic publishing and streaming of the vision sensor's data:  
topicName=simExtROS_enablePublisher('visionSensorData',1,  
    simros_strmcmd_get_vision_sensor_image,visionSensorHandle,0,'')  
  
-- Retrieve the handle of the passive vision sensor. We will use  
-- the passive vision sensor to visualize received data:  
passiveSensorHandle=simGetObjectHandle('PassiveVision_sensor')  
  
-- Now enable a topic subscription:  
simExtROS_enableSubscriber(topicName,1,  
    simros_strmcmd_set_vision_sensor_image,passiveSensorHandle,0,'')
```

Embedded script

Enable image streaming from ROS



Control Mechanisms – Feature Overview

	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)	400+ functions, extendable	400+ functions, extendable	500+ functions	>100 functions, extendable	>100 services, >30 publisher types, >25 subscriber types, extendable	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

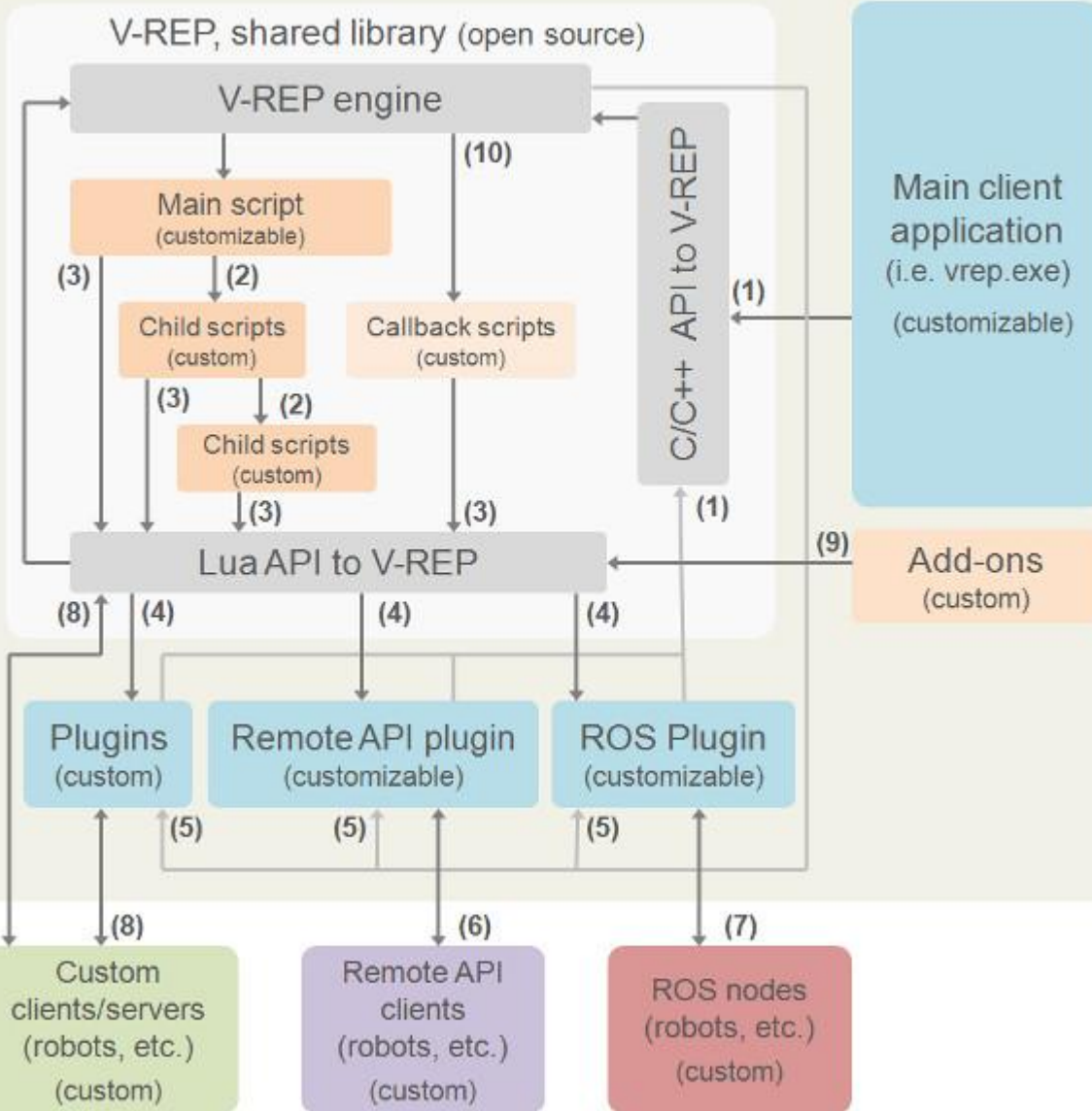
²) The execution of API functions is however very fast. Additionally, there is an optional JIT (Just in Time) compiler option that can be activated

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

Architecture Overview

V-REP framework



(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 (socket, serial, pipes, etc.)

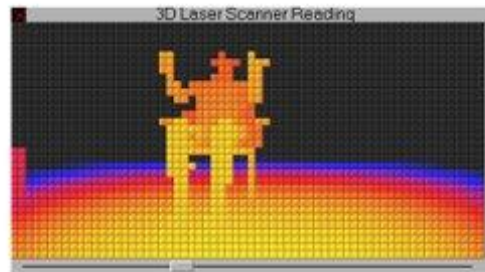
(9) Add-on calls to Lua API

(10) Script callback calls

Other Feature: Custom User Interfaces

Custom User Interfaces

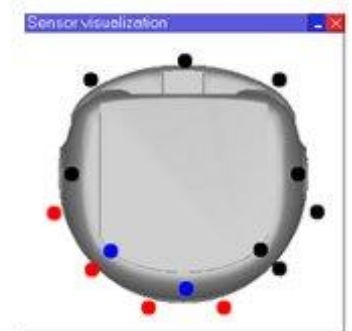
- Buttons, sliders, edit boxes and labels
- Can be textured or animated (e.g. via video stream)
- Can be attached (i.e. embedded) to scene objects
- Extremely portable
- Integrated edit mode



Joints	
All parameters to 0	
Store parameters	
Restore parameters	
Test all joints	
Test "test" joints	
Axis1	120.0
029.5	136.0
Axis2	119.3
013.0	023.2
Axis3	145.3
122.0	125.6
Axis4	078.2
097.2	063.4
Axis5	147.2
046.5	033.3
Axis6	119.9
135.5	086.4
Axis7	044.2
073.1	170.0
Axis8	069.1
000.0	017.0

20 x 4 LCD module

LCD line 1	12345678901234567890
LCD line 3	069.1
LCD line 4	017.0



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



- Headless mode support (i.e. via command line)
- Import formats: OBJ, STL, 3DS, DXF, COLLADA & URDF
- Integrated Reflexxes motion library: www.reflexxes.com
- Model browser and scene hierarchy
- Multilevel undo / redo
- Movie recorder
- Simulation of wireless communication
- Simulation of paint or welding seams
- Static & dynamic textures
- Exhaustive documentation
- Etc.

State-of-the-art distributed control architecture

- Embedded scripts
- Remote API
- ROS interface

Extremely fine-grained and large amount of features

- >400 different API function
- 12 types of simulation objects (force/torque sensor, joint, camera, etc.)
- Integrated physics, kinematics, collision/distance calculation & path planning

V-REP sets on several horses

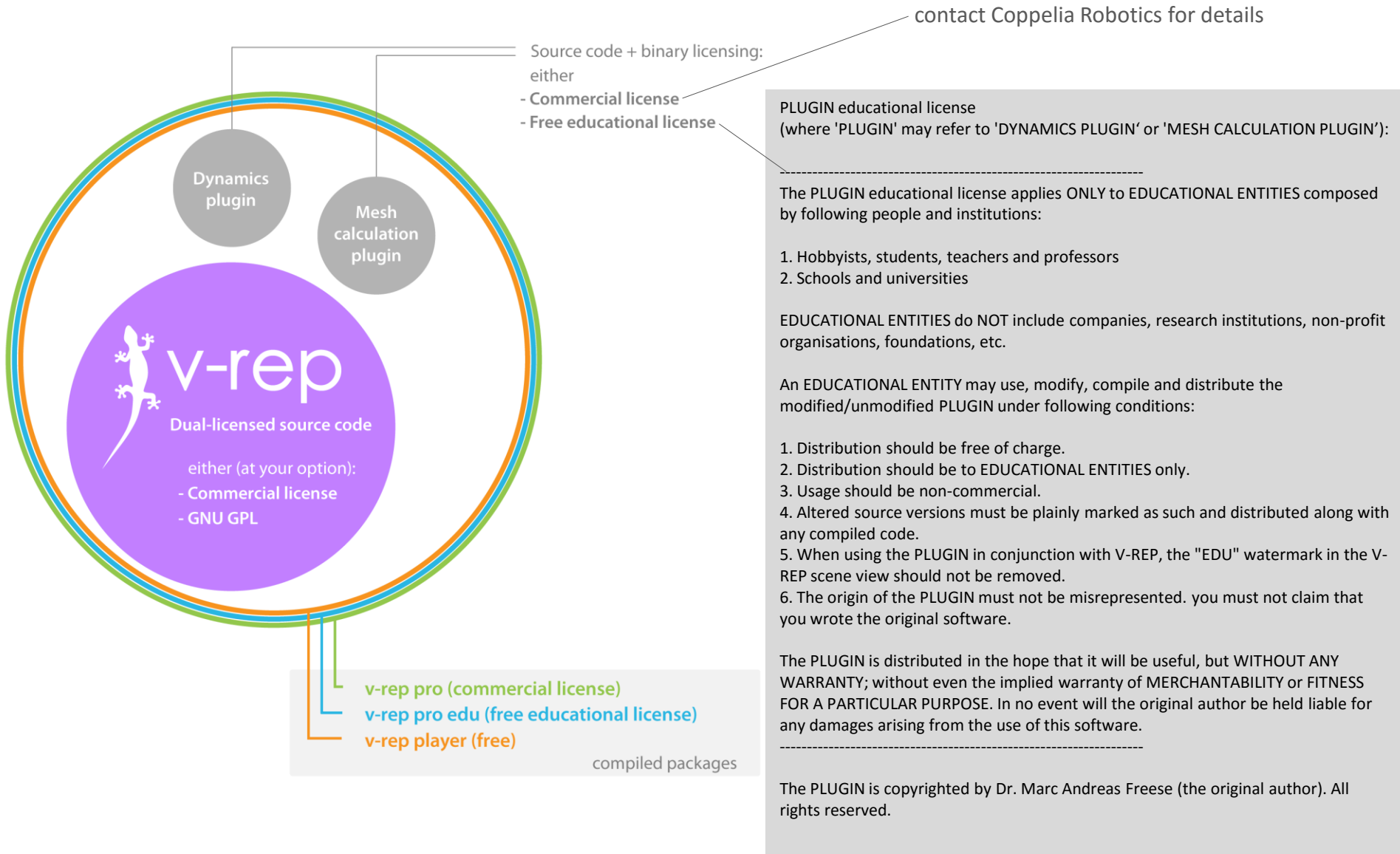
- Interfaces (plugins, embedded scripts, add-ons, Remote API, ROS)
- Languages (C/C++, Java, Python, Lua, Matlab, Octave, Lua, Urbi)
- Physics engines (Bullet, ODE, Vortex, Newton)
- Platforms (Windows, MacOS, Linux)

- V-REP PRO EDU**
 - For hobbyists, students, teachers, professors, schools and universities
 - Free
 - No limitations (i.e. fully functional)
 - No registration
 - Not for commercial applications
 - Not for companies, research institutions, non-profit organizations, etc.

- V-REP PRO**
 - For companies, research institutions, non-profit organizations, etc.
 - Not free
 - No limitations (i.e. fully functional)
 - For commercial applications

- V-REP PLAYER**
 - For everyone
 - Free, can be distributed
 - Limited editing capability, saving is disabled
 - For any application

V-REP Source Code Licensing



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](https://www.youtube.com/VirtualRobotPlatform)

V-REP contact: info_at_coppeliarobotics_dot_com