Virtual Robot Experimentation Platform

V-REP

www.coppeliarobotics.com

Many text items in this presentation are active and can be clicked

V-REP Overview

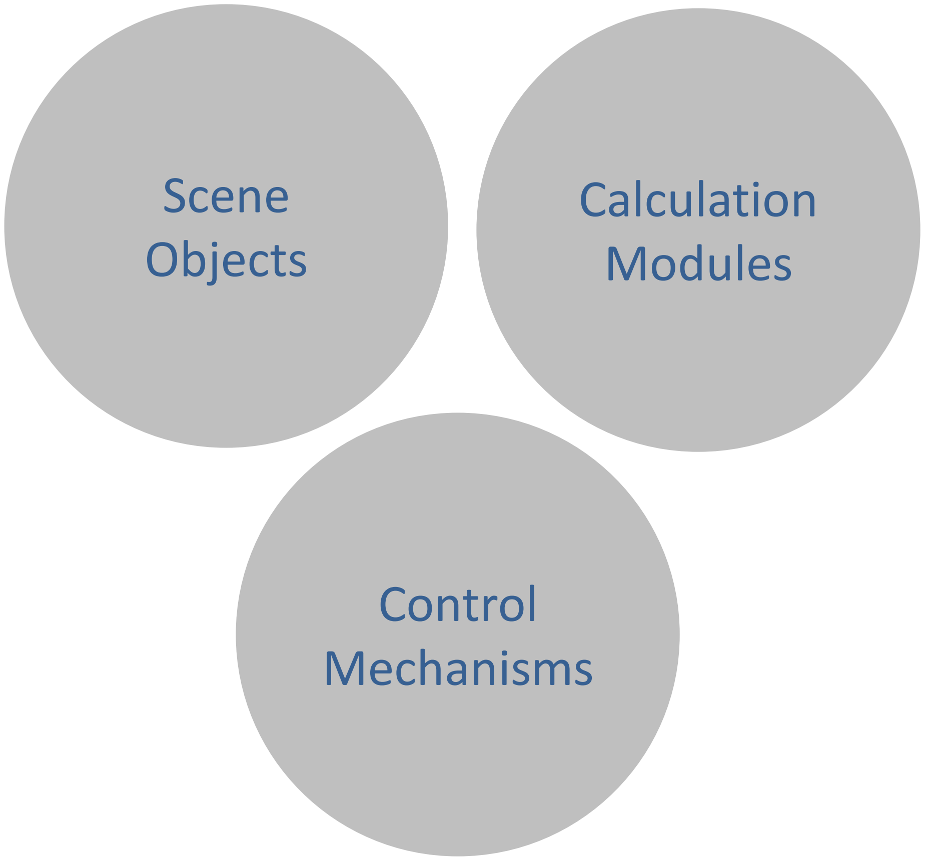
What is it ? General purpose robot simulator with integrated development environment

What can it do ? Sensors, mechanisms, robots and whole systems can be modelled and simulated in various ways **>> 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

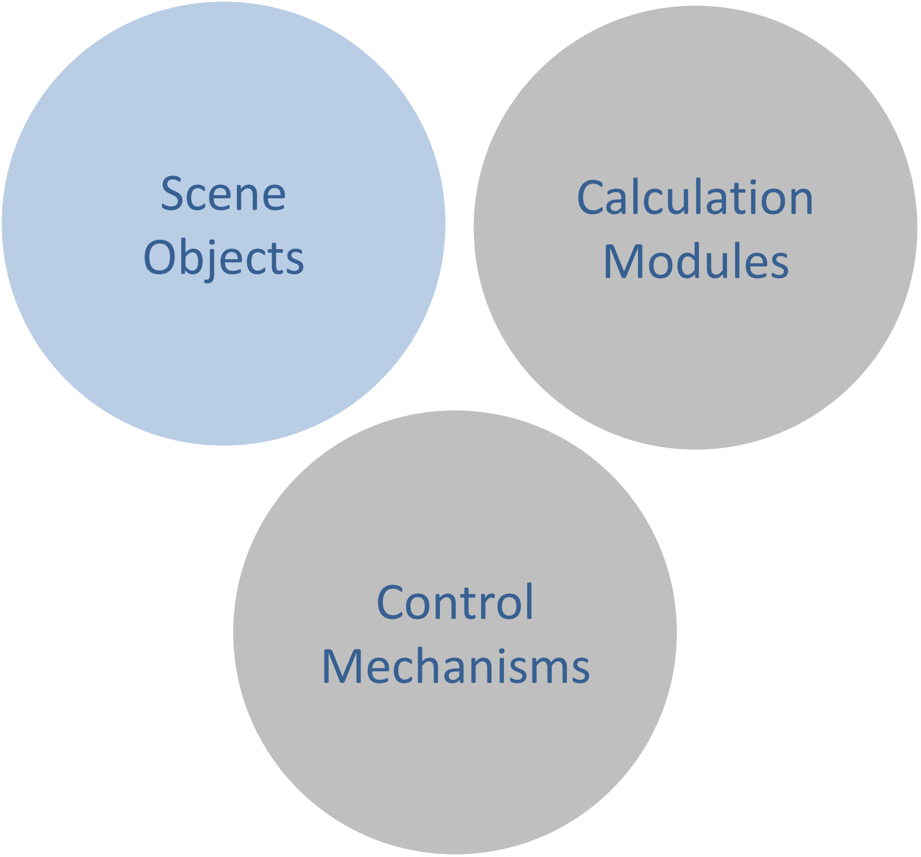
Calculation Modules

Control Mechanisms

Scene Objects

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

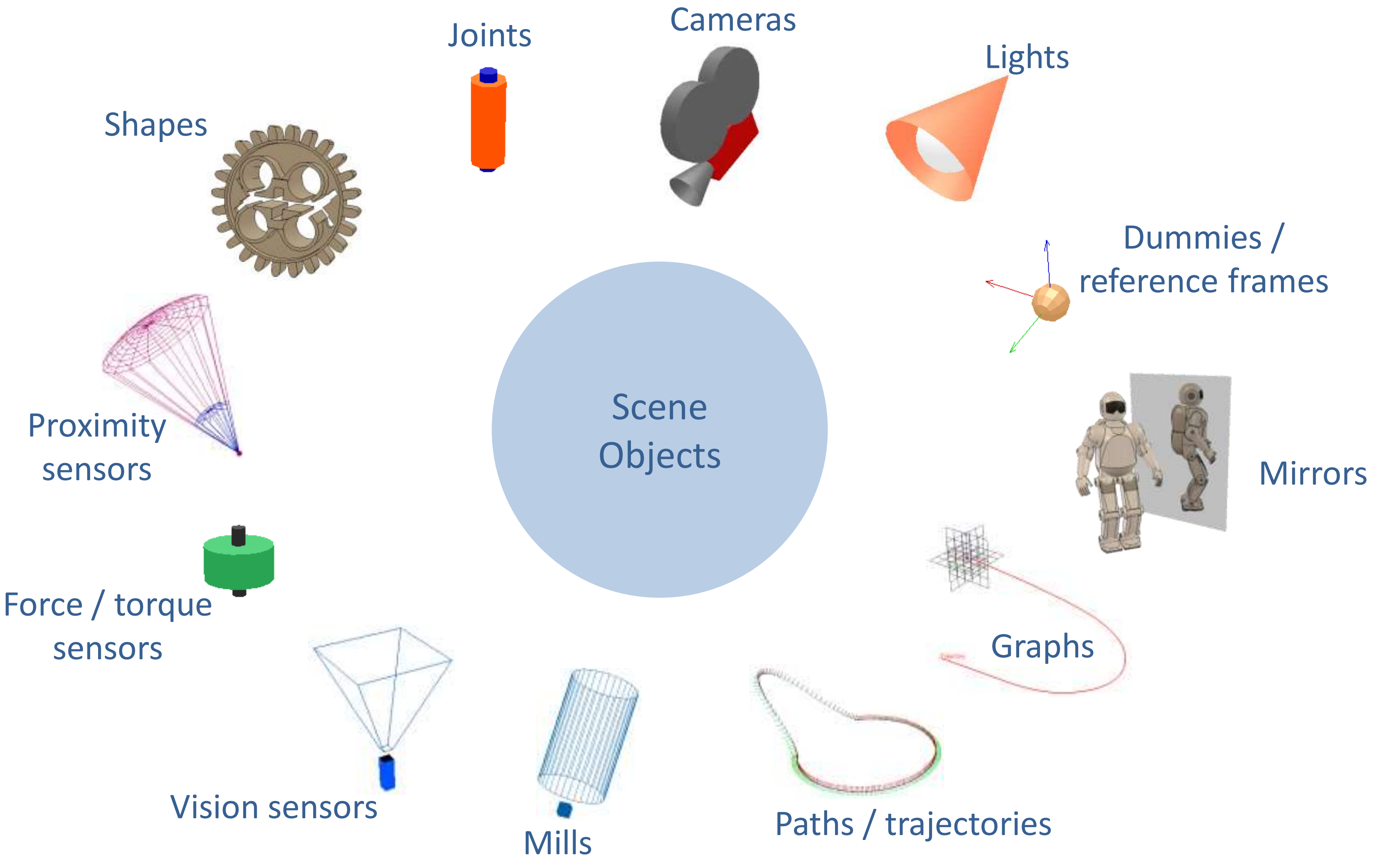


Scene Objects

Calculation Modules

Control Mechanisms

Scene Objects



Scene Objects

Shapes

Joints

Cameras

Lights

Dummies / reference frames

Mirrors

Graphs

Paths / trajectories

Mills

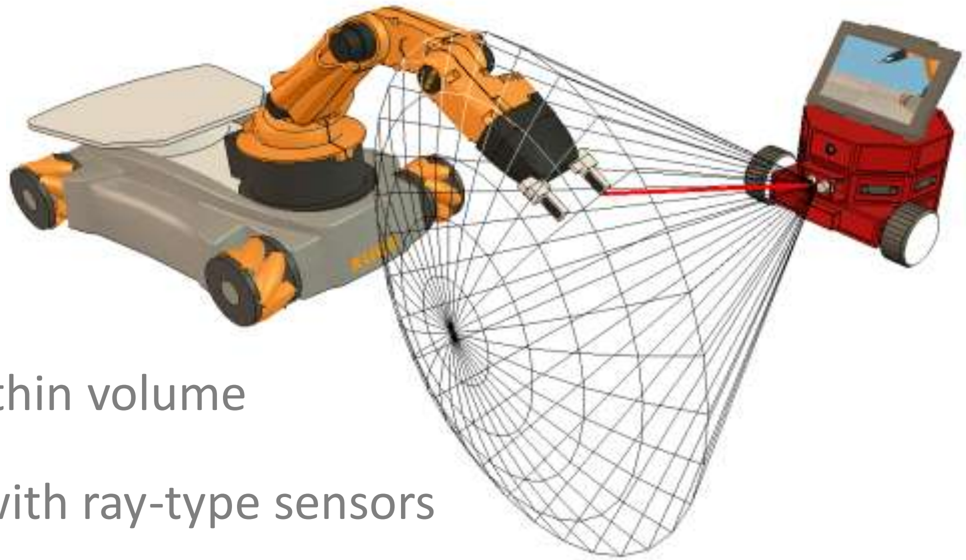
Vision sensors

Force / torque sensors

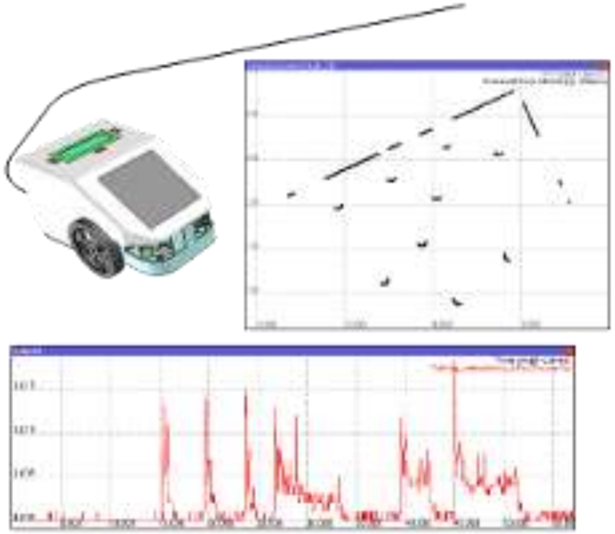
Proximity sensors

Proximity Sensors & Graphs

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

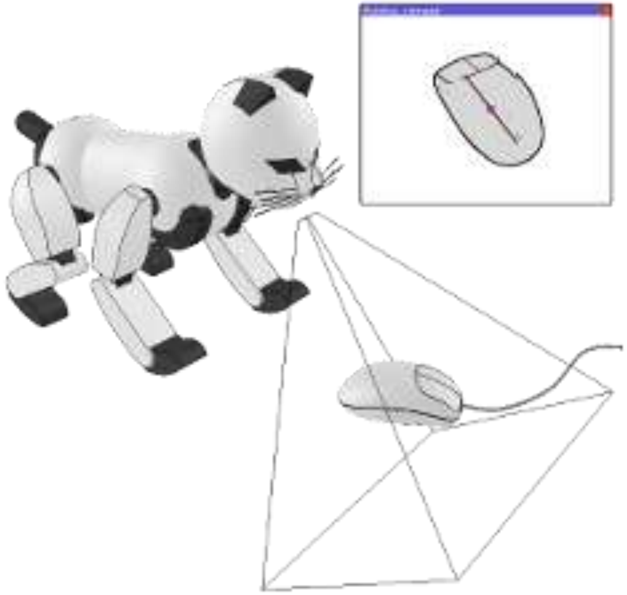


Graphs

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

>> Play demo video

Vision Sensors



Vision Sensors

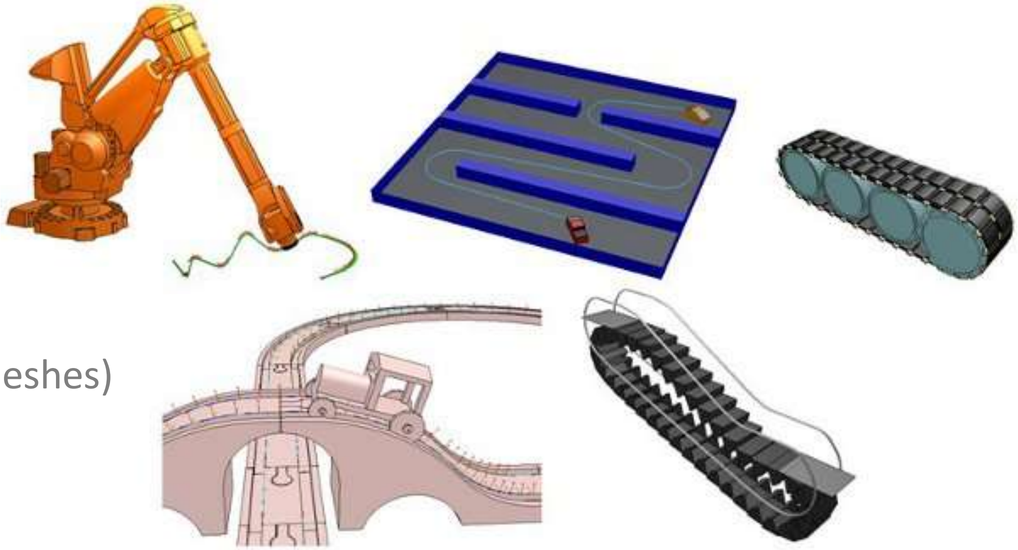
- Integrated image processing
- Extendable via plugin mechanism

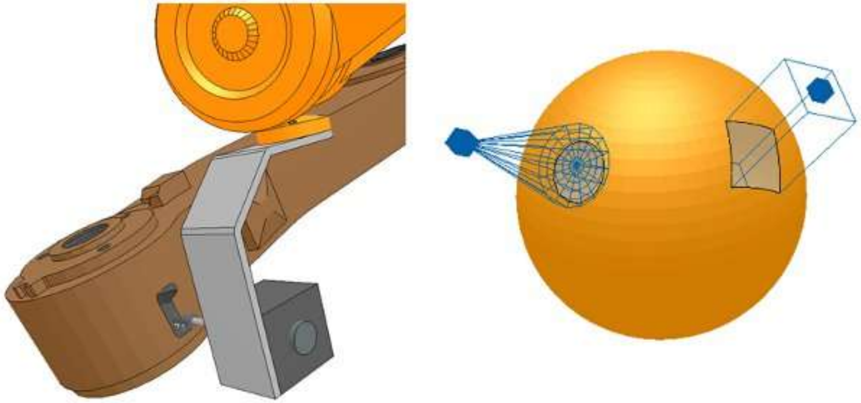
>> Play demo video

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)

>> Play demo video

Cameras, Lights and Mirrors

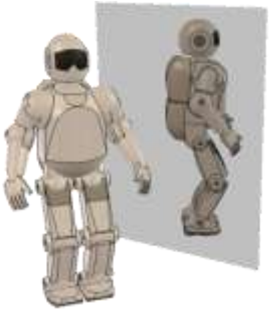
v-rep

Cameras

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

Lights

Spotlight / directional / omnidirectional

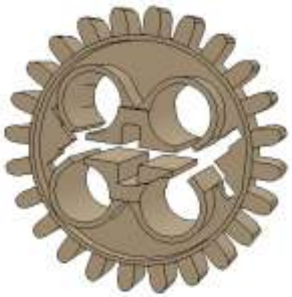


Mirrors

Mirror or scene / object clipping function

>> Play demo video

Joints, Shapes, Force/Torque Sensors, and Dummies

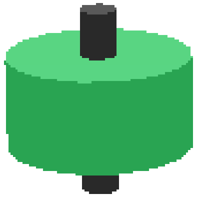


Shapes

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

Joints

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

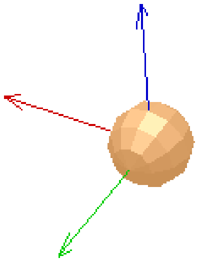


Force/Torque Sensors

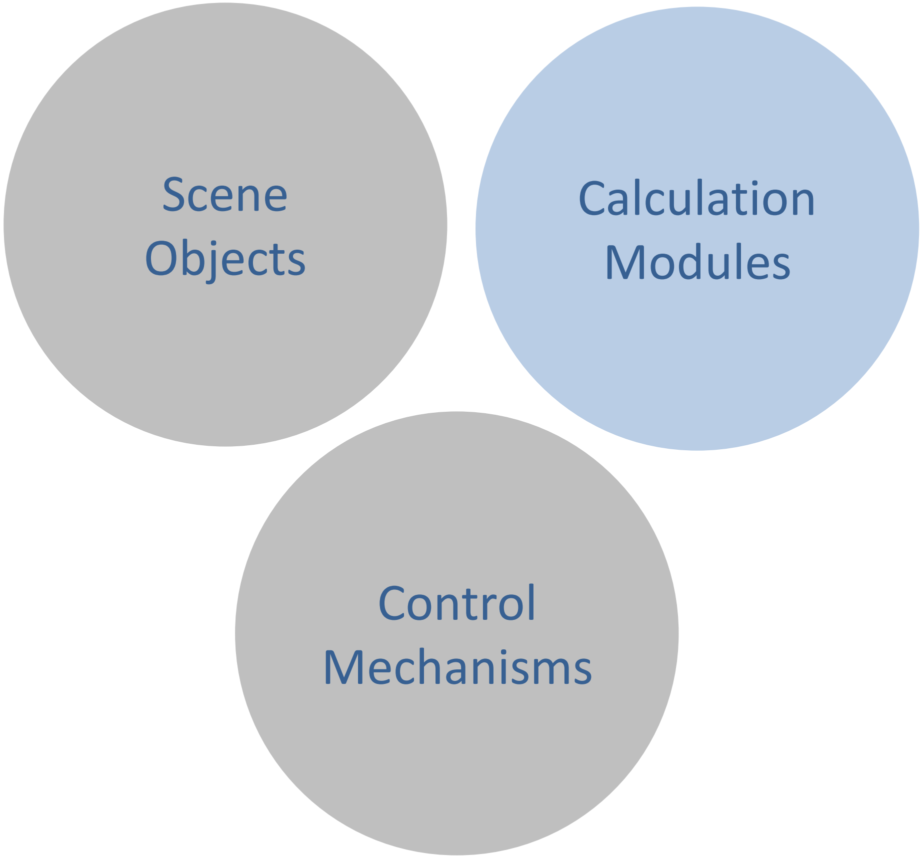
- Measures force and torque
- Can conditionally break apart

Dummies

Auxiliary refenrence frame & helper object



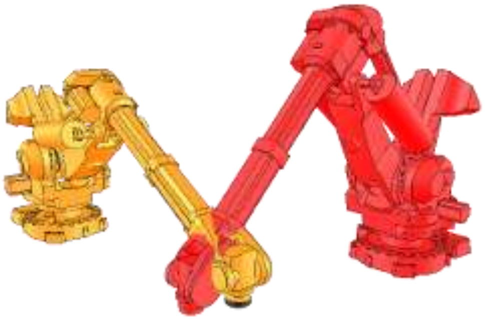
3 Central Elements



Calculation modules

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

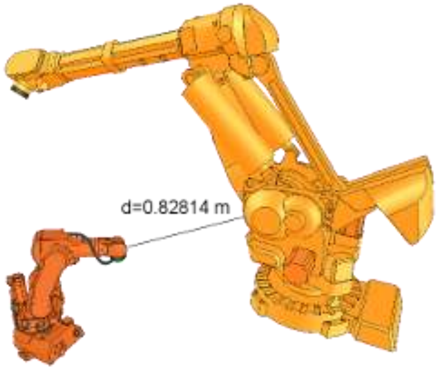
Calculation Modules



Collision detection

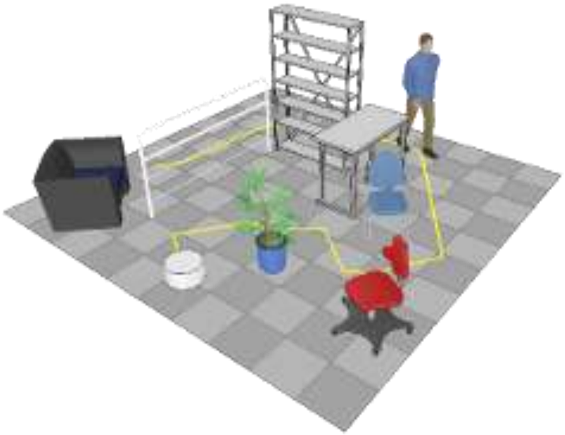
Calculation Modules

Physics / Dynamics



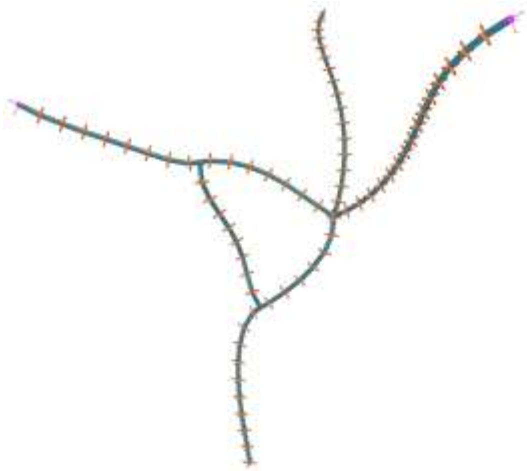
Minimum distance calculation

Forward / Inverse kinematics



Path planning

Kinematics and Distance Calculation

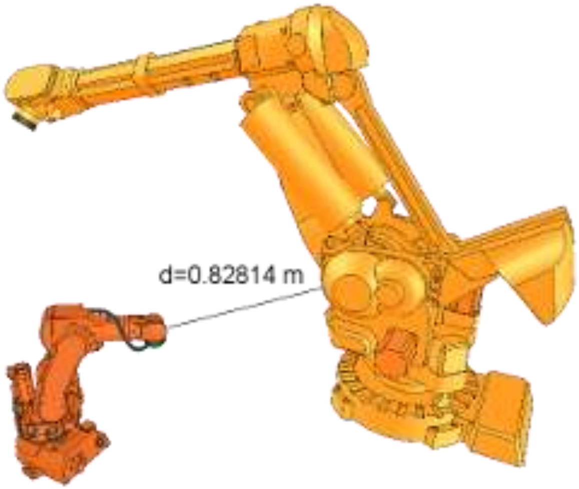


Inverse / forward Kinematics

- 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)



>> Play demo video

Dynamics

Dynamics / Physics

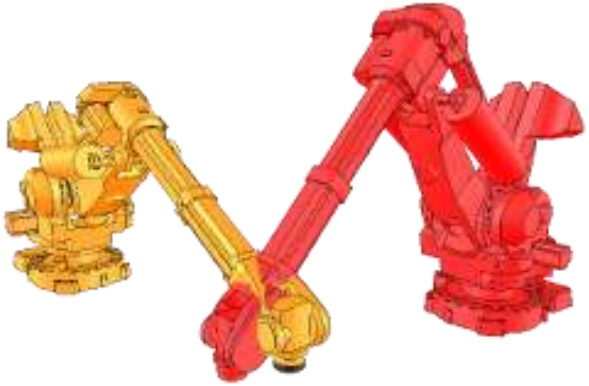
- 2 physics engines: Bullet and ODE
- Simple mouse click to switch
- Dynamic particles to simulate air or water jets
- Can work hand-in-hand with kinematics module

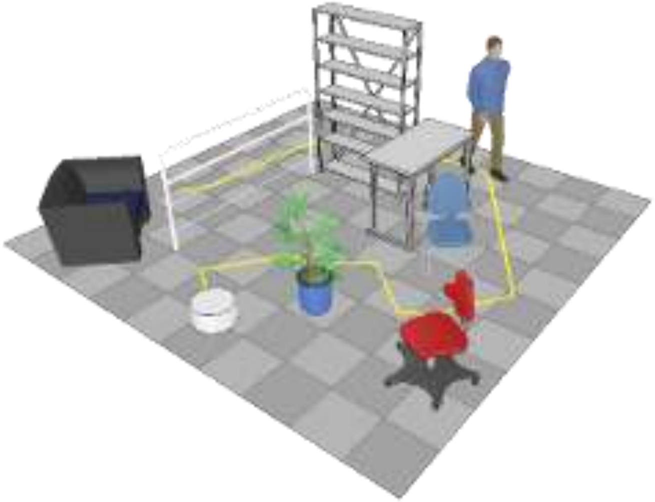
>> Play demo video

Collision Detection and Path Planning

Collision Detection

Any mesh (also open / concave / polygon soups)



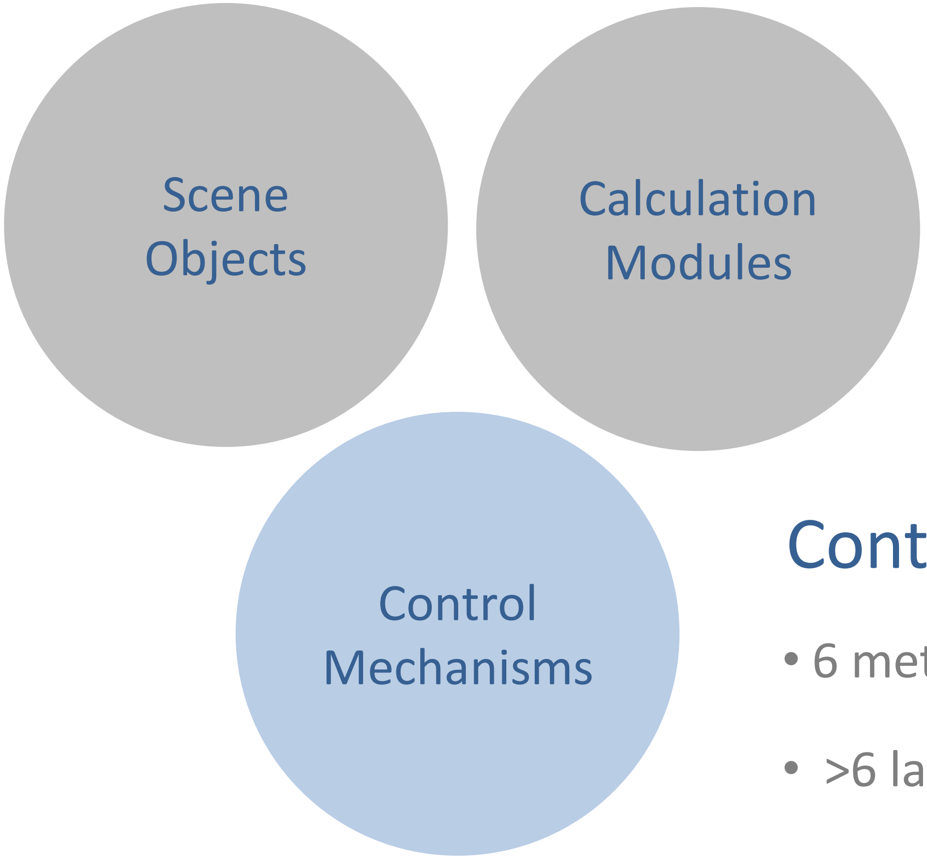


Path Planning

- Holonomic in 2-6 dimensions
- Non-holonomic for car-like vehicles

>> Play demo video

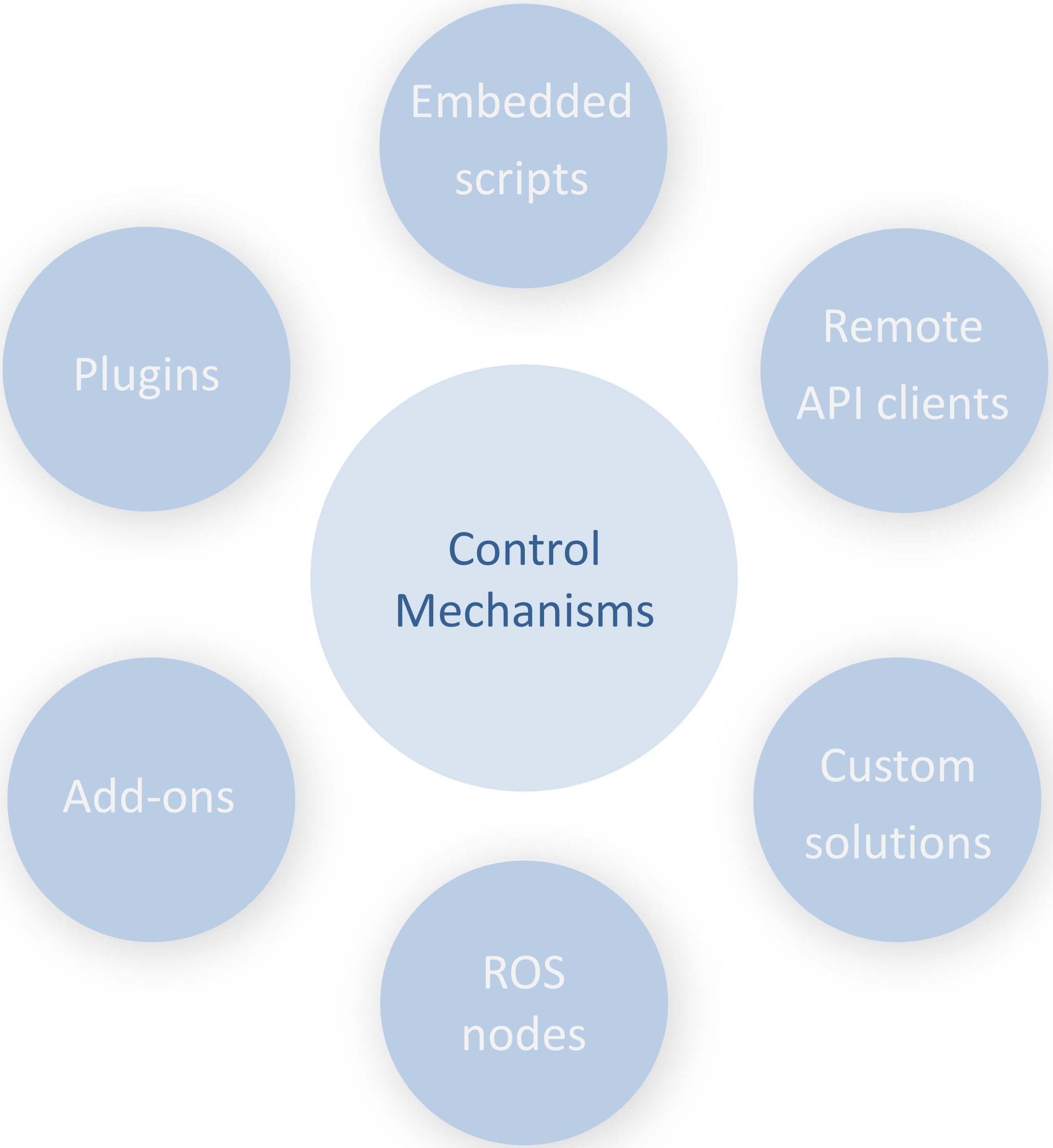
3 Central Elements



Control Mechanisms

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

Control Mechanisms



Control Mechanisms

Embedded scripts

Remote API clients

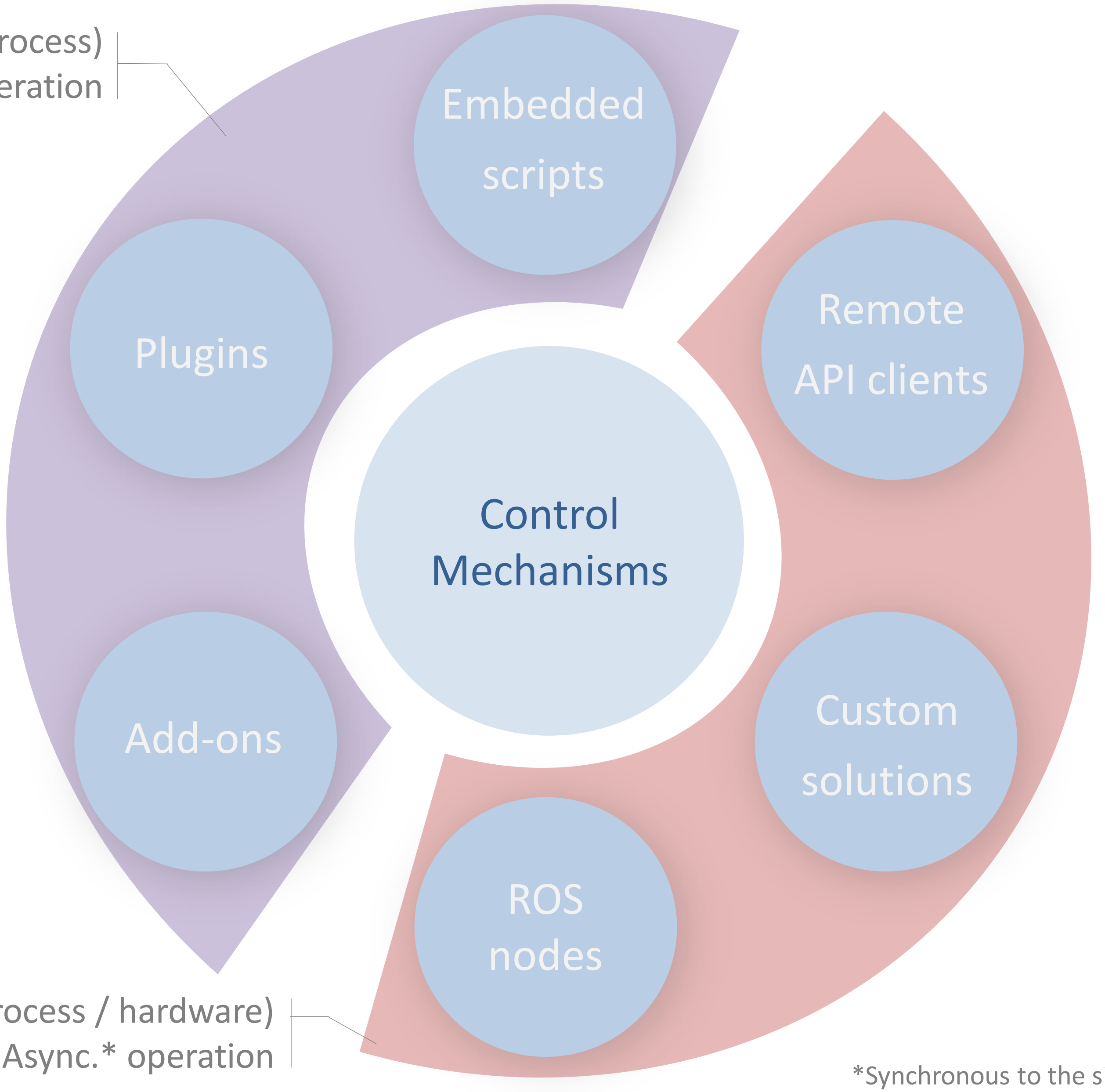
Custom solutions

ROS nodes

Add-ons

Plugins

Local and Remote Interfaces



Local (i.e. same process)
Sync.* or async.* operation

Embedded scripts

Plugins

Add-ons

Control Mechanisms

Remote API clients

Custom solutions

ROS nodes

Remote (i.e. different process / hardware)
Async.* operation

*Synchronous to the simulation loop

Local 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
- Lua interface
- Lightweight and easy to program
- Extremely portable solution
- Threaded or non-threaded. Threads can be synchronized easily
- Various types: main script, child scripts, callback scripts (e.g. custom joint controllers)

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 Interfaces

Remote API clients

Remote API

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

ROS nodes

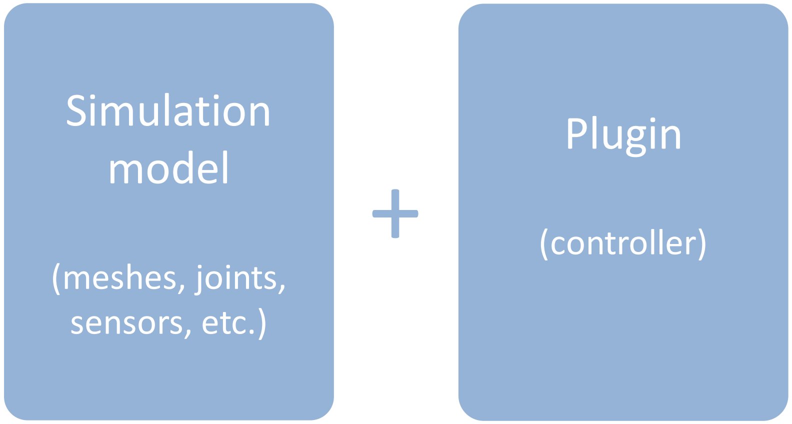
ROS Interface

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

Embedded Script Advantages 1/6

Controller Integration

Plugins



Simulation model

(meshes, joints, sensors, etc.)

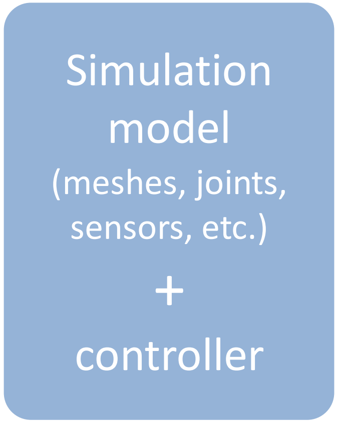
+

Plugin

(controller)

2 items

Embedded scripts



Simulation model

(meshes, joints, sensors, etc.)

+

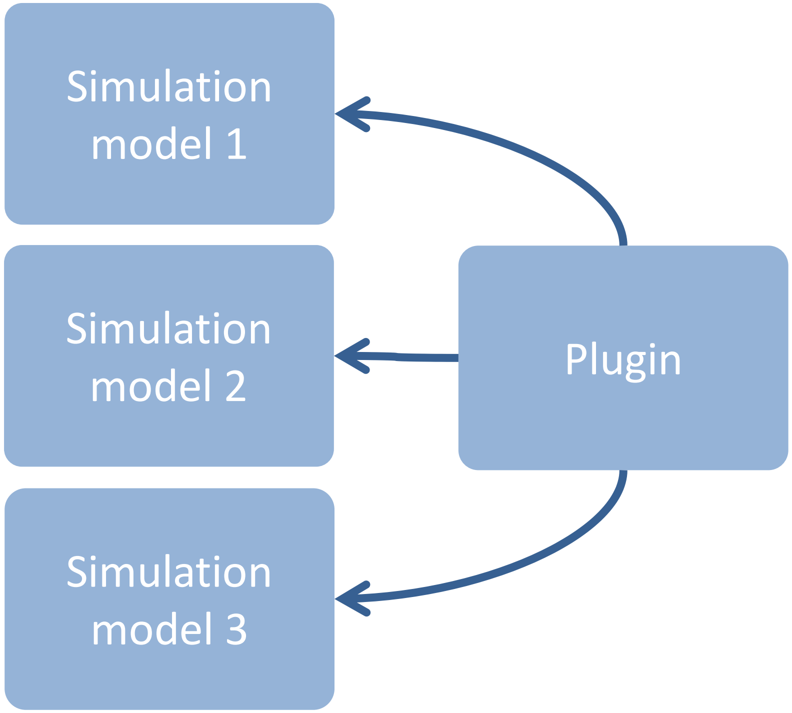
controller

1 item

Embedded Script Advantages 2/6

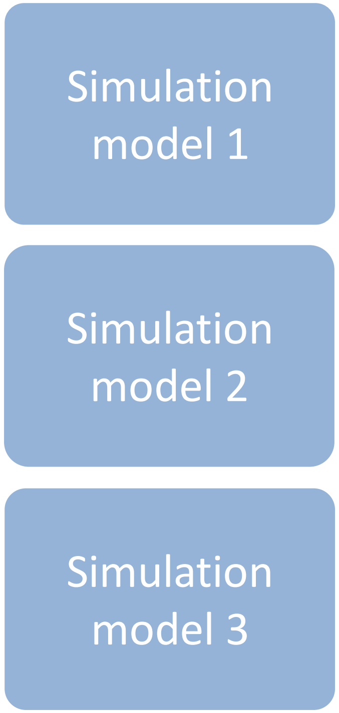
Scalability

Plugins



Plugin has to manage instances

Embedded scripts

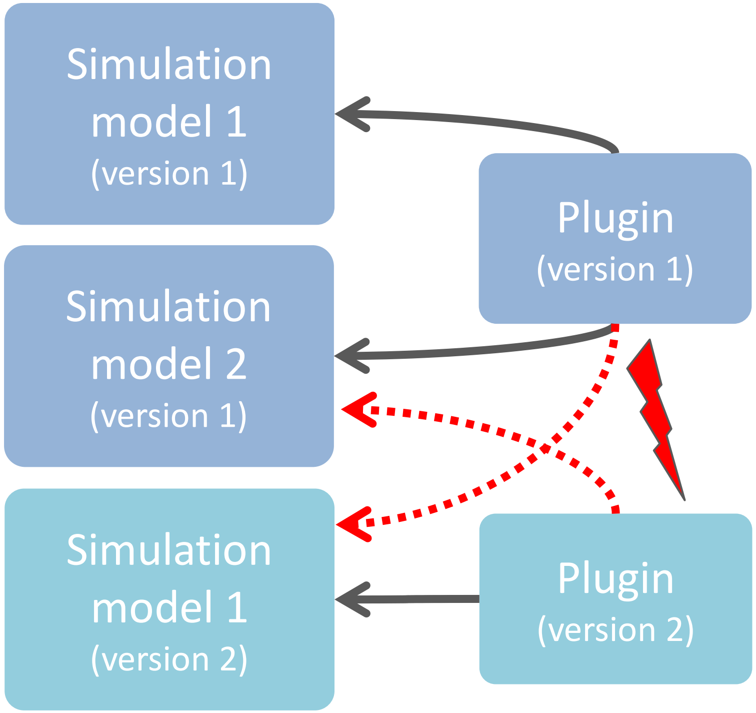


Scalability is inherent

Embedded Script Advantages 3/6

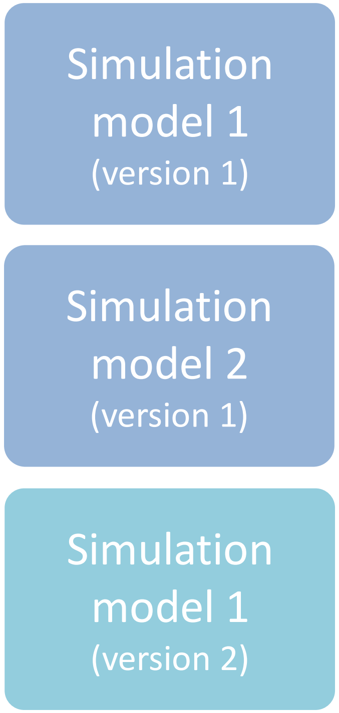
Version Conflicts

Plugins


Simulation model 1 (version 1)
Plugin (version 1)
Simulation model 2 (version 1)
Simulation model 1 (version 2)
Plugin (version 2)

High chances for conflicts

Embedded scripts


Simulation model 1 (version 1)
Simulation model 2 (version 1)
Simulation model 1 (version 2)

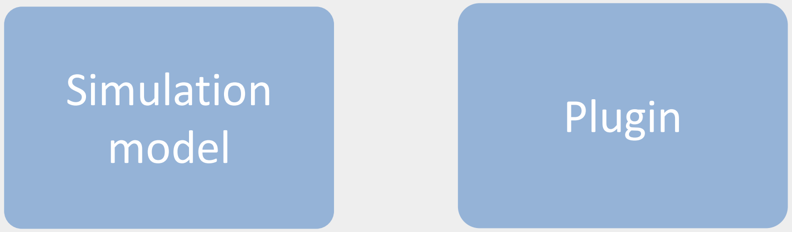
No chances for conflicts

Embedded Script Advantages 4/6

Portability

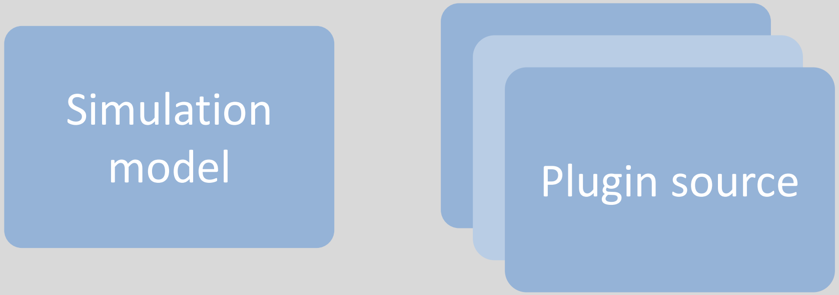
Plugins

Same OS



2 files

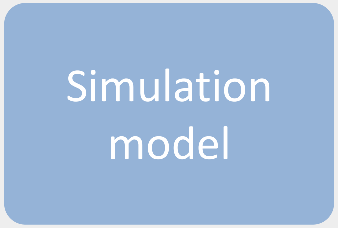
Different OS



Many files
Compilation required
OS-specific problems

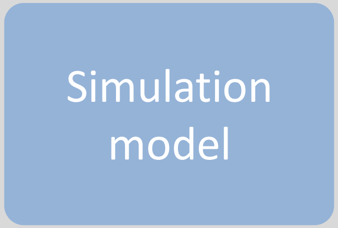
Embedded scripts

Same OS



1 file

Different OS



1 file
No compilation required
No OS-specific problems

Embedded Script Advantages 5/6

Other Considerations

Plugins

Creation, compilation and installation difficulty:
High

Model modification difficulty:
High

Maintenance over the years:
OS-dependant

Embedded scripts

Creation, compilation and installation difficulty:
Low

Model modification difficulty:
Low

Maintenance over the years:
OS-independant

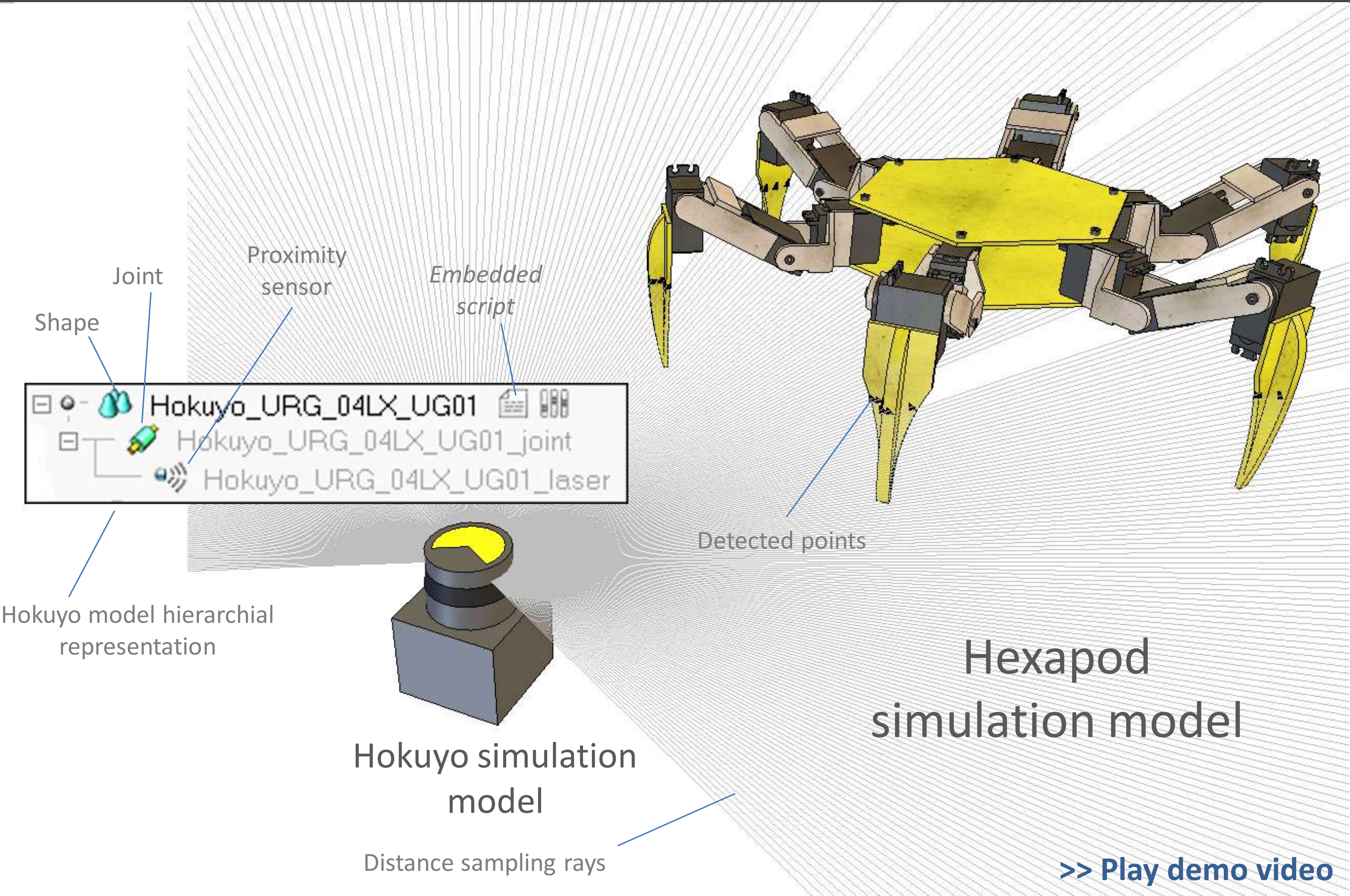
Embedded Script Advantages 6/6

Synchronization with Simulation Loop

	Plugins	Embedded scripts
Non-threaded	Control routine called at each simulation pass Easy	Control routine called at each simulation pass Easy

Threaded	Complex synchronization mechanism required
	Difficult
		Control routine thread can behave as a coroutine
		e.g. simSwitchThread()
		simSetThreadSwitchTiming(delay)
		simSetThreadIsFree(isFree)
		simSetThreadResumeLocation(location,order)
		Easy

Embedded Scripts – Simple Example



Hokuyo_URG_04LX_UG01

Hokuyo_URG_04LX_UG01_joint

Hokuyo_URG_04LX_UG01_laser

Shape

Joint

Proximity sensor

Embedded script

Hokuyo model hierarchial representation

Detected points

Hokuyo simulation model

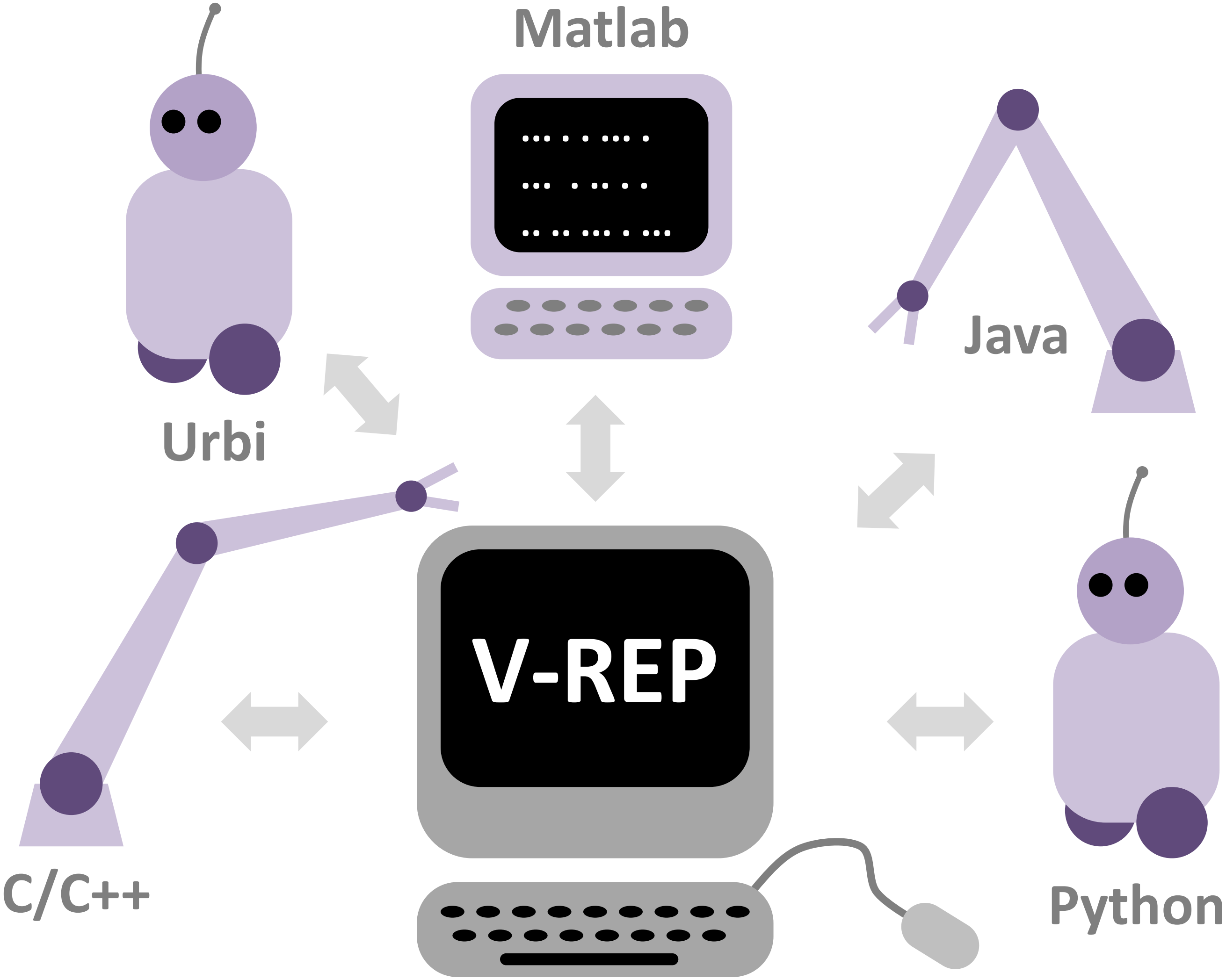
Hexapod simulation model

Distance sampling rays

>> Play demo video

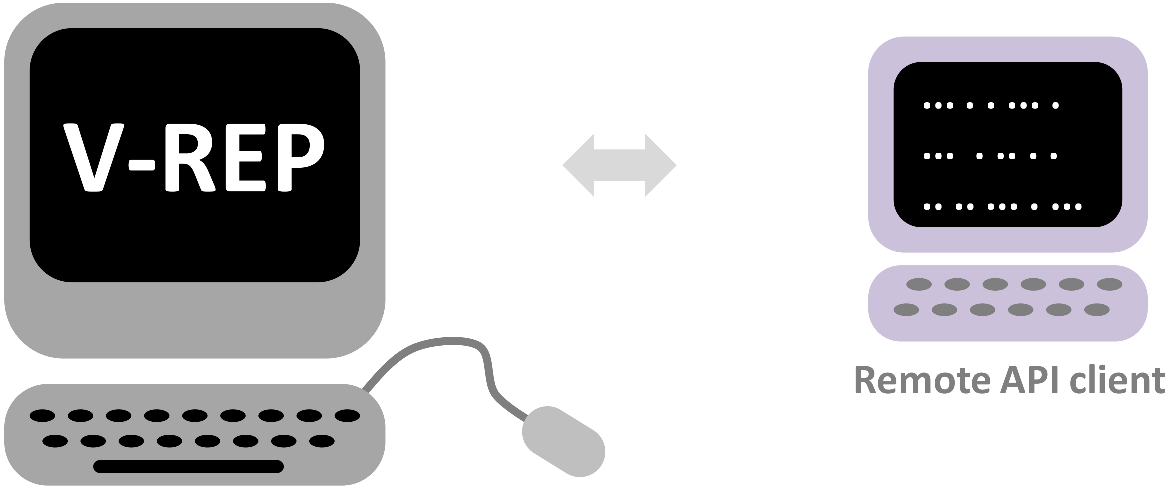
Remote API Advantages 1/2

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

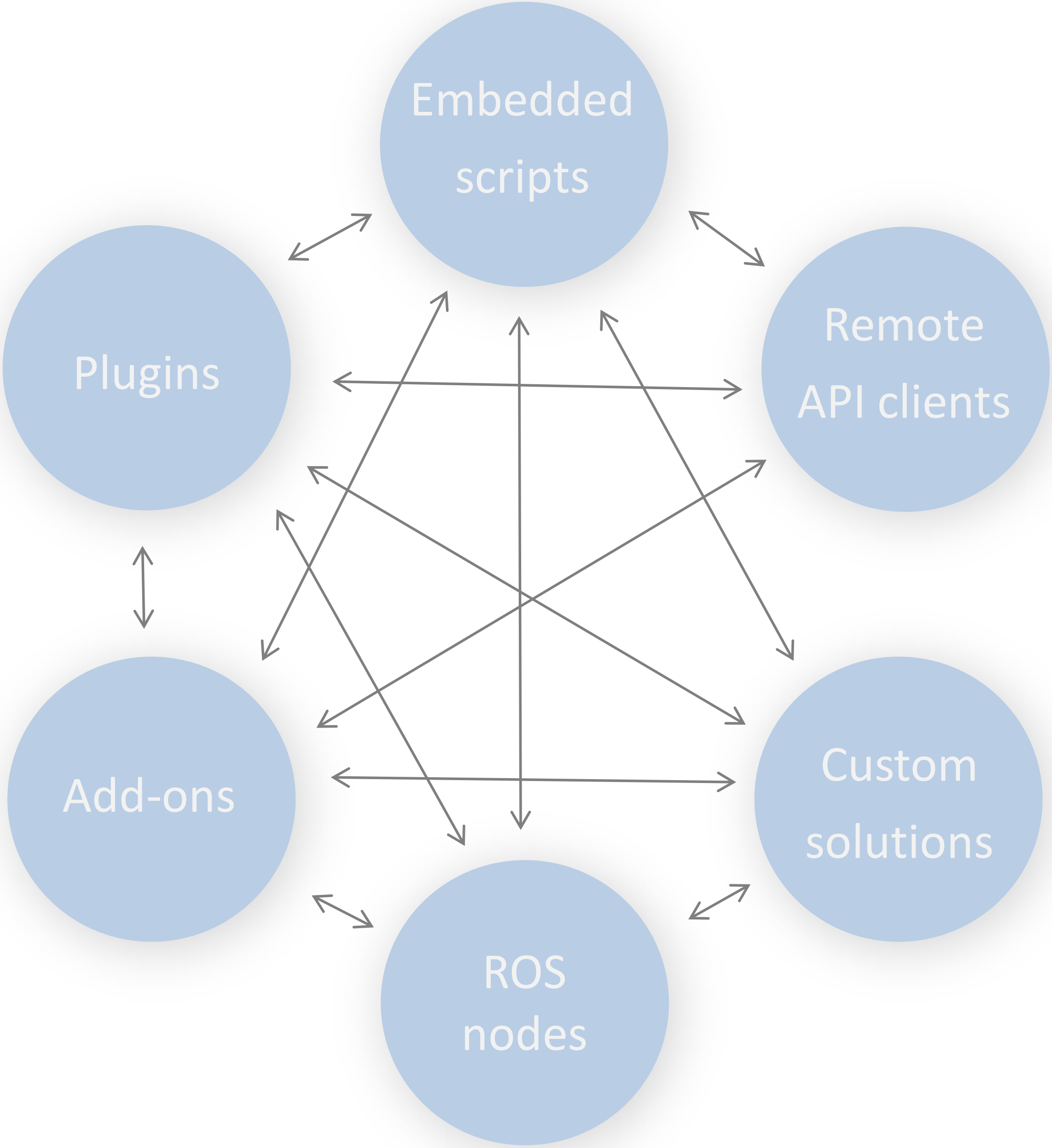
```
int errorCode=simxGetJointPosition(jointHandle,&position,operationMode);
```

- simx_error_noerror
- simx_error_timeout_flag
- simx_error_novalue_flag
- simx_error_remote_error_flag
- simx_error_local_error_flag
- etc.

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

Blocking mode

Collaborative Control Mechanisms



Embedded scripts

Remote API clients

Custom solutions

ROS nodes

Add-ons

Plugins

Example of Collaborative Mechanism 1 / 3

Remote API client

Appends coordinate information to the signal „objCreation“

```
int pos[3]={1.0f,2.0f,3.0f};
simxAppendStringSignal("objCreation",(simxChar*)pos,3*4,simx_opmode_oneshot);
```

Embedded script

Creates cylinders at the coordinates indicated in the signal „objCreation“

```
signalData=simGetStringSignal("objCreation")
simClearStringSignal("objCreation")
if signalData then
  data=simUnpackFloats(signalData)
  for i=0,(#data-1)/3,1 do
    handle=simCreatePureShape(2,22,{0.1,0.1,0.1},0.1)
    simSetObjectPosition(handle,-1,{data[3*i+1],data[3*i+2],data[3*i+3]})
  end
end
```

Example of Collaborative Mechanism 2 / 3

Plugin

```
void SCRIPT_DO_SOME_MAGIC_CALLBACK(SLuaCallBack* p)
{
  ... ( gets called when a script calls „simxExt_doSomeMagic“ )
}


// Initialization phase of plugin: register new script commands:
#define SCRIPT_DO_SOME_MAGIC "simExt_doSomeMagic"
int inArgs[]={2,sim_lua_arg_int,sim_lua_arg_float|sim_lua_arg_table};

simRegisterCustomLuaFunction(SCRIPT_DO_SOME_MAGIC,strConCat("number value1,
         number value2=",SCRIPT_DO_SOME_MAGIC,"(number index,table inVals)"),
         inArgs,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

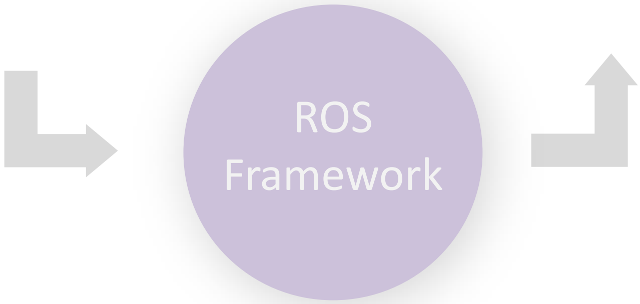
Example of Collaborative Mechanism 3 / 3

Enable image streaming to ROS

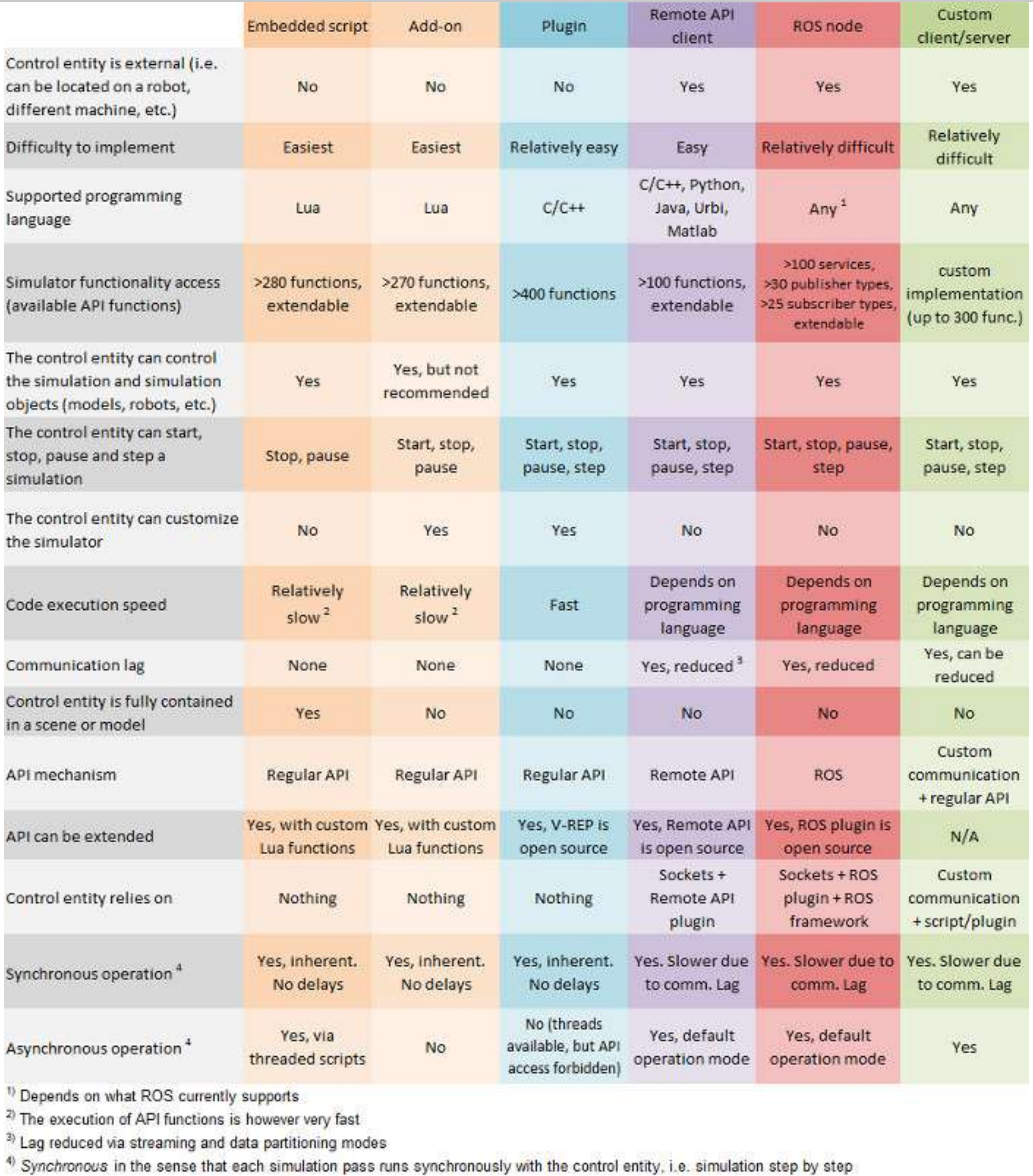
```
-- Following in the script initialization phase (executed just once):

-- 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, Urbi, Matlab	Any[1]	Any
Simulator functionality access (available API functions)	>280 functions, extendable	>270 functions, extendable	>400 functions	>100 functions, extendable	>100 services, >30 publisher types, >25 subscriber types, extendable	custom implementation (up to 300 func.)
The control entity can control the simulation and simulation objects (models, robots, etc.)	Yes	Yes, but not recommended	Yes	Yes	Yes	Yes
The control entity can start, stop, pause and step a simulation	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	No	Yes	Yes	No	No	No
Code execution speed	Relatively slow[2]	Relatively slow[2]	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	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	Nothing	Nothing	Nothing	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

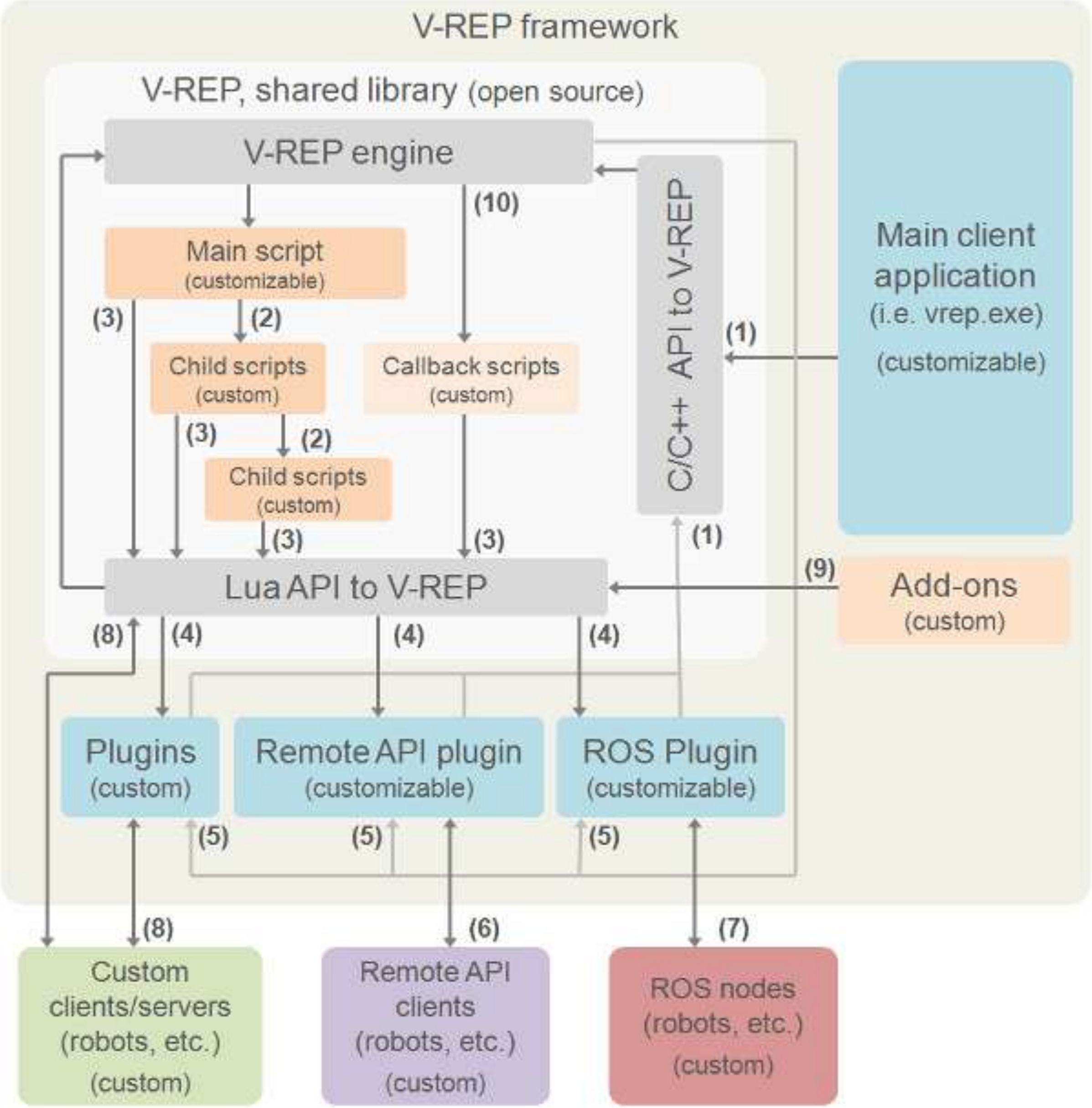
[1] Depends on what ROS currently supports

[2] The execution of API functions is however very fast

[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



(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.)

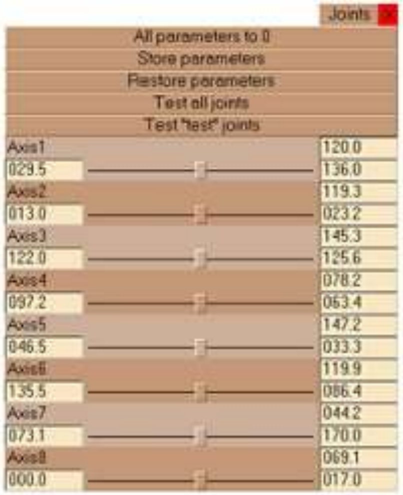
(9) Add-on calls to Lua API

(10) Script callback calls

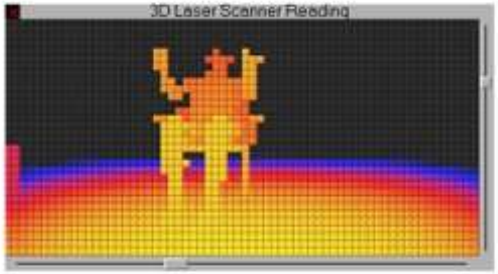
Other Feature: Custom User Interfaces

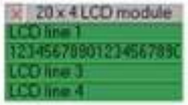
Custom User Interfaces

- 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

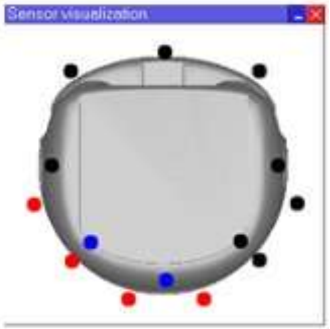


Axis1 120.0
029.5 136.0
Axis2 119.3
013.0 023.2
Axis3 145.3
122.0 125.6
Axis4 078.2
097.2 063.4
Axis5 147.2
046.5 033.3
Axis6 119.9
135.5 086.4
Axis7 044.2
073.1 170.0
Axis8 069.1
000.0 017.0





LCD line 1
12345678901234567890
LCD line 3
LCD line 4



Other Feature: Mesh Edit Modes

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

More Features

- 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.

V-REP Overview

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, Urbi)
- Physics engines (Bullet, ODE)
- Platforms (Windows, MacOS, Linux)

V-REP Flavours

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.

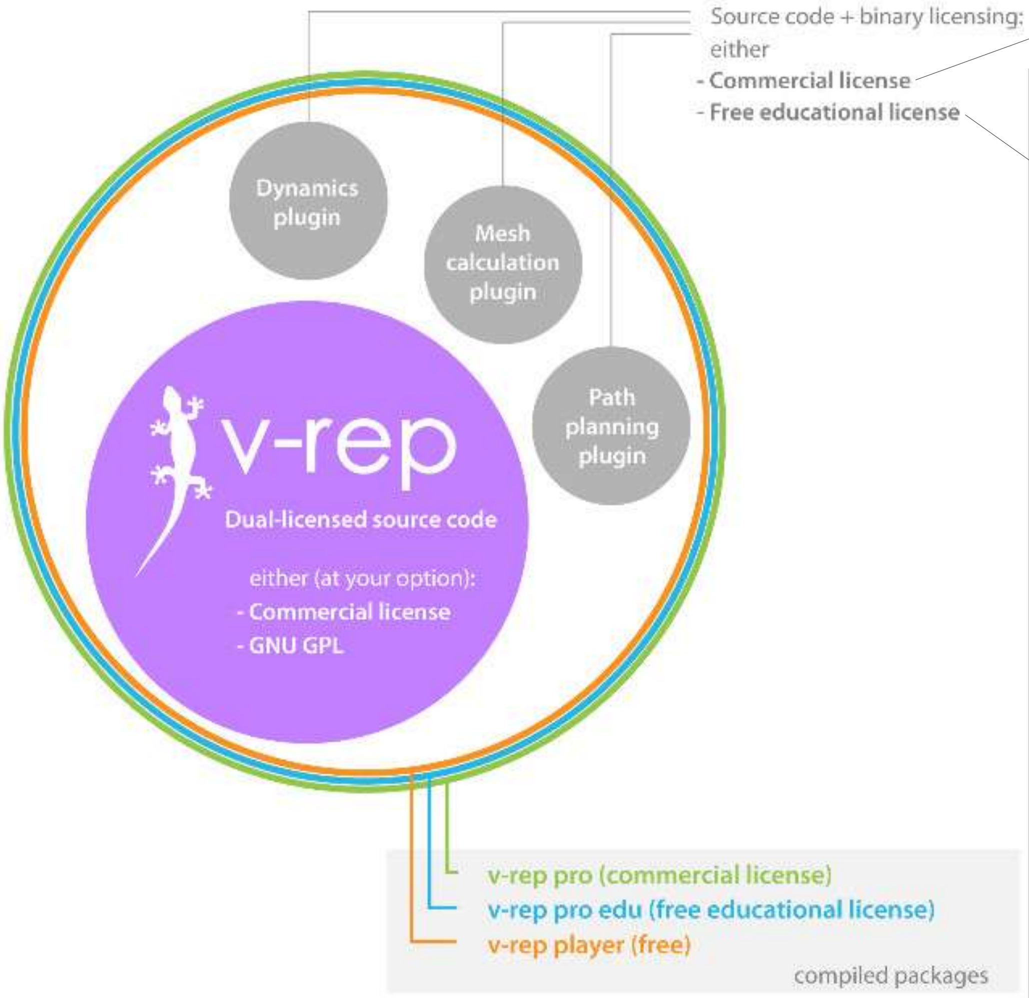
V-REP PRO

- 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



Source code + binary licensing:
either
- Commercial license
- Free educational license

contact Coppelia Robotics for details

Dynamics plugin

Mesh calculation plugin

Path planning plugin

v-rep

Dual-licensed source code

either (at your option):
- Commercial license
- GNU GPL

v-rep pro (commercial license)

v-rep pro edu (free educational license)

v-rep player (free)

compiled packages

PLUGIN educational license
(where 'PLUGIN' may refer to 'DYNAMICS PLUGIN', 'MESH CALCULATION PLUGIN' or 'PATH PLANNING 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.

The PLUGIN is copyrighted by Dr. Marc Andreas Freese (the original author). All rights reserved.

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 contact: info_at_coppeliarobotics_dot_com