Virtual Robot Experimentation Platform

V-REP

www.coppeliarobotics.com

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

V-REP Overview

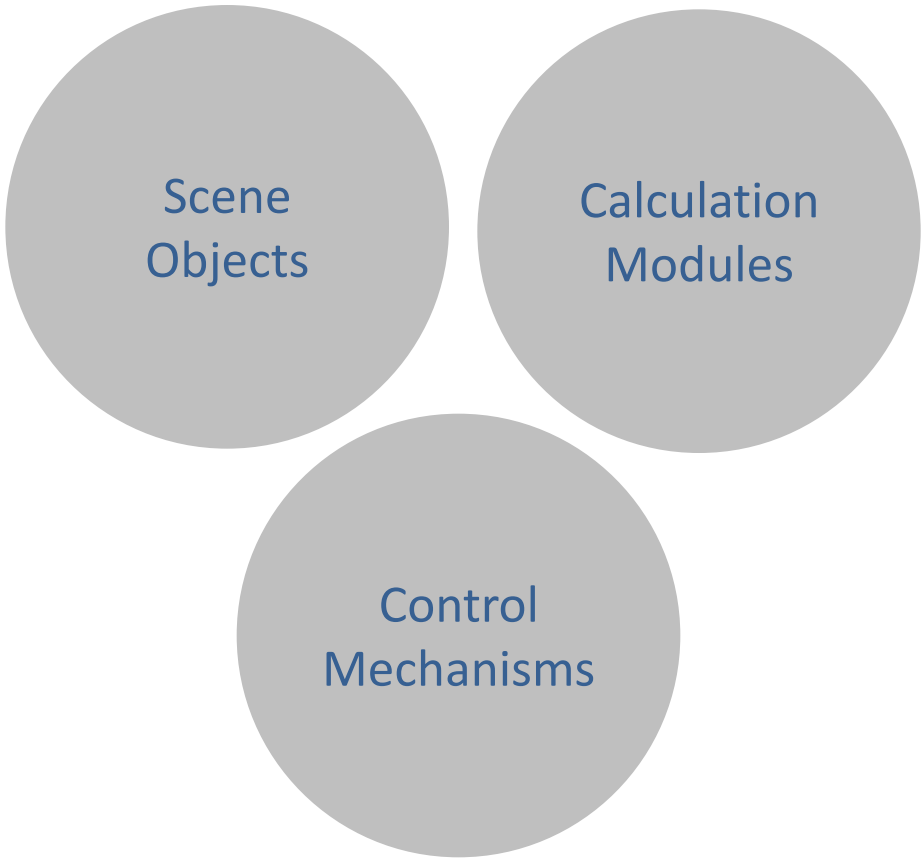
What is it ? General purpose robot simulator with integrated development environment

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

Typical applications ?

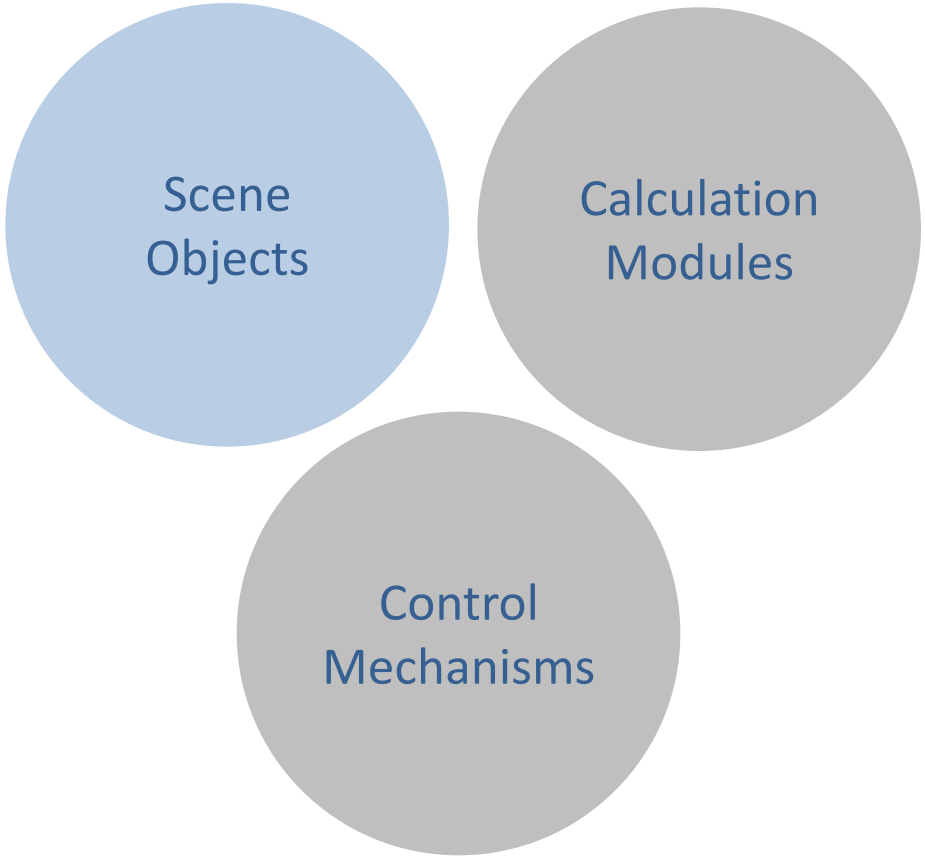
- Fast prototyping and verification
- Fast algorithm development
- Robotics related education
- Remote monitoring
- Hardware control
- Simulation of factory automation systems
- Safety monitoring
- Product presentation
- etc.

3 Central Elements



Scene Objects

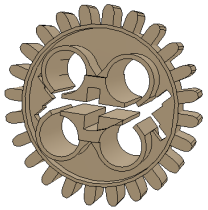
- Basic building blocks
- 14 different types
- Can be combined with each other
- Can form complex systems together with calculation modules and control mechanisms



Scene Objects

Scene Objects

Meshes/Shapes

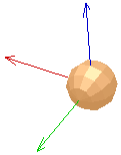


Joints

Cameras

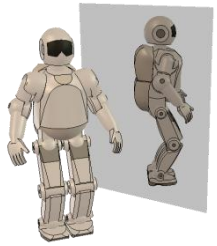
Lights

Dummies/
reference frames

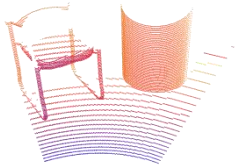


Octrees

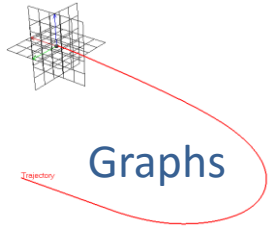
Mirrors



Point
clouds



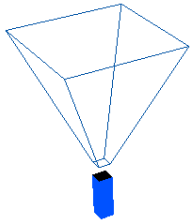
Graphs



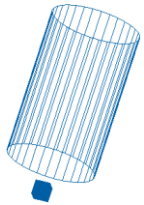
Proximity
sensors

Force/torque
sensors

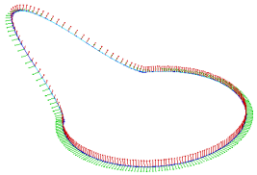
Vision sensors



Mills

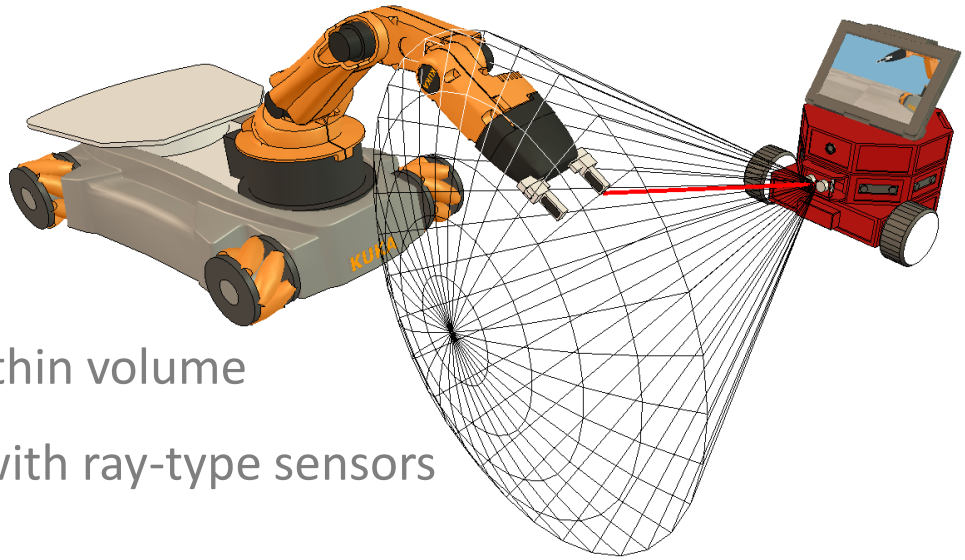


Paths/trajectories

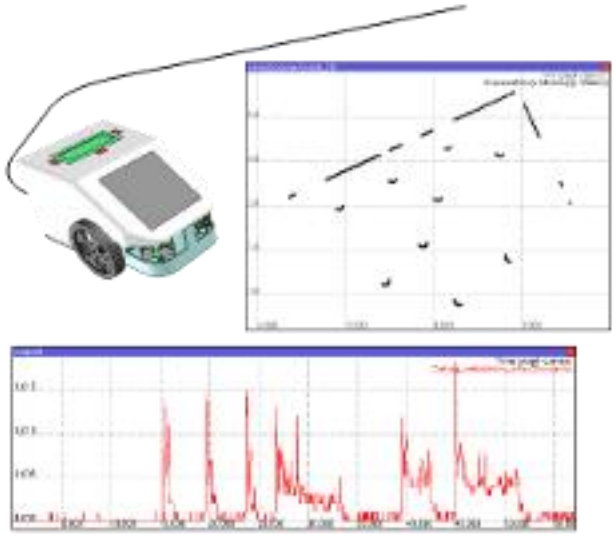


Proximity Sensors & Graphs

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

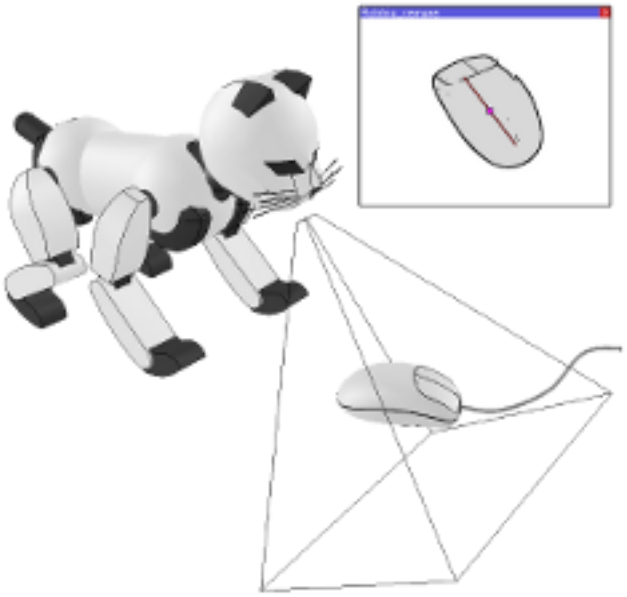


Graphs

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

>> Play demo video1 **>> Play demo video2**

Vision Sensors



Vision Sensors

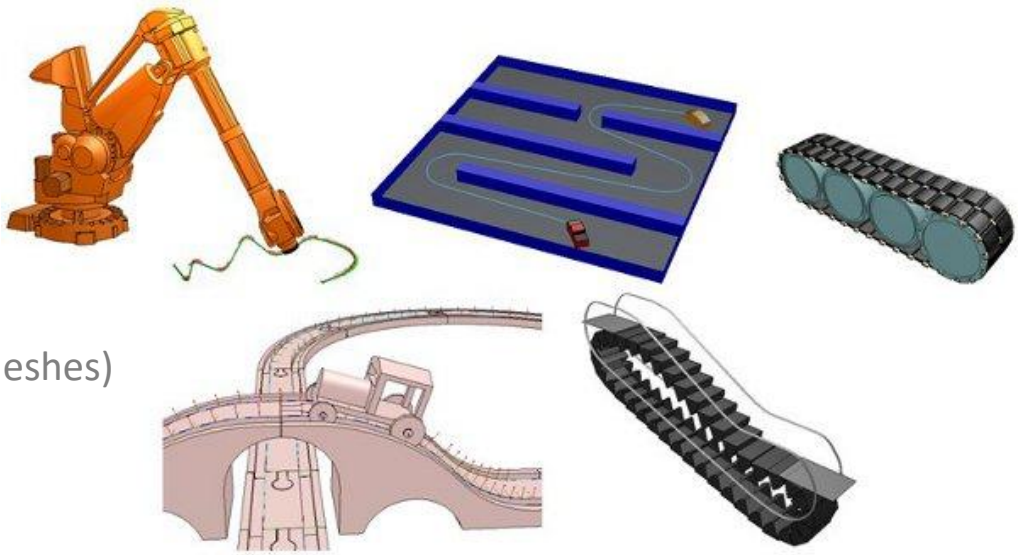
- Integrated image processing
- Extendable via plugin mechanism
- Ray-traced rendering also available

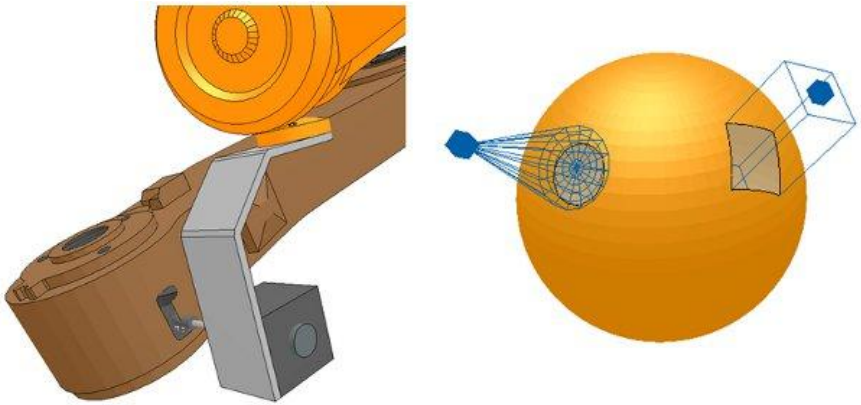
>> Play demo video

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

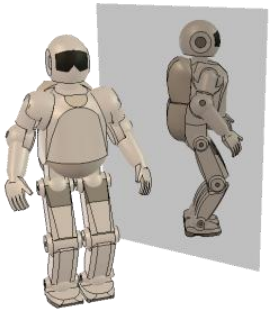
Cameras, Lights and Mirrors

Cameras

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

Lights

Spotlight / directional / omnidirectional

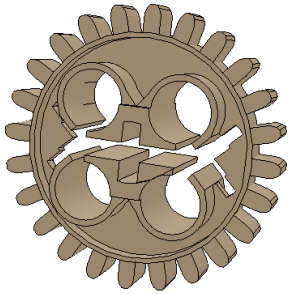


Mirrors

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

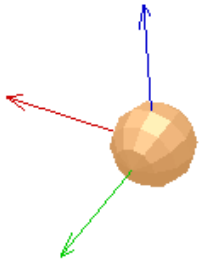
- Revolute-type
- Prismatic-type
- Screw-type
- Spherical-type

Force/Torque Sensors

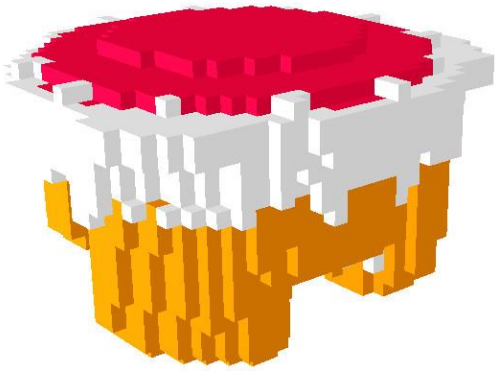
- Measures force and torque
- Can conditionally break apart

Dummies

Auxiliary refenrence frame & helper object



Octrees and Point Clouds

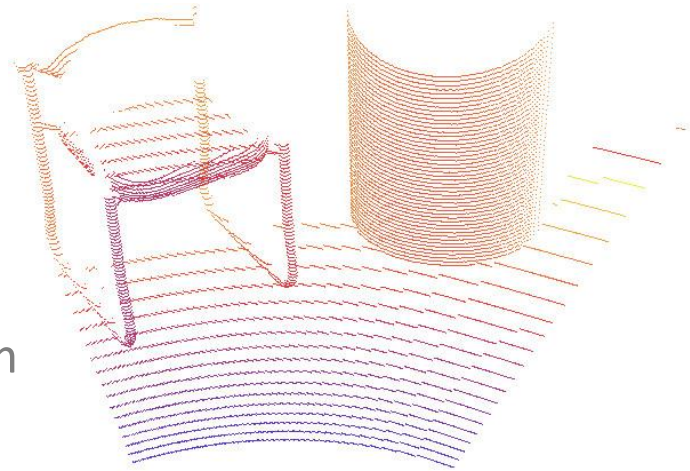


Octrees

- Spatial partitioning, made up by a tree data structure for fast data access
- Voxel-based, can be modified during simulation
- Can be used as a simplified representation of meshes, as an occupancy grid/space, etc.
- Can be used for fast collision detection, minimum distance calculation, proximity sensor detection

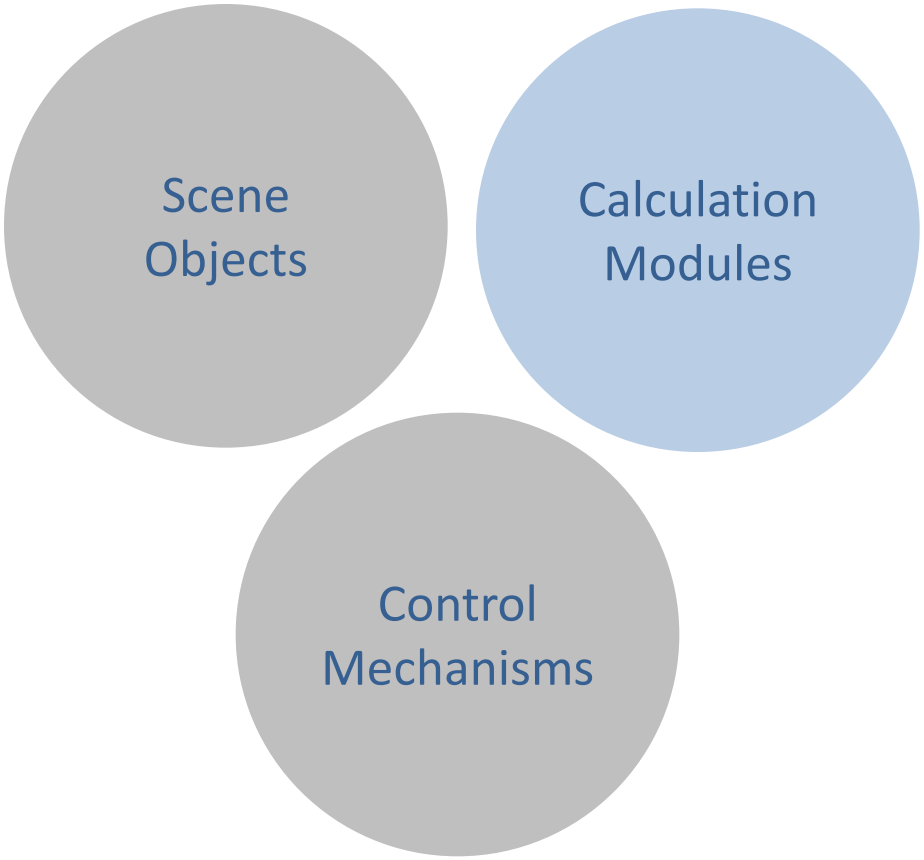
Point Clouds

- Point container
- Octree-based, for fast data access
- Can be used for fast collision detection, minimum distance calculation, proximity sensor detection



>> Play demo video1 **>> Play demo video2** **>> Play demo video3**

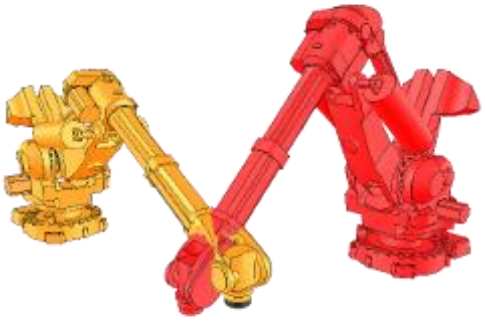
3 Central Elements



Calculation modules

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

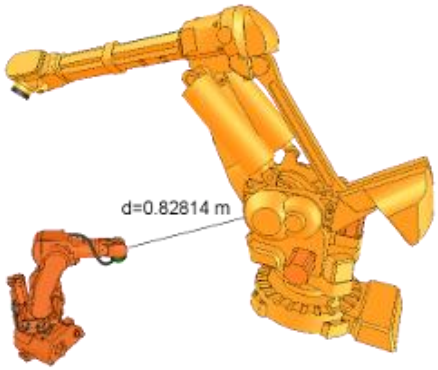
Calculation Modules



Collision detection

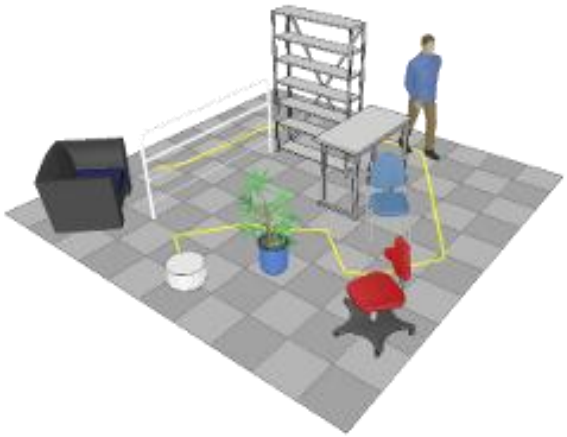
Calculation Modules

Physics / Dynamics



Minimum distance calculation

Forward / Inverse kinematics



Path / motion planning

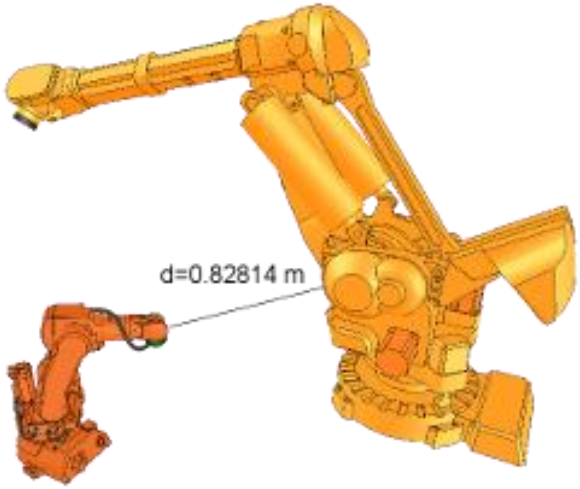
Kinematics and Distance Calculation

Inverse / forward Kinematics

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

Minimum Distance Calculation

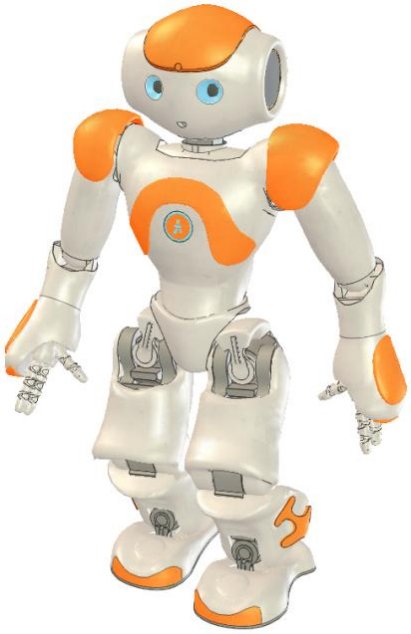
- Any mesh (also open / concave / polygon soups)
- Any octree
- Any point cloud
- Any individual point



>> Play demo video1

>> Play demo video2

Dynamics



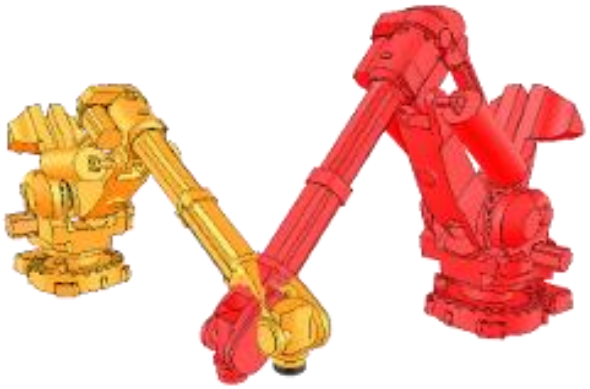
Dynamics / Physics

- 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

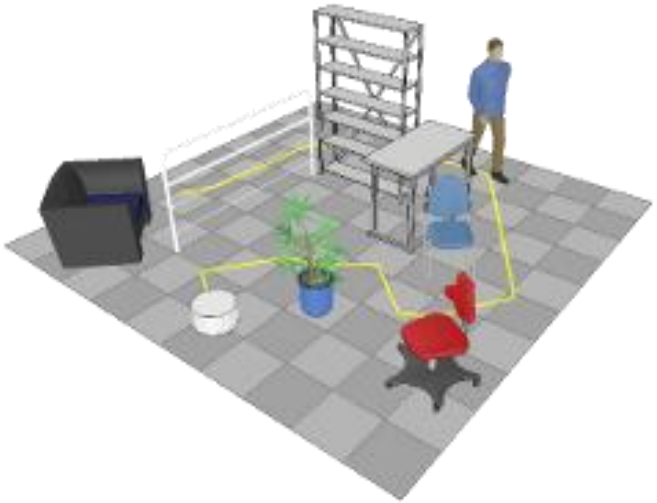
>> Play demo video

Collision Detection and Path Planning

Collision Detection



- Any mesh (also open / concave / polygon soups)
- Any octree
- Any point cloud
- Any individual point



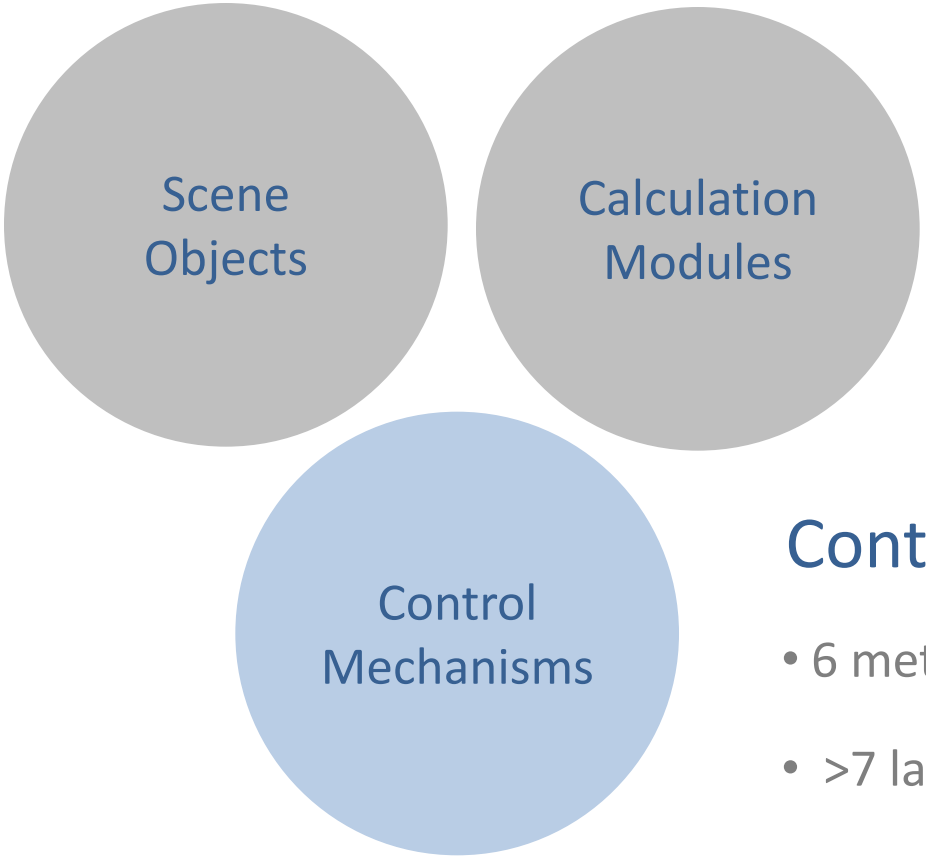
Path / Motion Planning

Supported via an OMPL plugin for V-REP

>> Play demo video1

>> Play demo video2

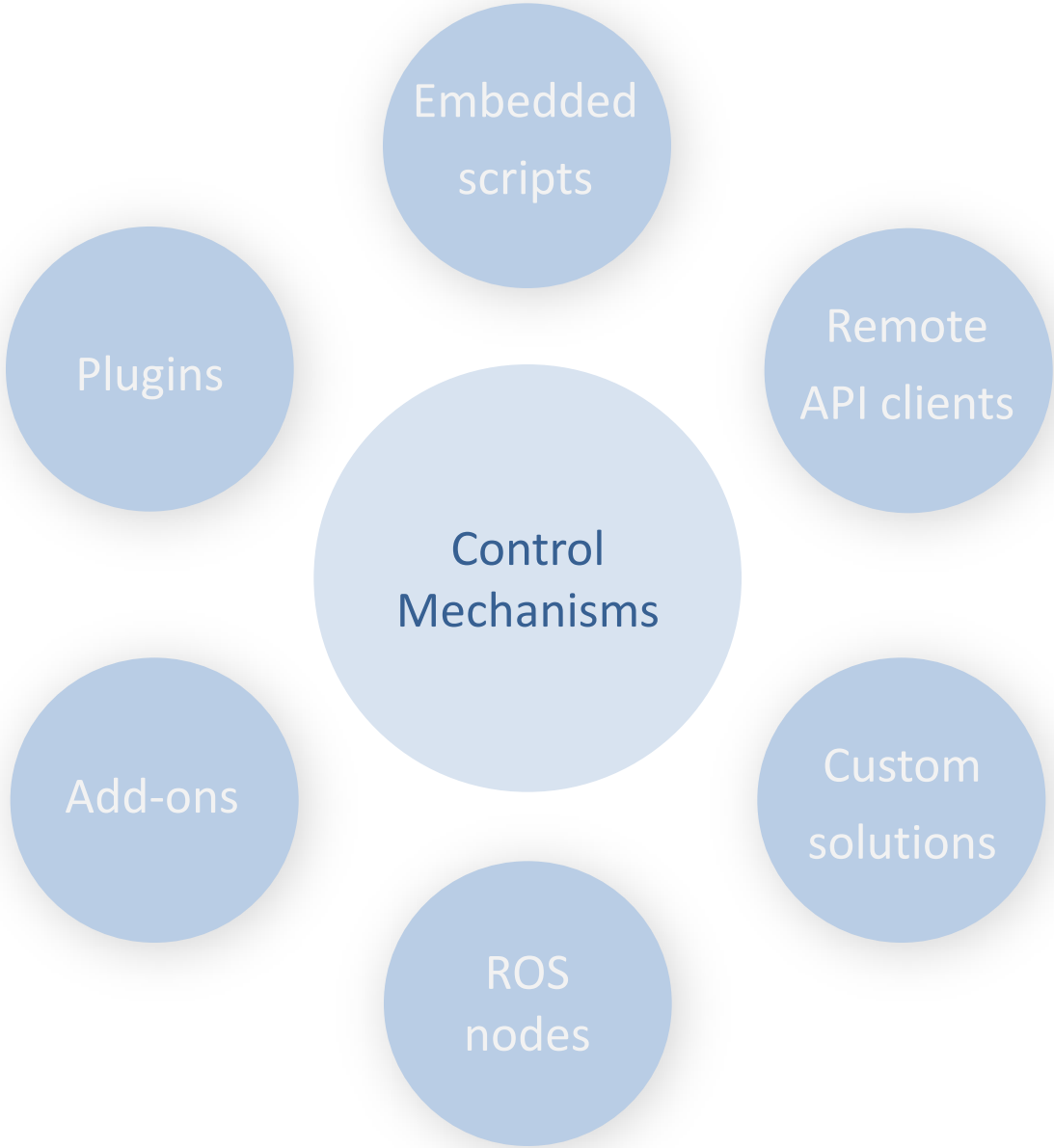
3 Central Elements



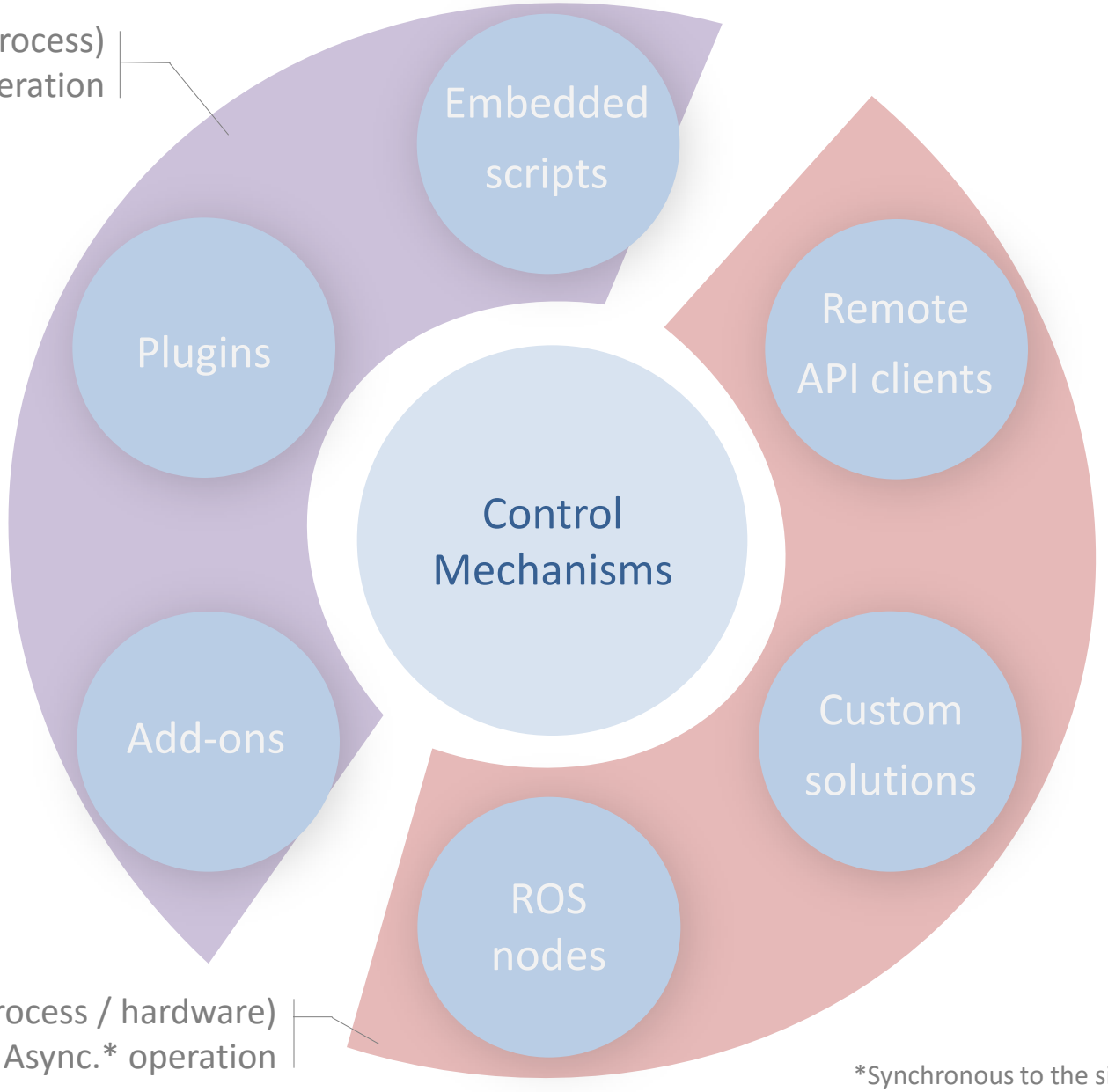
Control Mechanisms

- 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



*Synchronous to the simulation loop

Local Interfaces

Plugins

- > 500 API functions. Extendable
- C/C++ interface
- Can customize the simulator
- Can register new embedded script commands

Embedded Scripts

- > 500 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

- > 400 API functions. Extendable
- Lua interface
- Can customize the simulator
- Lightweight and easy to set-up
- Many Lua extension libraries available

Remote Interfaces

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 Interfaces

- Plugin-based
- Supports all standard messages, extendable
- Naturally duplicates the ROS C++ API

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.)

+

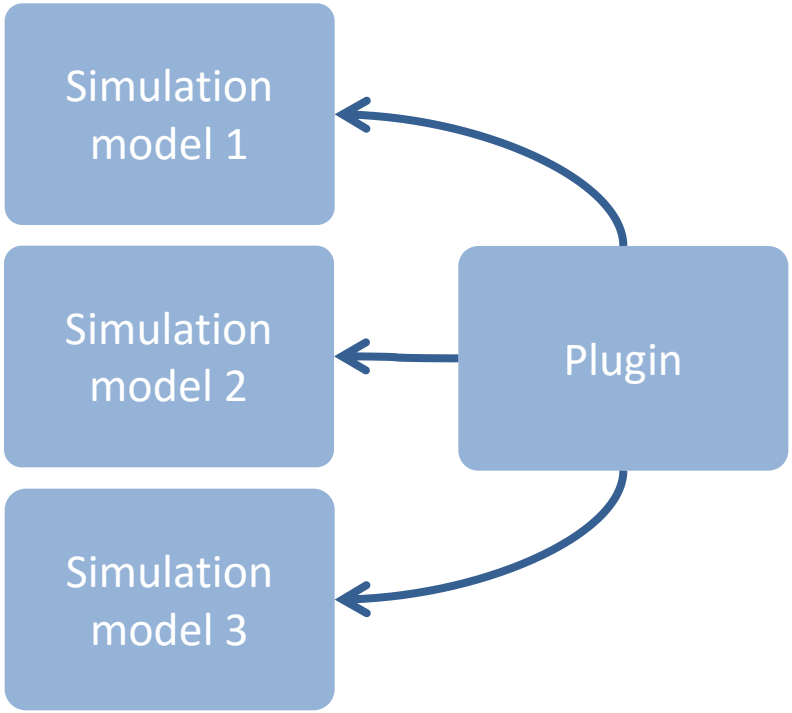
controller

1 item

Embedded Script Advantages 2/6

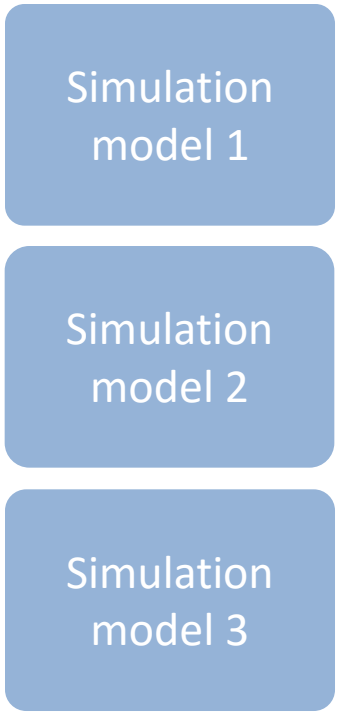
Scalability

Plugins



Plugin has to manage instances

Embedded scripts

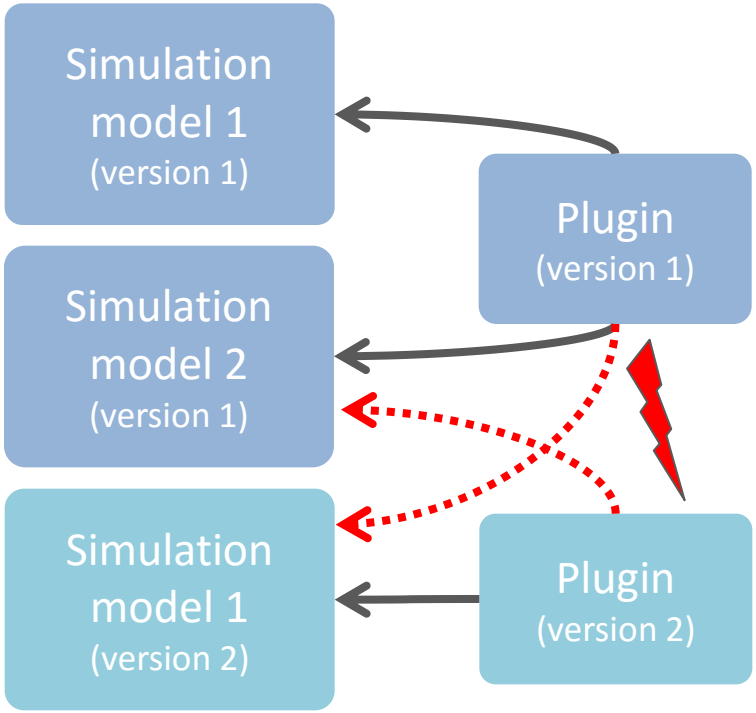


Scalability is inherent

Embedded Script Advantages 3/6

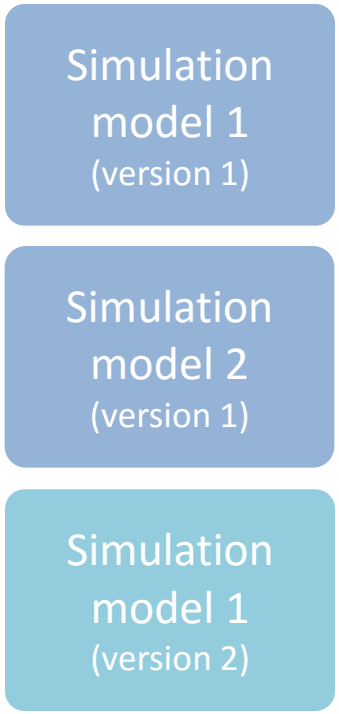
Version Conflicts

Plugins



High chances for conflicts

Embedded scripts



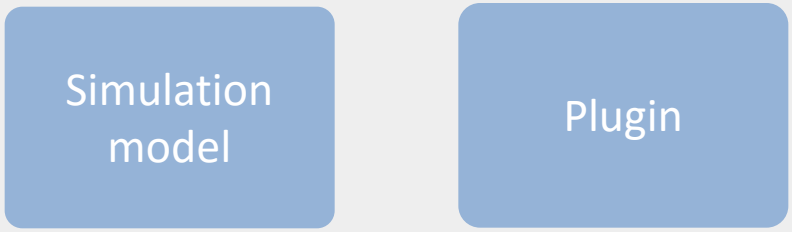
No chances for conflicts

Embedded Script Advantages 4/6

Portability

Plugins

Same OS



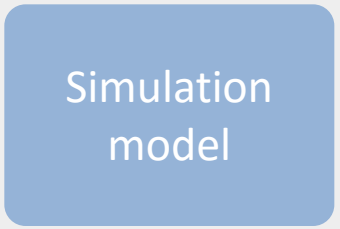
2 files

Different OS

Many files
Compilation required
OS-specific problems

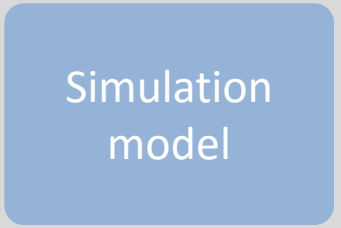
Embedded scripts

Same OS



1 file

Different OS



1 file
No compilation required
No OS-specific problems

Embedded Script Advantages 5/6

Other Considerations

Plugins	Embedded scripts
Creation, compilation and installation difficulty: High	Creation, compilation and installation difficulty: Low
Model modification difficulty: High	Model modification difficulty: Low
Maintenance over the years: OS-dependent Compiler-dependent Framework-dependant	Maintenance over the years: OS-independent Compiler-independent Framework-independant

Embedded Script Advantages 6/6

Synchronization with Simulation Loop

Plugins

Non-threaded

Control routine called at each simulation pass

Easy

Threaded

Complex synchronization mechanism required

Difficult

Embedded scripts

Non-threaded

Control routine called at each simulation pass

Easy

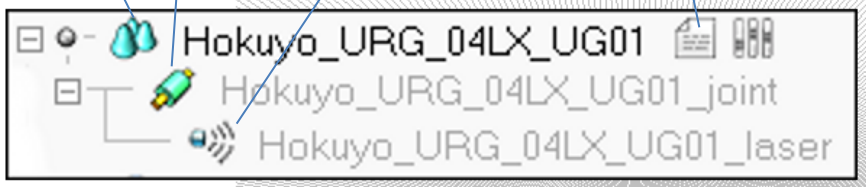
Threaded

Control routine thread can behave as a coroutine

e.g. simSwitchThread()
simSetThreadSwitchTiming(delay)
simSetThreadIsFree(isFree)
simSetThreadResumeLocation(location,order)

Easy

Embedded Scripts – Simple Example



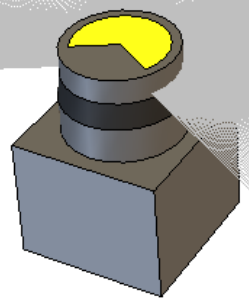
Shape

Joint

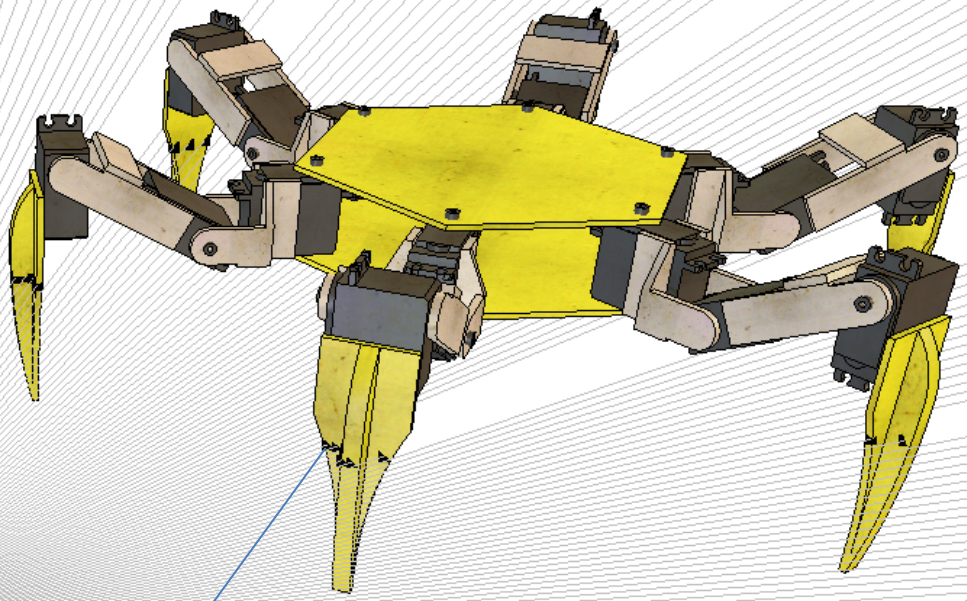
Proximity sensor

Embedded script

Hokuyo model hierarchial representation



Hokuyo simulation model



Detected points

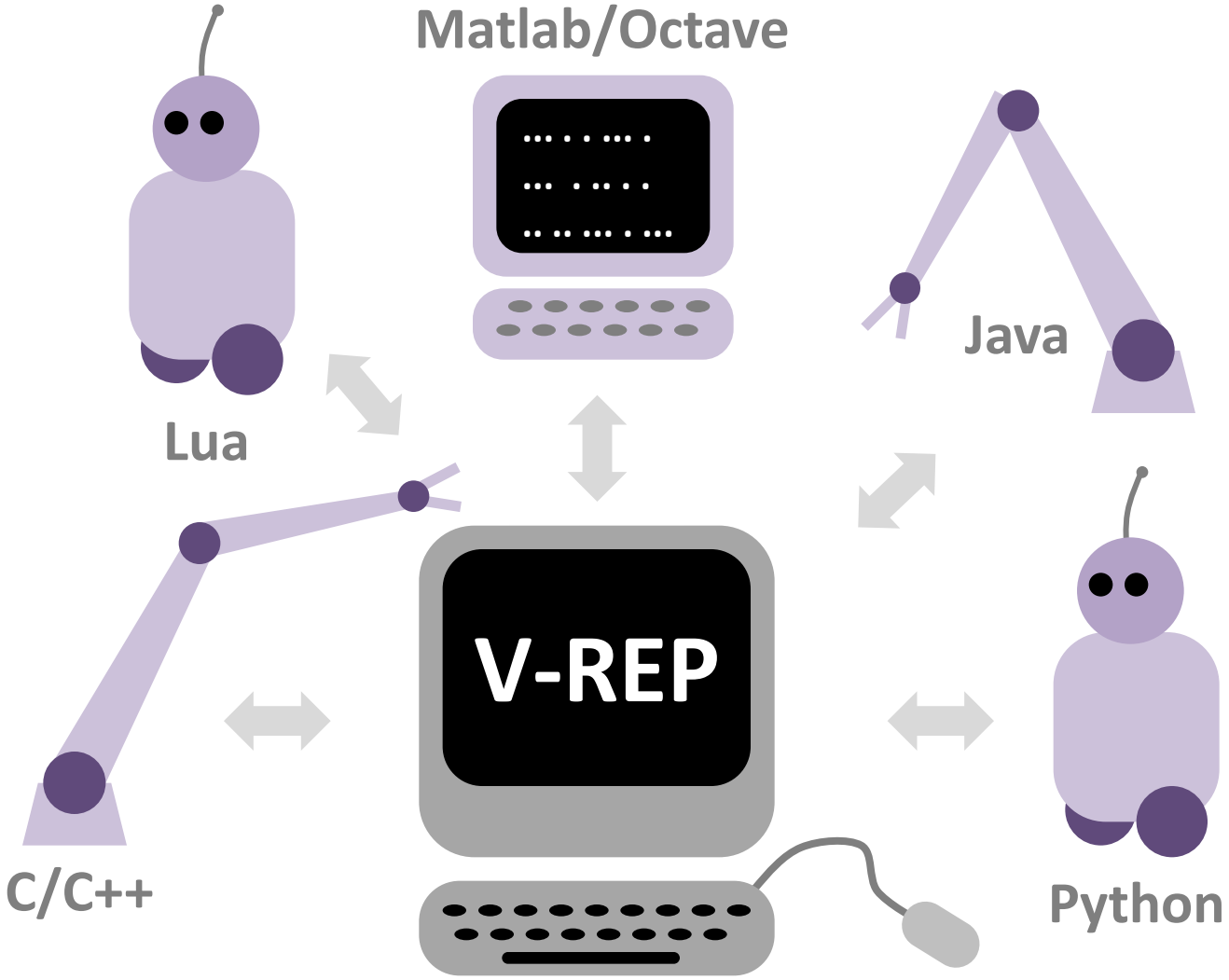
Hexapod simulation model

Distance sampling rays

>> Play demo video

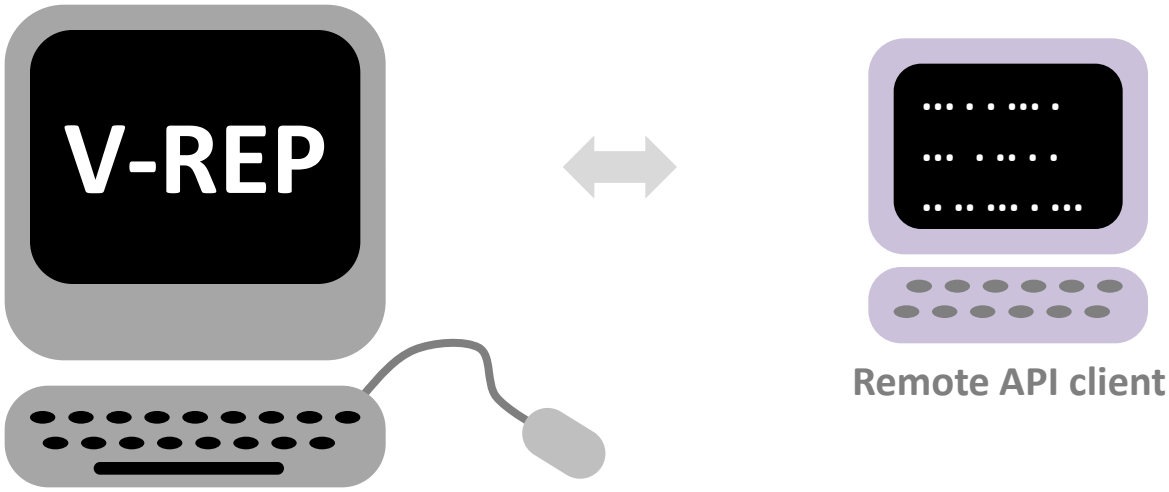
Remote API Advantages 1/2

Runs on any hardware, lightweight, several languages



Remote API Advantages 2/2

Easy to use, almost like a *regular API*



Remote API function

Regular arguments

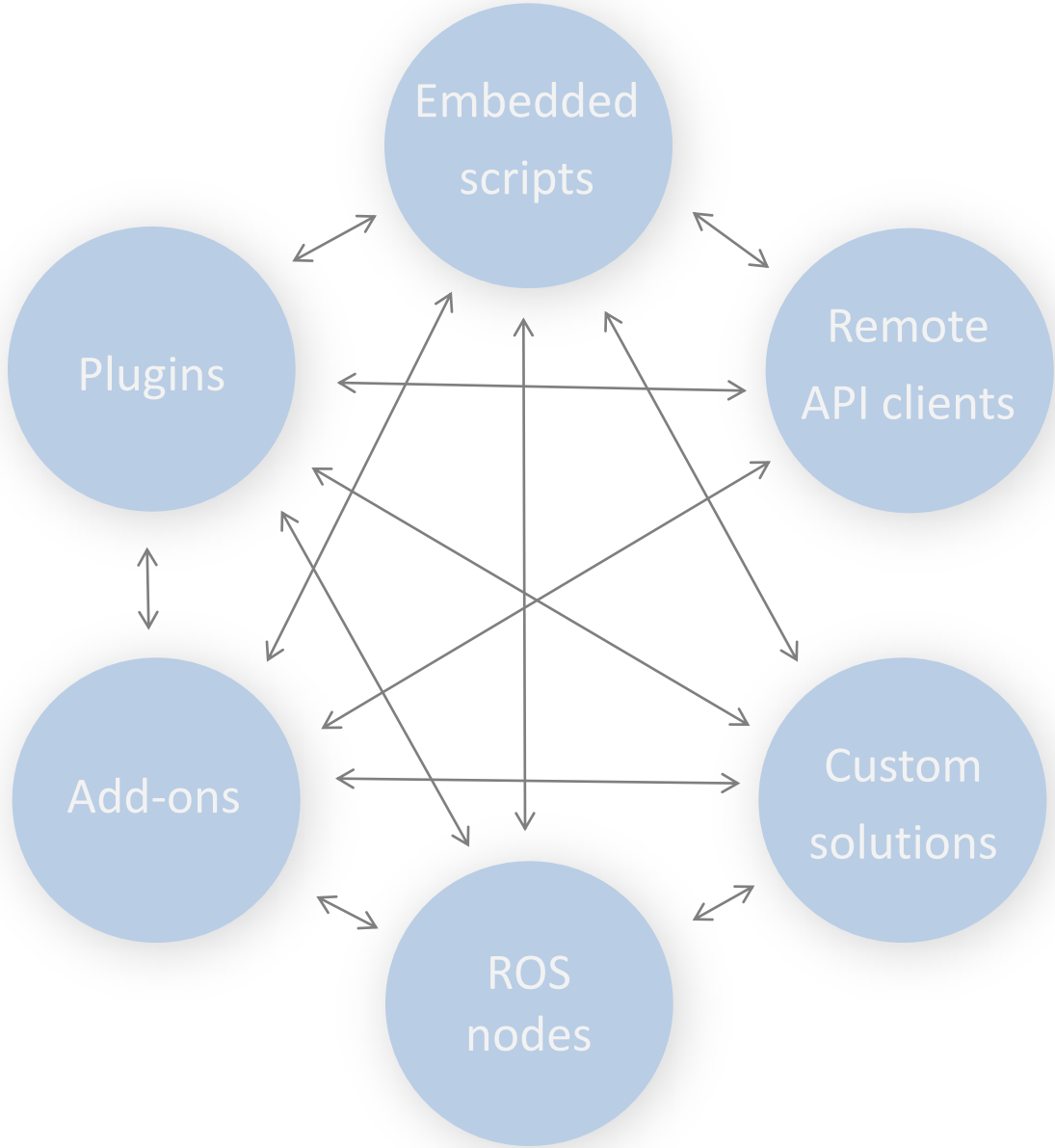
```
int returnCode=simxGetJointPosition(jointHandle,&position,operationMode);
```

- 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_blocking
- simx_opmode_streaming
- simx_opmode_discontinue
- simx_opmode_buffer
- etc.

Collaborative Control Mechanisms



Example of Collaborative Mechanism 1 / 3

Remotely call a script function

Remote API client (e.g. C++)

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

Calls

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

Embedded script

```
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
```

Example of Collaborative Mechanism 2 / 3

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

// Initialization phase of plugin: register new script commands:
simRegisterScriptCallbackFunction(..., SCRIPT_DO_SOME_MAGIC_CALLBACK);
```

Plugin

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

```
-- Left motor speed subscriber callback
function LVel_cb(msg)
    simSetJointTargetVelocity(leftMotor,msg.data)
end

-- Right motor speed subscriber callback
function RVel_cb(msg)
    simSetJointTargetVelocity(rightMotor,msg.data)
end

-- Initialization
if (sim_call_type==sim_childscriptcall_initialization) then
    ...
    pub=simExtRosInterface_advertise('/sensorData','std_msgs/Bool')
    subL=simExtRosInterface_subscribe('/leftVel','std_msgs/Float32','LVel_cb')
    subR=simExtRosInterface_subscribe('/rightVel','std_msgs/Float32','RVel_cb')
end

-- Actuation phase, once per simulation step
if (sim_call_type==sim_childscriptcall_actuation) then
    local result=simReadProximitySensor(noseSensor)
    local detectionTrigger={}
    detectionTrigger['data']=result>0
    simExtRosInterface_publish(pub,detectionTrigger)
    simExtRosInterface_sendTransform(...)
end
```

Embedded script

from ROS

from ROS

Subscriber callbacks

Advertise publisher and subscribers

to ROS

Publish sensor data and send transform

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)	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

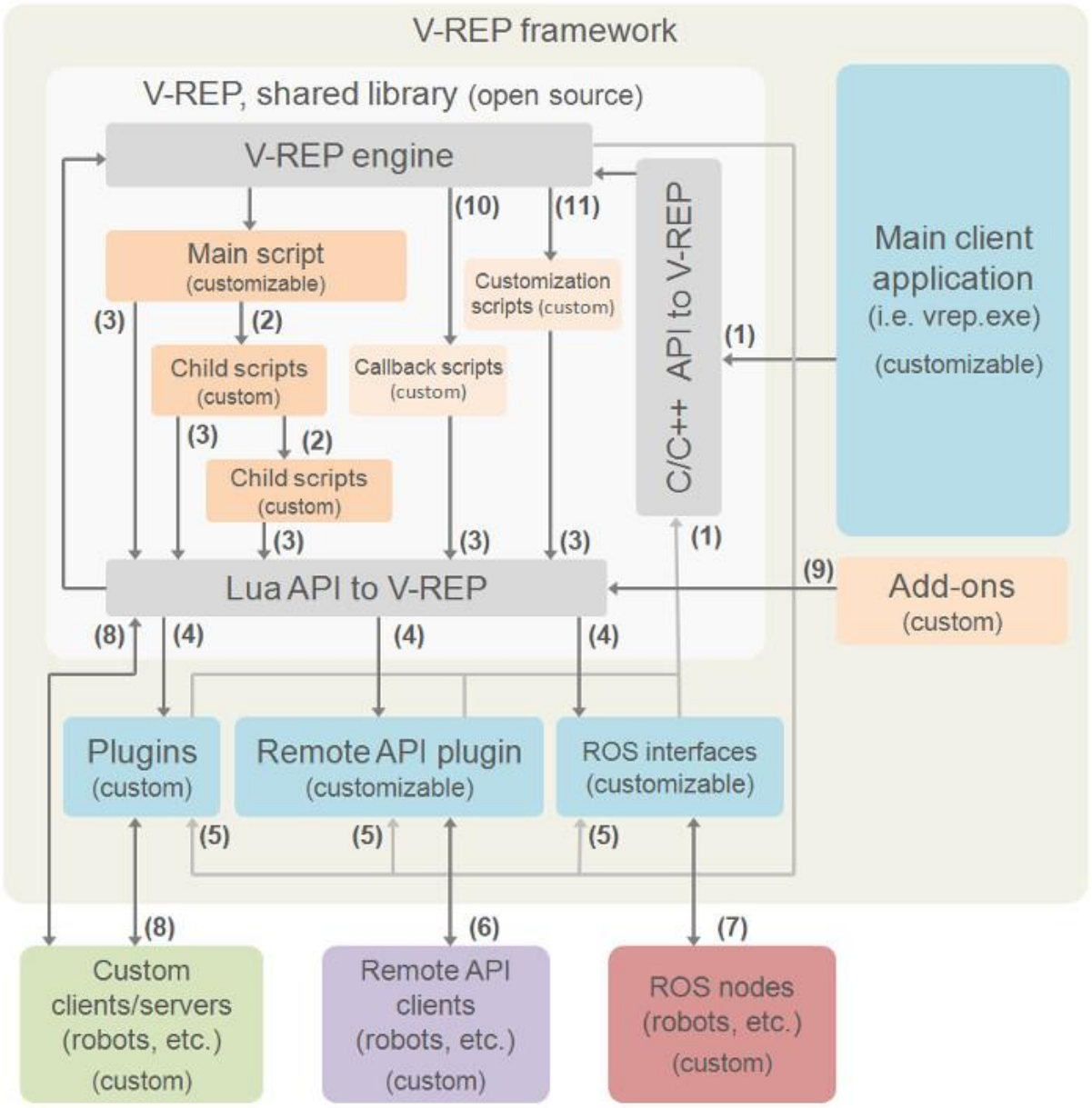
[1] Depends on what ROS currently supports

[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

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.)

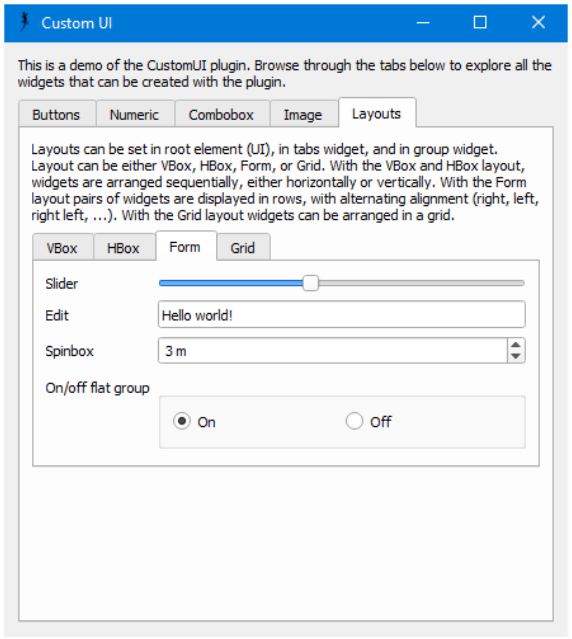
(9) Add-on calls to Lua API

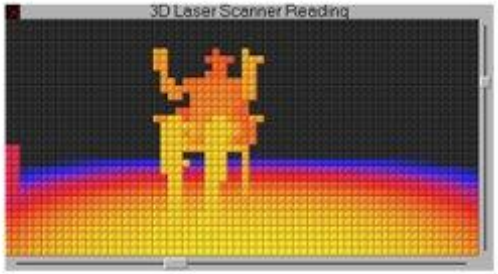
(10) Script callback calls

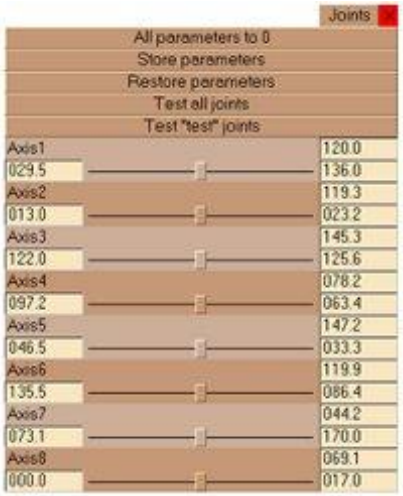
Other Feature: Custom User Interfaces

Custom User Interfaces

- OpenGl-based or
- Qt-based







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
- Mesh decimation

More Features

- 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.

V-REP Overview

State-of-the-art distributed control architecture

- Embedded scripts
- Remote API
- 2 ROS interfaces

Extremely fine-grained and large amount of features

- >500 different API function
- 14 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 interfaces)
- Languages (C/C++, Java, Python, Lua, Matlab, Octave, Lua, Urbi)
- Physics engines (Bullet, ODE, Vortex, Newton)
- 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.

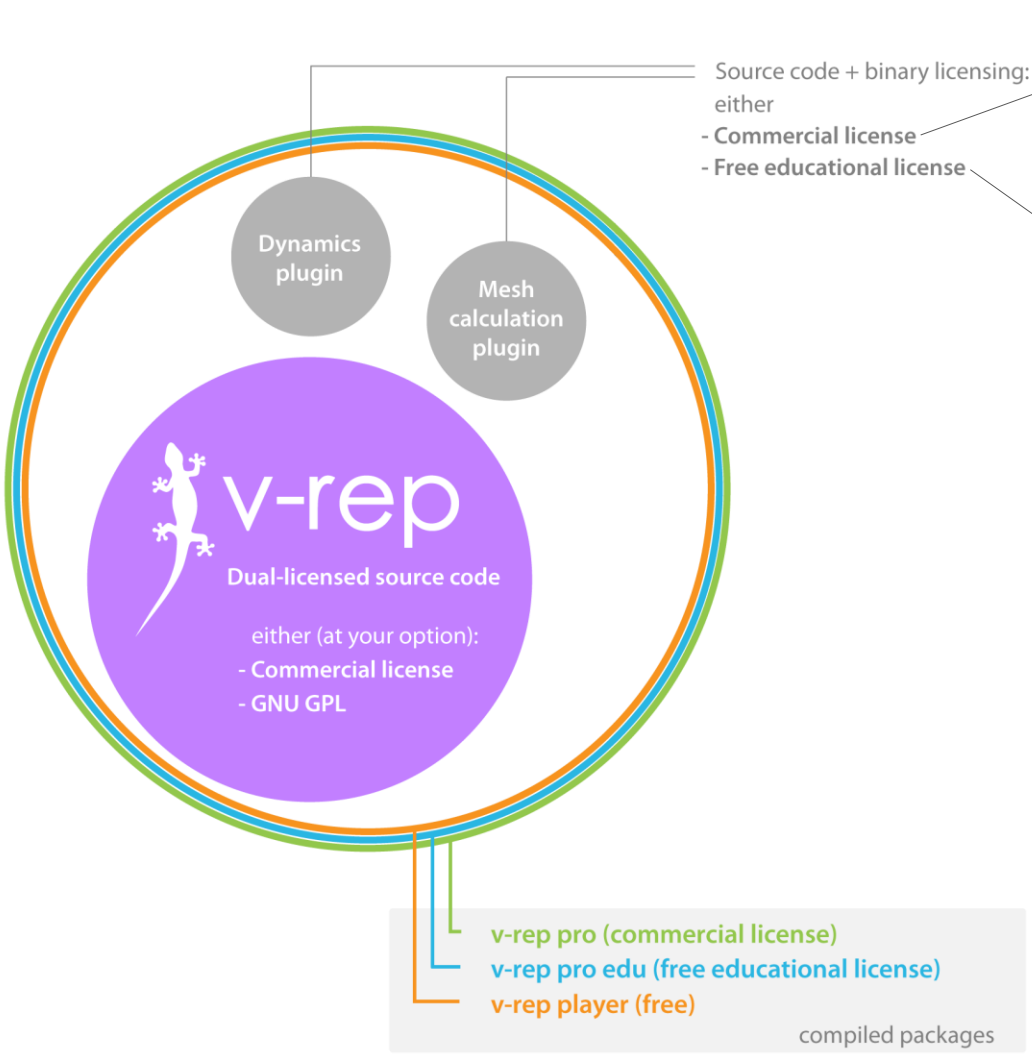
V-REP PRO

- 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



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

V-REP contact: info_at_coppeliarobotics_dot_com