Fundamentals of Robotics

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Virtual robot experimentation platform
V-REP

Class 1 - Introduction

DEPARTAMENTO DE
ELECTRONICA

Por: Juan Pablo Vásconez

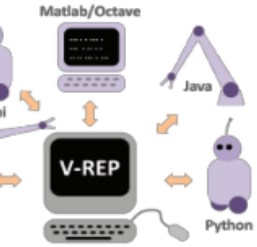
Valparaíso, Sep 2021

V-REP

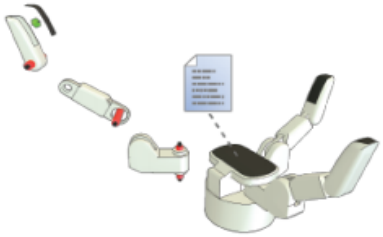
Virtual Robot Experimentation Platform (V-Rep) is a general purpose robot simulator developed by Coppelia Robotics

V-rep is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution

Controllers can be written in C/C++, Python, Java, Lua, Matlab, Octave or Urbi.



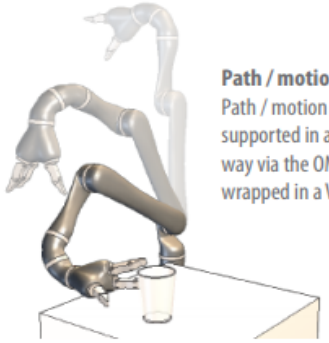
Remote API: More than 100 embeddable V-REP functions: control a simulation or the simulator itself remotely (e.g. from a real robot or another PC). Easy to use, supports sync. or async. operation, is optimized for heavy data transfer and minimizes communication lag. Six languages are supported



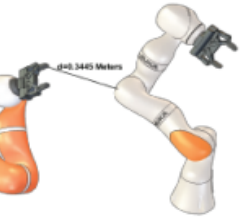
Building block concept: Anything - from sensors or actuators, to whole robotic systems - can be built within V-REP by combining basic objects and linking various functionality via embedded scripts. Each scene object can have its own embedded script

Forward/inverse kinematics: Full forward/inverse kinematics calculations module for any type of mechanism (branched, closed, redundant, containing nested loops, etc.). Can be embedded

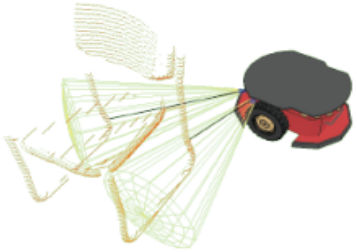
Dynamics/Physics: Fast and customizable dynamics calculations to simulate real-world physics and object interactions (collision response, grasping, etc.). Four engines are supported: Bullet, ODE, Newton and Vortex



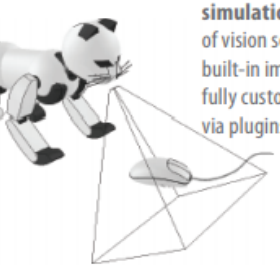
Path / motion planning: Path / motion planning is supported in a very flexible way via the OMPL library wrapped in a V-REP plugin



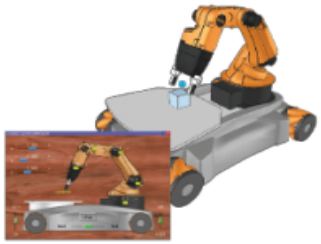
Collision detection and distance calculation: Fast interference checking and minimum distance calculation between any mesh, octree or point cloud



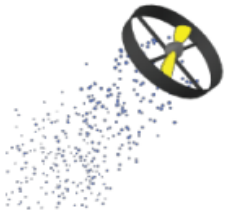
Proximity sensor simulation: Powerful proximity sensor simulation, fully customizable. Performs an exact minimum distance calculation within a custom detection volume (more realistic than discrete detection rays). Operates on meshes, octrees and point clouds



Vision sensor simulation: Simulation of vision sensors with built-in image processing, fully customizable (e.g. via plugins)



Custom user interfaces: Unlimited number of fully customizable user interface elements. Windows style or OpenGL style.



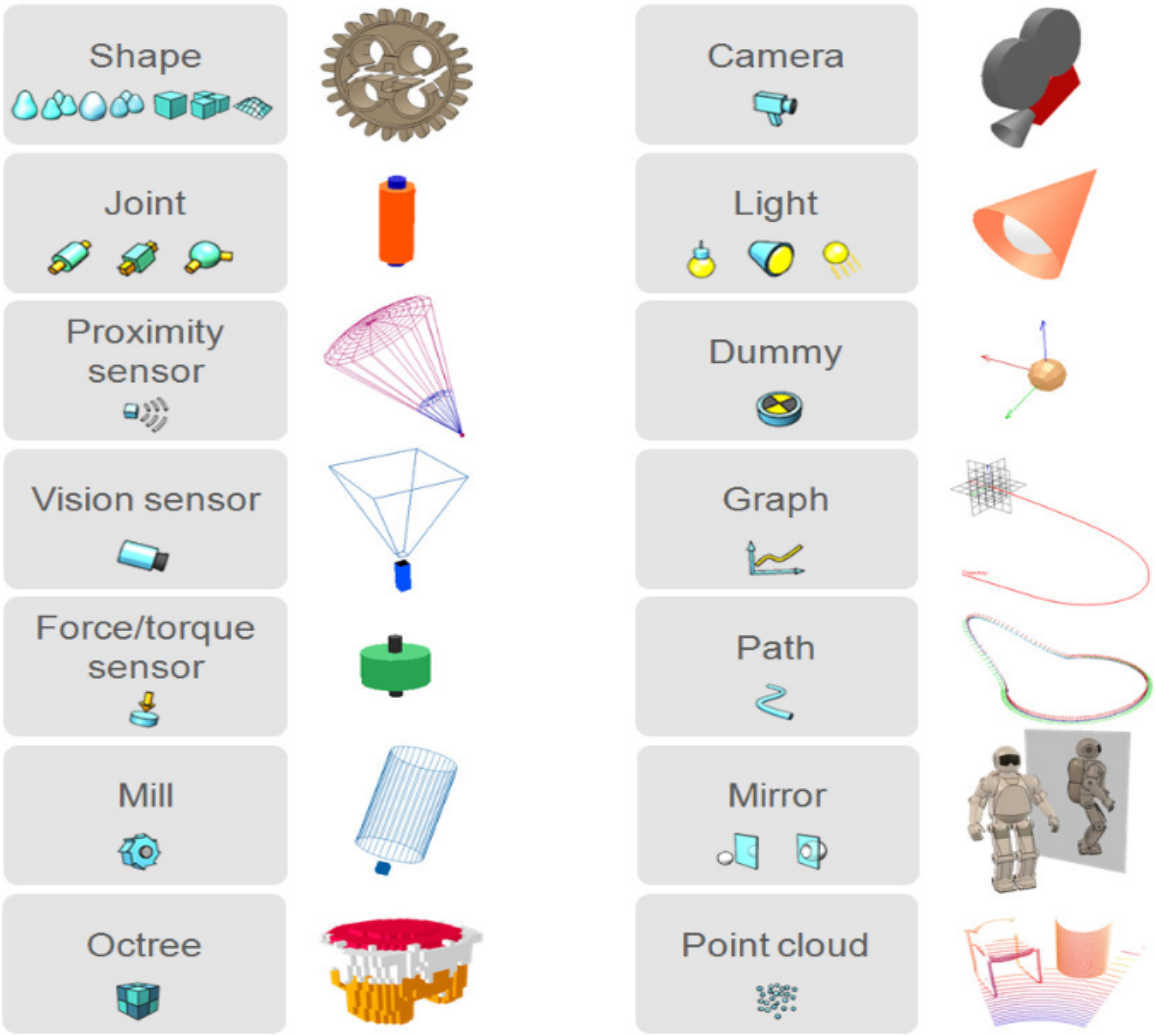
Many more features: e.g. headless mode, data recording and visualization, mesh edit modes, RRS-1 specification support, convex decomposition functions, Reflexxes Motion Library, support for haptic devices, video recorder, simulation of paint or welding seams, full source code available, etc.

Scene objects

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Scene objects

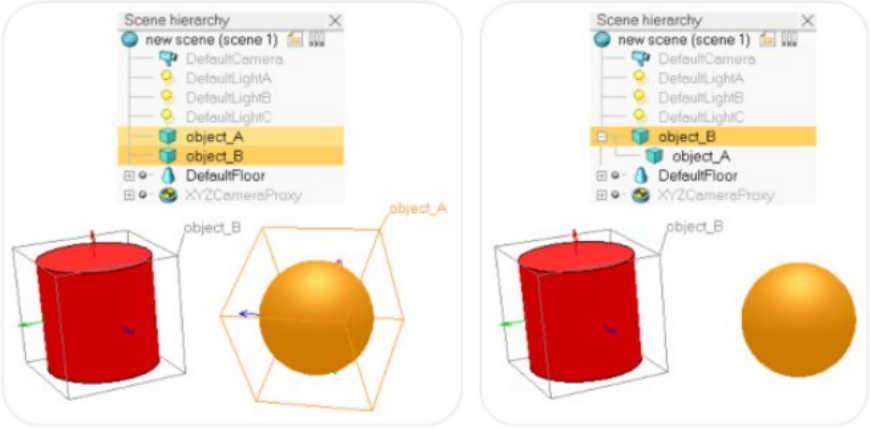
The main elements in V-REP that are used for building a simulation scene are scene objects (objects in short). Objects are visible in the scene hierarchy and in the scene view. In the scene view, objects have a three dimensional representation as illustrated in following figure:



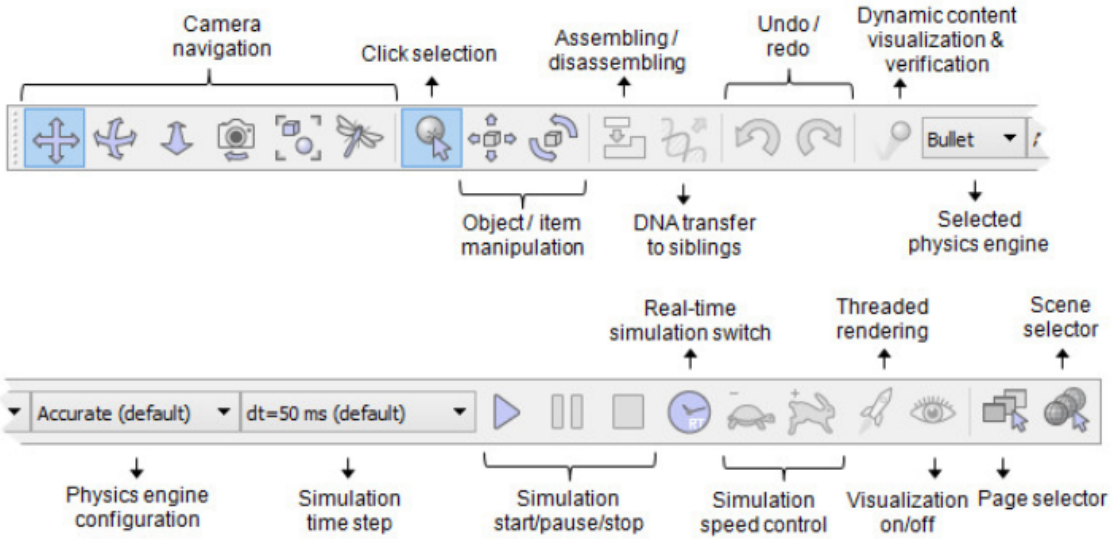
Objects properties

- Some of above objects can have special properties allowing other objects or calculation modules to interact with them. Objects can be:

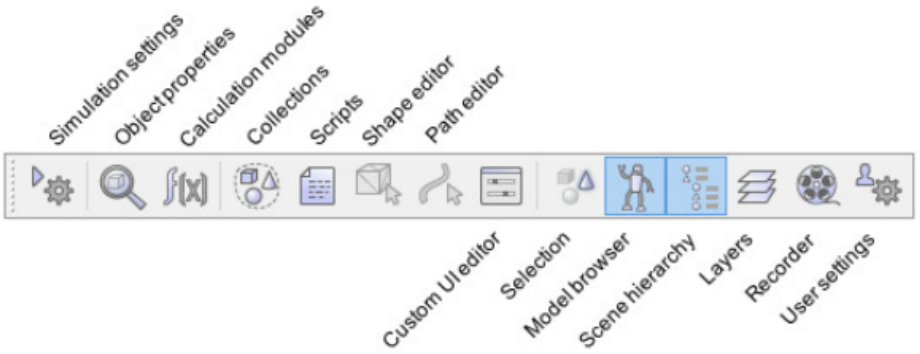
•Measurable: measurable objects can have the minimum distance between them and other measurable objects calculated.
•Detectable: detectable objects can be detected by proximity sensors.
•Cuttable: cuttable objects can be cut by mills.
•Collidable: collidable objects can be tested for collision against other collidable objects.
•Renderable: renderable objects can be seen or detected by vision sensors.
•Viewable: viewable objects can be looked through, looked at, or their image content can be visualized in views.

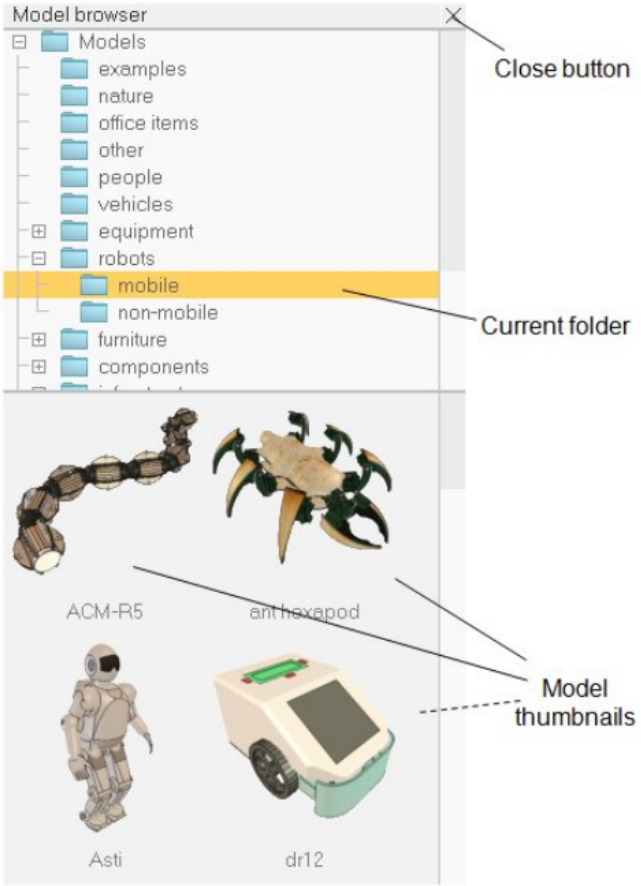


User interface

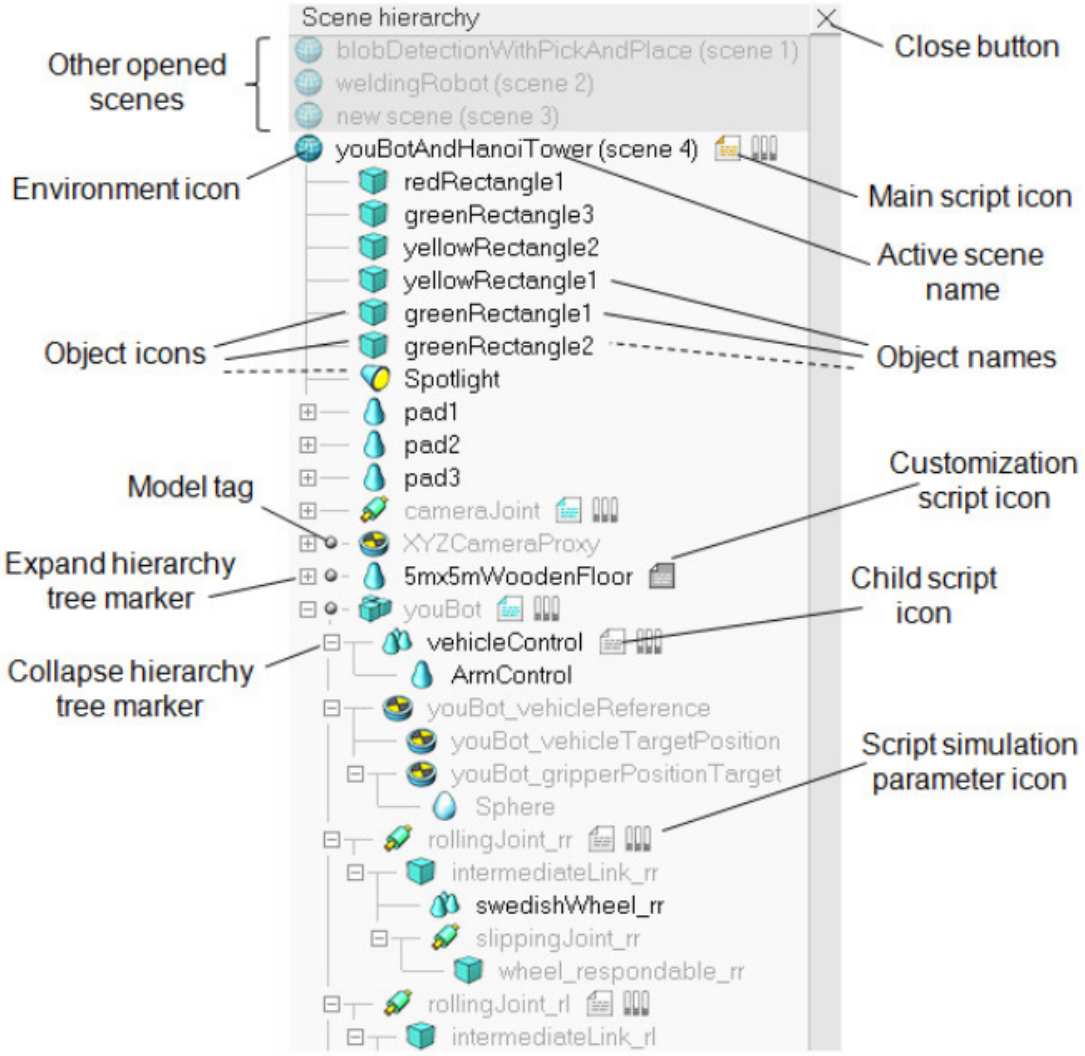


[Toolbar 1]





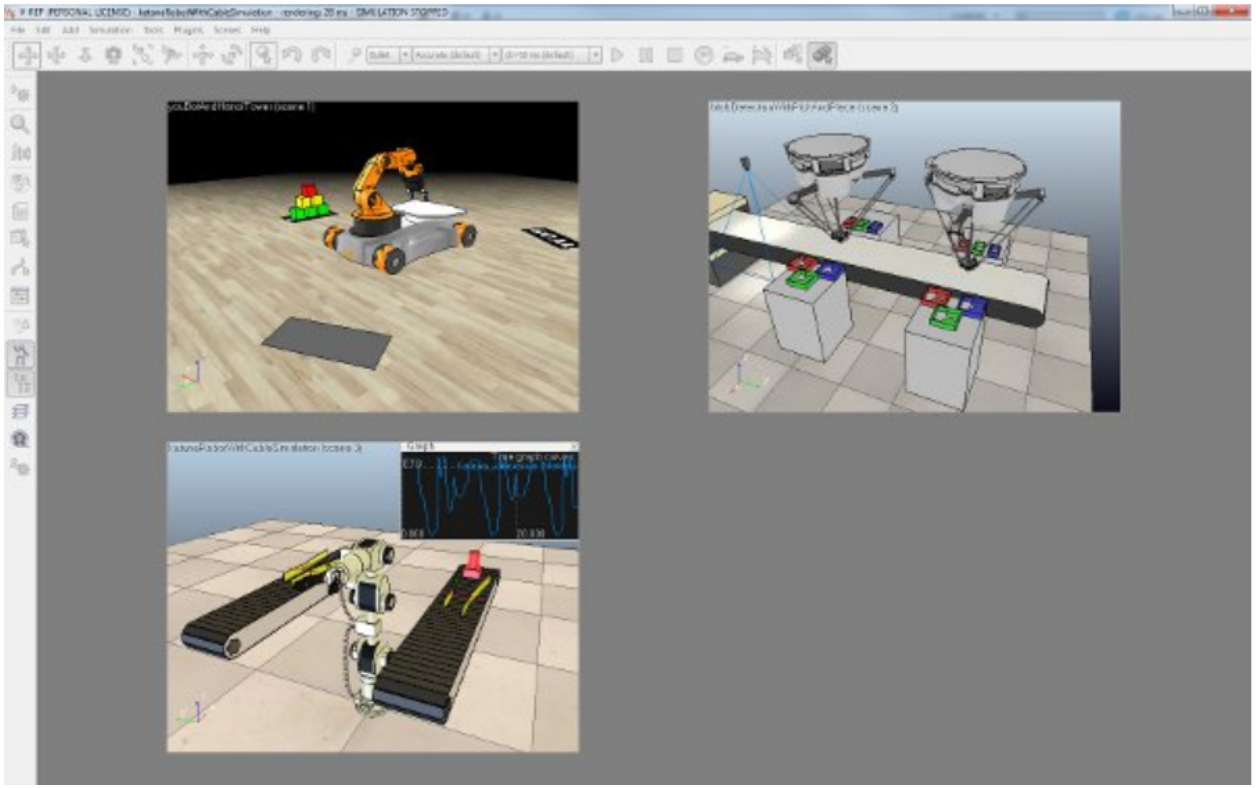
Scene hierarchy



Scenes

Compared to models, a scene can contain exactly the same type of elements, but additionally also includes following elements, specific to scenes:

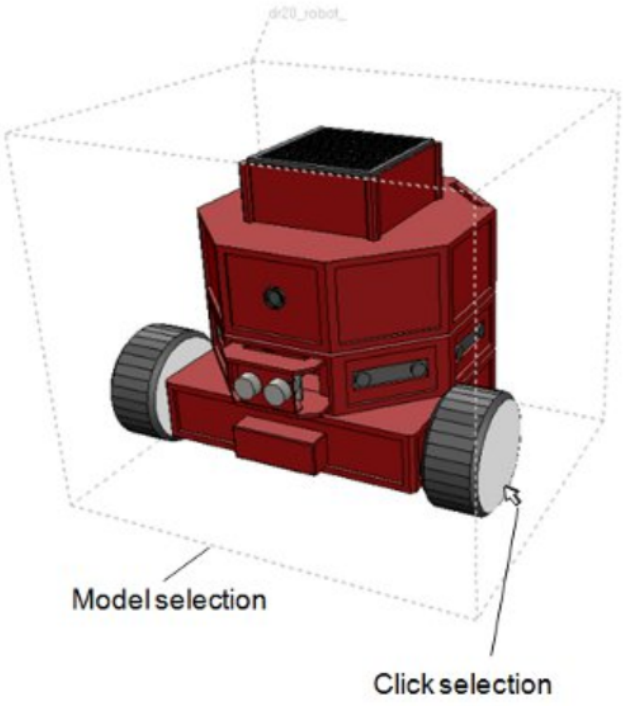
- The environment
- The main script
- Pages and views

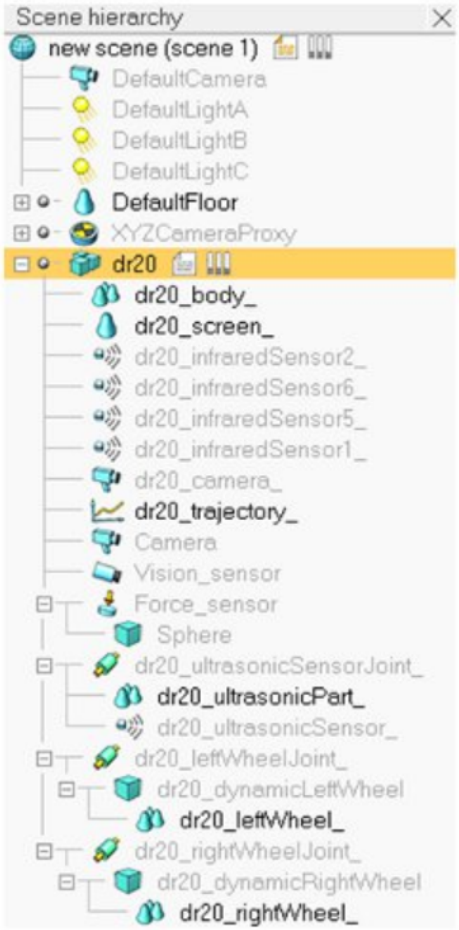


Models

A model is a sub-element of a scene. A model by itself cannot exist, except in a file ("*.ttm"-file type), nor can it be simulated by itself. A model has to be contained in a scene in order to be operational.

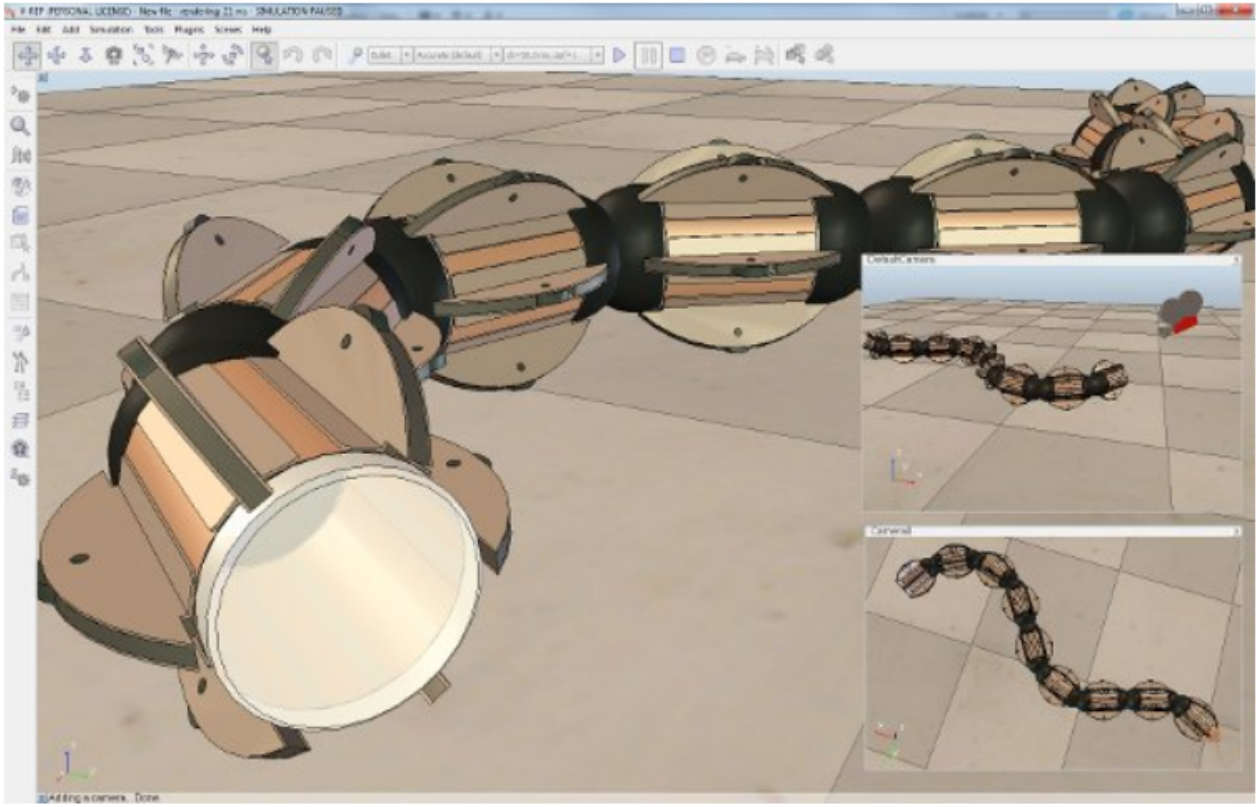
Models are defined by a selection of scene objects built on a same hierarchy tree, where the base of the tree has to be an object flagged as **object is model base**.





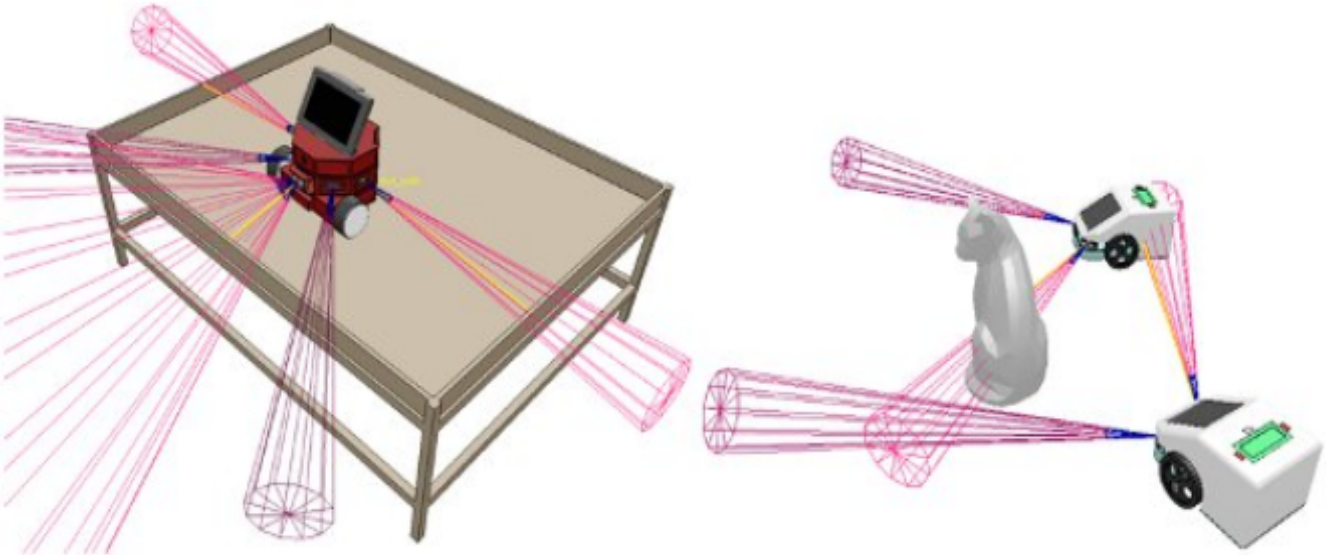
Cameras

Cameras are viewable objects, which means that you can *look through* them and display a view of what they are *looking at*. You can have as many cameras as needed in your scene, each one of them offering a different view of the scene. Following shows an example of a scene containing several cameras:



Proximity sensors

V-REP offers a very powerful and efficient way to simulate proximity sensors. The user can model almost any type of proximity sensor, from ultrasonic to infrared, and so on. The scene objects that allow for this functionality are proximity sensors (which are different from vision sensors), which can detect detectable entities. Following figures illustrates simulations using proximity sensors:

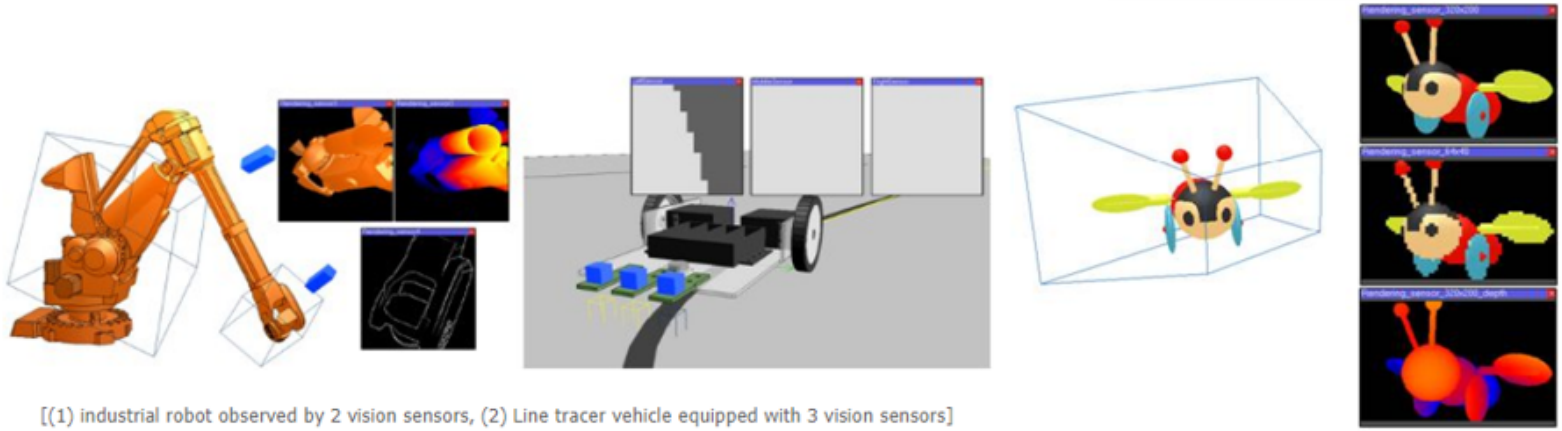


[Mobile robots using proximity sensors]

See collision detection demo

Vision sensors

V-REP offers, next to proximity sensors, another type of sensors: vision sensors. Vision sensors, which are viewable objects, operate in a very similar way as camera objects: they will render the objects that are in their field of view and trigger detection if specified thresholds are over- or under-shot. Vision sensors, which can detect renderable entities, should be used over proximity sensors mainly when color, light or structure plays a role in the detection process (e.g. infrared sensors, or, more generally, sensors sensible to light (cameras, etc.)). However, depending on the graphic card the application is running on, or on the complexity of the scene objects, vision sensors might be a little bit slower than proximity sensors. Following illustrates applications using vision sensors:



[(1) industrial robot observed by 2 vision sensors, (2) Line tracer vehicle equipped with 3 vision sensors]

Make sure not to mix-up vision sensors with cameras. Following are the main differences:

- A vision sensor has a fixed resolution. A camera has no specific resolution (i.e. it adjusts automatically to the view size).
- A vision sensor's image content can be accessed via the API, and image processing filters are available. A camera's image content is not directly available via the API (but via a callback mechanism), and image processing not directly supported.
- A vision sensor generally requires more CPU time and operates slower than cameras.
- A vision sensor can only display renderable objects. A camera can display all object types.

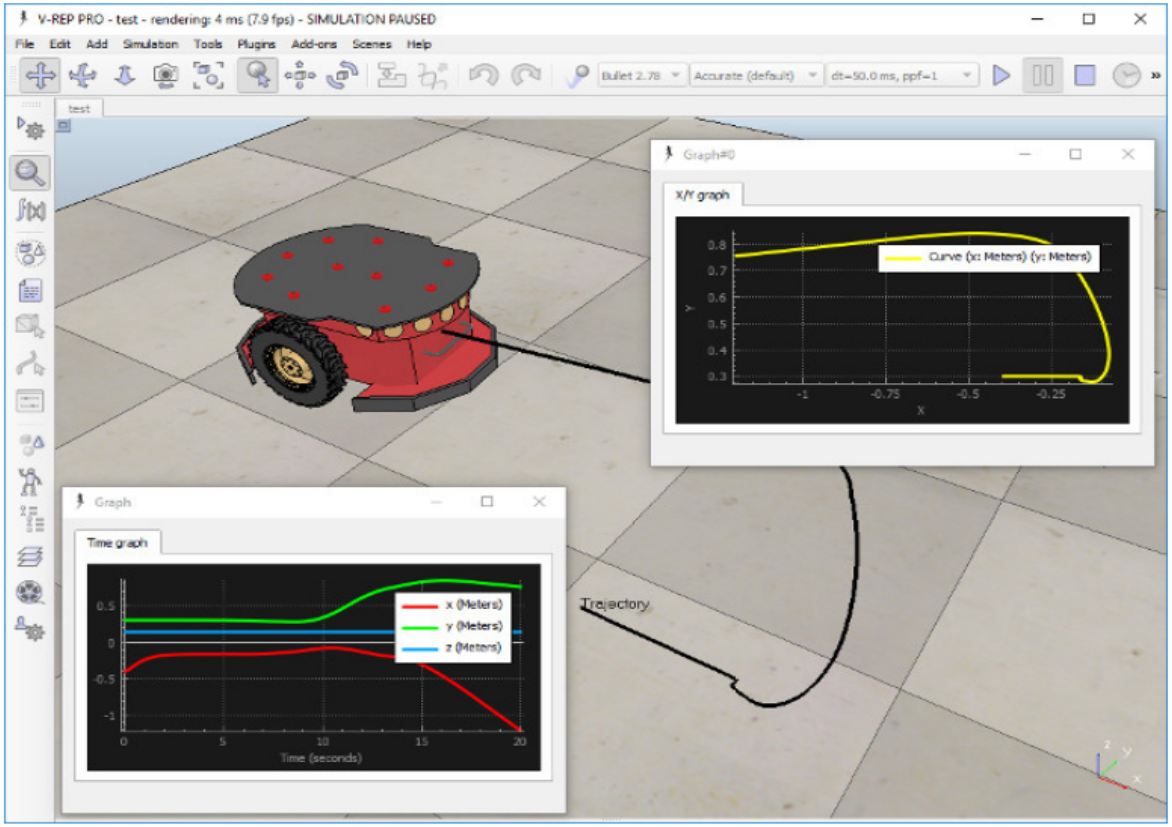
See image processing demo

Graphs

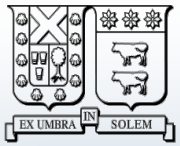
Graphs are objects that can be used to record, visualize or export data from a simulation. They are very powerful and flexible. The user can select from a multitude of data types applied to specific objects to record. Data is recorded as a data stream (sequential list of data values) that can be visualized in three different ways:

- **Time graphs**: a time graph of a single data stream (time vs data stream).
- **X/Y graphs**: a graph that combines two data streams (data stream A vs data stream B).
- **3D curves**: a 3D graph that combines three data streams in time (time vs (data stream A, data stream B, data stream C)).

Following figure illustrates the graph functionality:



Fundamentals of Robotics



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Virtual robot experimentation platform
V-REP

Class 2 – Features and sensors (part 1)

DEPARTAMENTO DE
ELECTRONICA

Valparaíso, Oct 2021

Collision detection

V-REP can detect collisions between two collidable entities in a very flexible way. The calculation is an exact interference calculation. The collision detection module will only detect collisions; it does however not directly react to them (for collision response, refer to the dynamics module). Following figure illustrates the collision detection functionality:

[(1) Simple collision detection between two manipulators, (2) Exhaustive collision detection between two shapes]

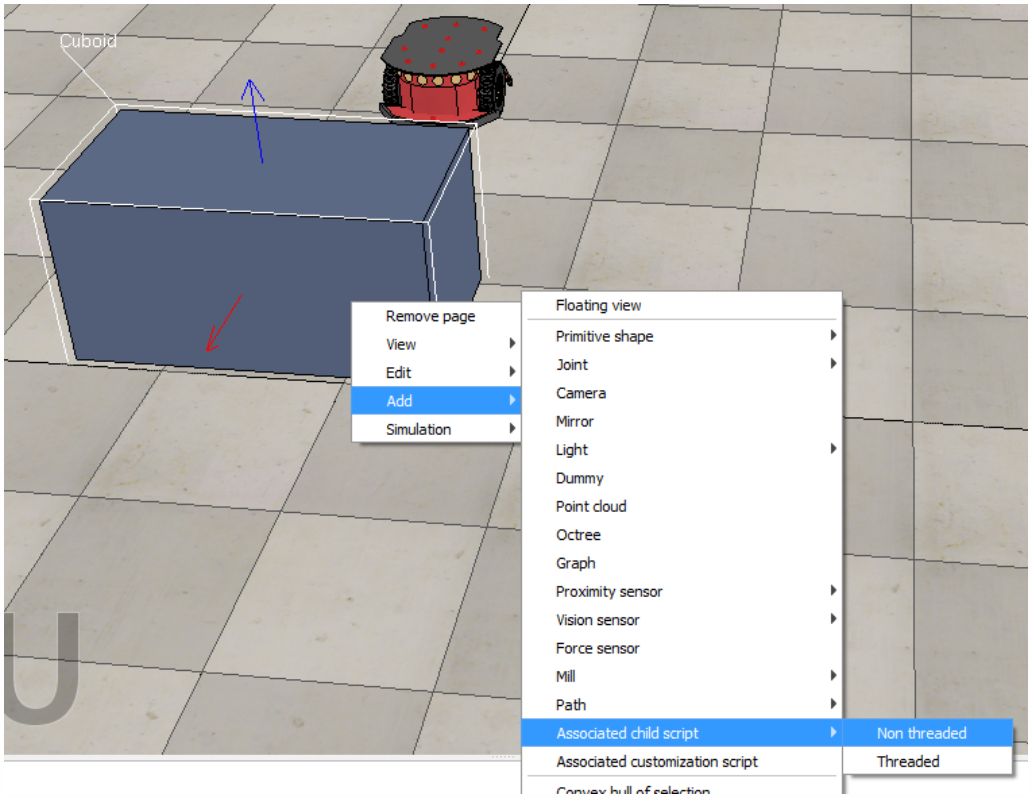
The collision detection module allows registering collision objects which are collidable entity-pairs (collider entity and collidee entity). During simulation, the collision state of each registered collision object can then be visualized with a different coloring, or recorded in a graph object. Refer to graphs and graph data stream types for more information about how to record collision states. Registered collision objects are automatically duplicated (copied) if their two composing entities are duplicated simultaneously in a copy-paste operation.

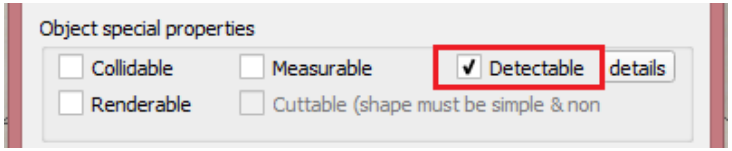
The collision detection dialog is part of the calculation module properties dialog, which is located at [Menu bar --> Tools --> Calculation module properties]. You can also open the dialog with a click on its toolbar button:

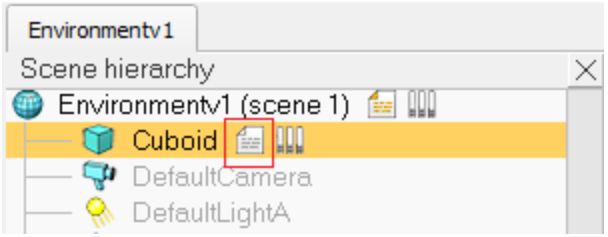
[Calculation module properties toolbar button]

See collision det demo

Preparing V-REP object



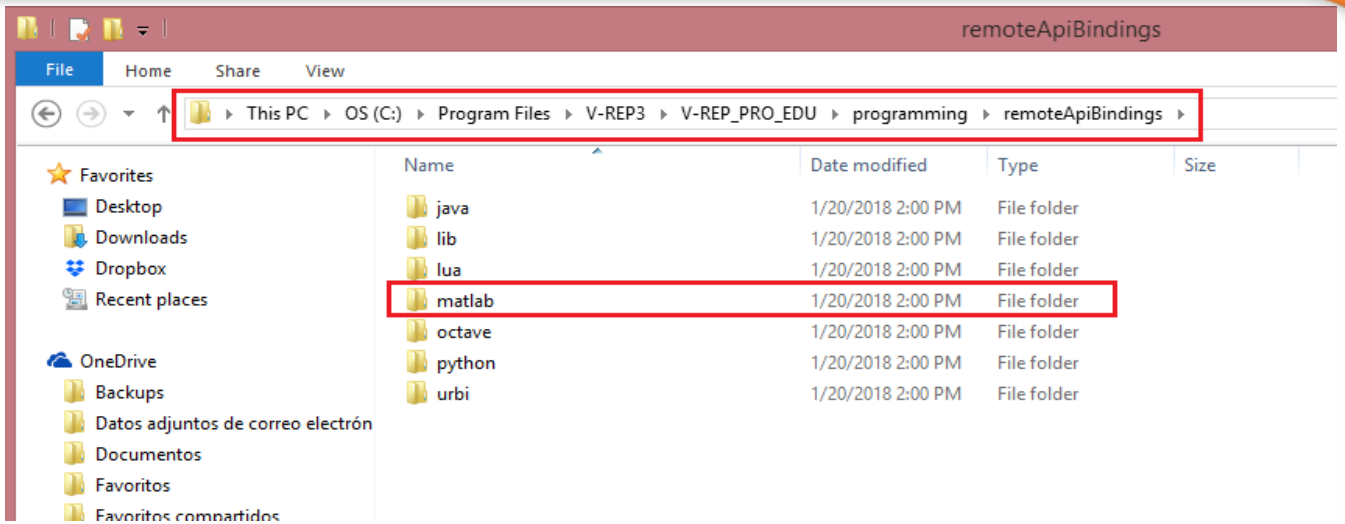


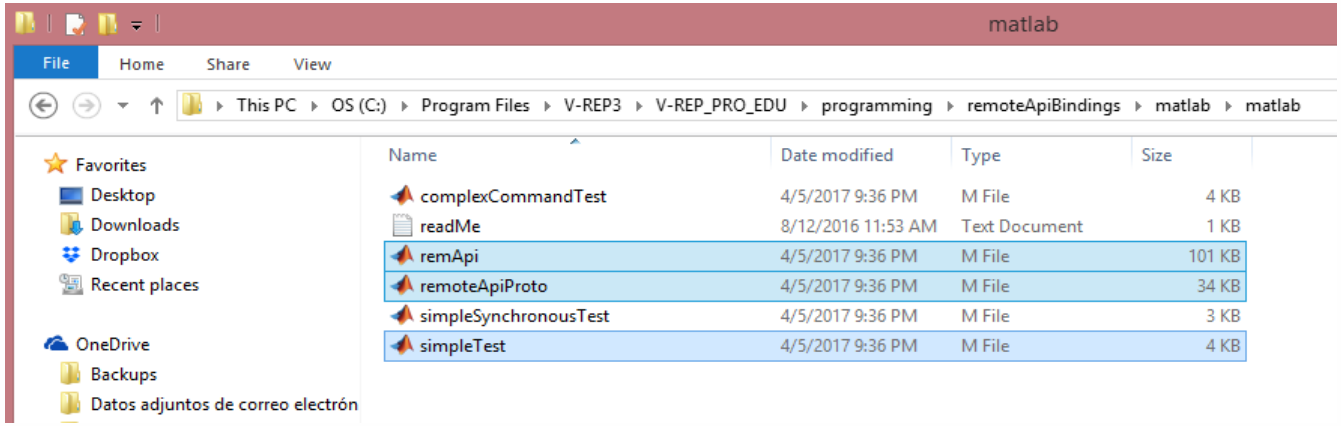


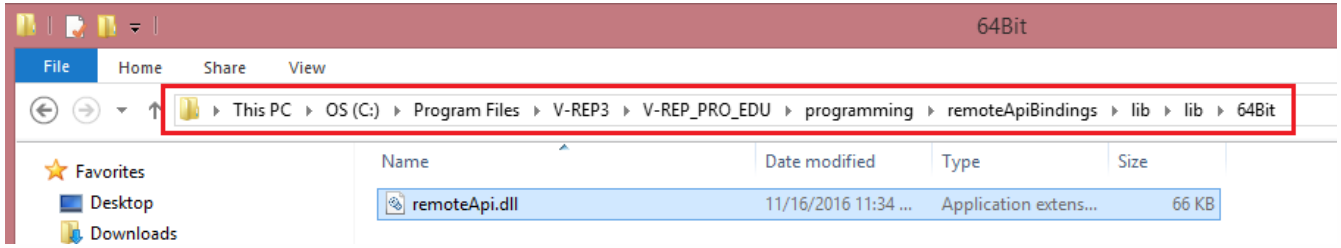
Non-threaded child script (tree_001_Mesh_5)

```
1  -- DO NOT WRITE CODE OUTSIDE OF THE if-then-end SECTIONS BELOW!! (unless the code is a funct
2
3  if (sim_call_type==sim_childscriptcall_initialization) then
4      simExtRemoteApiStart(19999)
5      -- Put some initialization code here
6
7      -- Make sure you read the section on "Accessing general-type objects programmatically"
8      -- For instance, if you wish to retrieve the handle of a scene object, use following ins
```

Preparing Matlab Files



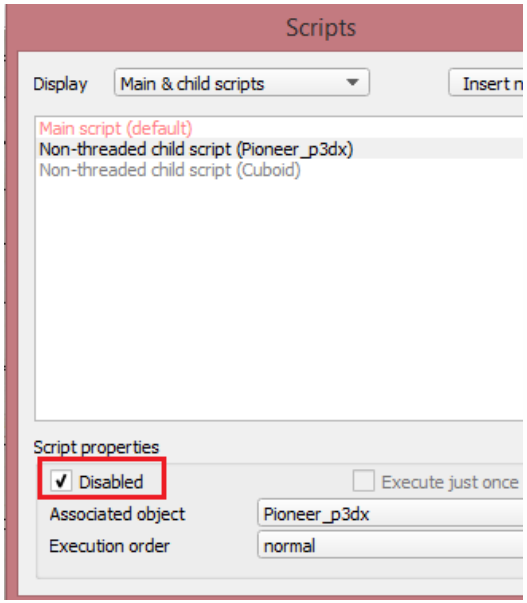


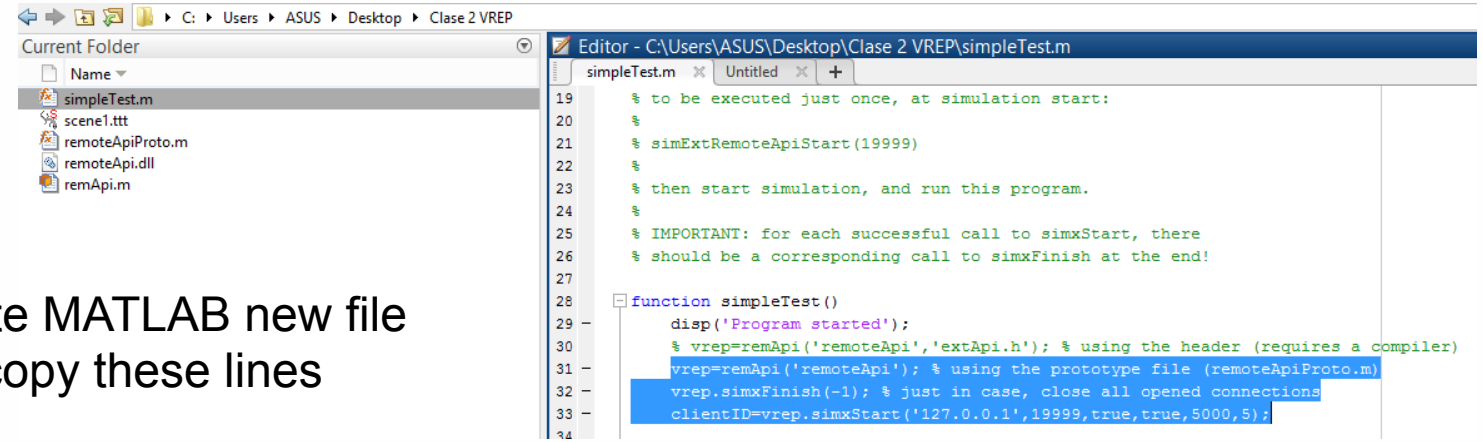


Move selected files to a new folder

Disable Pioneer script

Disable pioneer script





Create MATLAB new file and copy these lines

V-REP-Matlab functions info

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

simxStart

Description	Starts a communication thread with the server (i.e. V-REP). A same client may start several communication threads (but only one communication thread for a given IP and port). This should be the very first remote API function called on the client side. Make sure to start an appropriate remote API server service on the server side, that will wait for a connection. See also simxFinish. This is a remote API helper function.
Matlab synopsis	[number clientID]=simxStart(string connectionAddress,number connectionPort,boolean waitUntilConnected,boolean doNotReconnectOnceDisconnected,number timeOutInMs,number commThreadCycleInMs)
Matlab parameters	**connectionAddress**: the ip address where the server is located (i.e. V-REP) **connectionPort**: the port number where to connect. Specify a negative port number in order to use shared memory, instead of socket communication. **waitUntilConnected**: if true, then the function blocks until connected (or timed out). **doNotReconnectOnceDisconnected**: if true, then the communication thread will not attempt a second connection if a connection was lost. **timeOutInMs**: if positive: the connection time-out in milliseconds for the first connection attempt. In that case, the time-out for blocking function calls is 5000 milliseconds. if negative: its positive value is the time-out for blocking function calls. In that case, the connection time-out for the first connection attempt is 5000 milliseconds. **commThreadCycleInMs**: indicates how often data packets are sent back and forth. Reducing this number improves responsiveness, and a default value of 5 is recommended.
Matlab return values	**clientID**: the client ID, or -1 if the connection to the server was not possible (i.e. a timeout was reached). A call to simxStart should always be followed at the end with a call to simxFinish if simxStart didn't return -1
Other languages	C/C++, Python, Java, Octave, Lua

simxStartSimulation (regular API equivalent: sim.startSimulation)

Description	Requests a start of a simulation (or a resume of a paused simulation). This function is only executed by continuous remote API server services. See also simxPauseSimulation and simxStopSimulation.
Matlab synopsis	[number returnCode]=simxStartSimulation(number clientID,number operationMode)
Matlab parameters	**clientID**: the client ID. refer to simxStart. **operationMode**: a remote API function operation mode. Recommended operation mode for this function is simx_opmode_oneshot.
Matlab return values	**returnCode**: a remote API function return code
Other languages	C/C++, Python, Java, Octave, Lua

http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm#simxStart

V-REP-Matlab

Editor - C:\Users\ASUS\Desktop\Clase 2 VREP\pioneer.m

simpleTest.m | pioneer.m

```
1     % vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
2 -   vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
3 -   vrep.simxFinish(-1); % just in case, close all opened connections
4 -   clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
5
6 -   if (clientID>-1)    %check if connection is successfull
7 -       disp('Connected')
8
9         %code here
10
11 -       vrep.simxFinish(-1);
12 -  end
13
14 -  vrep.delete();
```

V-REP-Matlab functions info

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

simxGetObjectHandle (regular API equivalent: sim.getObjectHandle)

Description	Retrieves an object handle based on its name. If the client application is launched from a child script, then you could also let the child script figure out what handle correspond to what objects, and send the handles as additional arguments to the client application during its launch. See also simxGetObjectGroupData.
Matlab synopsis	[number returnCode,number handle]=simxGetObjectHandle(number clientID,string objectName,number operationMode)
Matlab parameters	**clientID**: the client ID. refer to simxStart. **objectName**: name of the object. If possibe, don't rely on the automatic name adjustment mechanism, and always specify the full object name, including the #: if the object is 'myJoint', specify 'myJoint#', if the object is 'myJoint#0', specify 'myJoint#0', etc. **operationMode**: a remote API function operation mode. Recommended operation mode for this function is simx_opmode_blocking
Matlab return values	**returnCode**: a remote API function return code **handle**: the handle
Other languages	C/C++, Python, Java, Octave, Lua

Editor - C:\Users\ASUS\Desktop\Clase 2 VREP\pioneer.m*

simpleTest.m | pioneer.m*

```
1     % vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
2 -   vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
3 -   vrep.simxFinish(-1); % just in case, close all opened connections
4 -   clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
5
6 -   if (clientID>-1)    %check if connection is successfull
7 -       disp('Connected')
8 -       [number returnCode,number handle]=simxGetObjectHandle(number clientID,string objectName,number operationMode)
9         %code here
10 -      vrep.simxFinish(-1);
11 -  end
12
13 -  vrep.delete();
```

V-REP-Matlab functions info

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Matlab return values	**returnCode**: a remote API function return code **handle**: the handle

Remote API function return codes

A remote API function return can be **a combination of following flags**:

simx_return_ok (**0**)
The function executed fine
simx_return_novalue_flag (**1** (i.e. bit 0))
There is no command reply in the input buffer. This should not always be considered as an error, depending on the selected operation mode
simx_return_timeout_flag (**2** (i.e. bit 1))
The function timed out (probably the network is down or too slow)
simx_return_illegal_opmode_flag (**4** (i.e. bit 2))
The specified operation mode is not supported for the given function
simx_return_remote_error_flag (**8** (i.e. bit 3))
The function caused an error on the server side (e.g. an invalid handle was specified)
simx_return_split_progress_flag (**16** (i.e. bit 4))
The communication thread is still processing previous split command of the same type
simx_return_local_error_flag (**32** (i.e. bit 5))
The function caused an error on the client side
simx_return_initialize_error_flag (**64** (i.e. bit 6))
simxStart was not yet called

Editor - C:\Users\ASUS\Desktop\Clase 2 VREP\pioneer.m

simpleTest.m | pioneer.m

```
1     % vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
2 -   vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
3 -   vrep.simxFinish(-1); % just in case, close all opened connections
4 -   clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
5
6 -   if (clientID>-1)    %check if connection is successfull
7 -       disp('Connected')
8 -       [returnCode,left_Motor]=vrep.simxGetObjectHandle(number clientID,string objectName,number operationMode)
9         %code here
10 -      vrep.simxFinish(-1);
11 -  end
12
13 -  vrep.delete();
```

V-REP-Matlab functions info

parent, or an object handle relative to whose reference frame you want the position
operationMode: a remote API function operation mode. Recommended operation modes for this function are simx_opmode_streaming (the first call) and simx_opmode_buffer (the following calls)

Remote API function operation modes

When most remote API functions are called, they will be translated into commands that might travel to the server, and come back as a command reply from the server. The operation mode of a remote API function defines what exactly happens to those commands and command replies:

simx_opmode_oneshot
: **Non-blocking mode**. The command is sent and a previous reply to the same command returned (if available). The function doesn't wait for the actual reply.

simx_opmode_blocking (or simx_opmode_oneshot_wait)
: **Blocking mode**. The command is sent, and the function will wait for the actual reply and return it (if the function doesn't time out). The received command reply will be removed from the inbox buffer (other operation modes will leave their command replies in the inbox buffer)

simx_opmode_streaming + alpha

Editor - C:\Users\ASUS\Desktop\Clase 2 VREP\pioneer.m

simpleTest.m | pioneer.m

```
1     % vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
2 -   vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
3 -   vrep.simxFinish(-1); % just in case, close all opened connections
4 -   clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
5
6 -   if (clientID>-1)    %check if connection is successfull
7 -       disp('Connected')
8 -       [returnCode,left_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.simx_opmode_blocking)
9         %code here
10 -      vrep.simxFinish(-1);
11 -  end
12
```

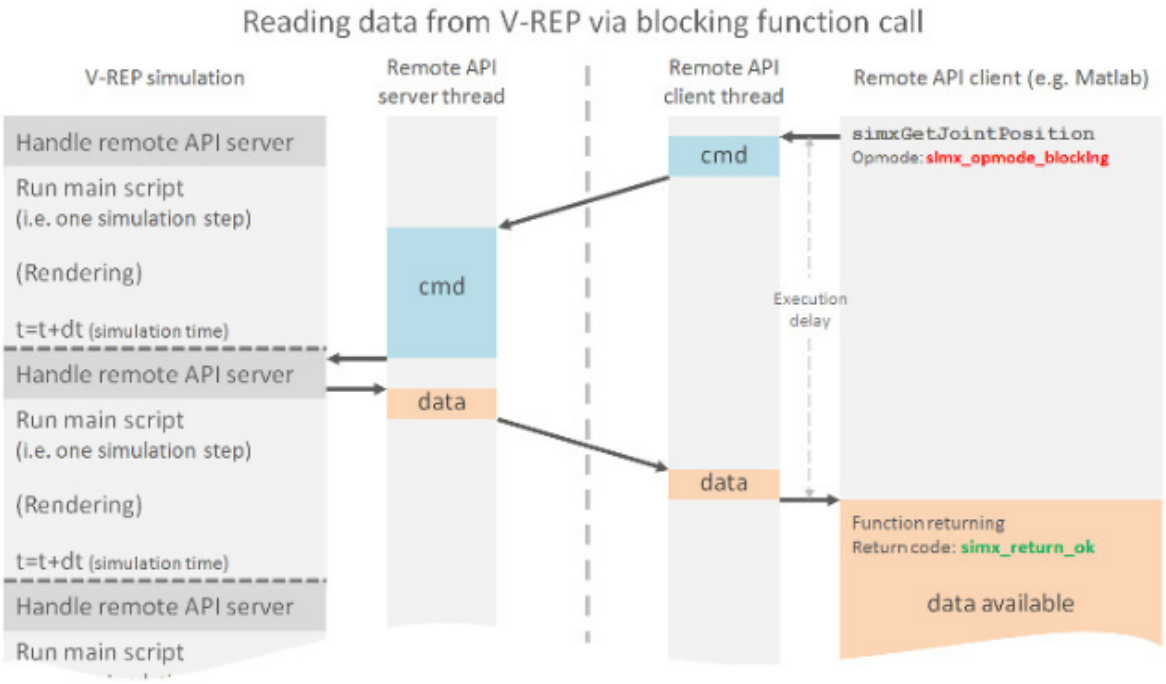
V-rep model name

V-REP-Matlab functions info

Blocking function calls: a blocking function call is the naive or regular approach, and meant for situations where we cannot afford not to wait for a reply from the server, like in following situation:

```
// Following function (blocking mode) will retrieve an object handle:
if (simxGetObjectHandle(clientID,"myJoint",&jointHandle,simx_opmode_blocking)==simx_return_ok)
{
    // here we have the joint handle in variable jointHandle!
}
```

Following diagram illustrates a blocking function call:



V-REP-Matlab functions info

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

simxSetJointTargetVelocity (regular API equivalent: sim.setJointTargetVelocity)

Description	Sets the intrinsic target velocity of a non-spherical joint. This command makes only sense when the joint mode is in torque/force mode: the dynamics functionality and the joint motor have to be enabled (position control should however be disabled)
Matlab synopsis	[number returnCode]=simxSetJointTargetVelocity(number clientID,number jointHandle,number targetVelocity,number operationMode)
Matlab parameters	**clientID**: the client ID. refer to simxStart. **jointHandle**: handle of the joint **targetVelocity**: target velocity of the joint (linear or angular velocity depending on the joint-type) **operationMode**: a remote API function operation mode. Recommended operation modes for this function are simx_opmode_oneshot or simx_opmode_streaming
Matlab return values	**returnCode**: a remote API function return code
Other languages	C/C++, Python, Java, Octave, Lua

Editor - C:\Users\ASUS\Desktop\Clase 2 VREP\pioneer.m*

simpleTest.m | pioneer.m*

```
1     % vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
2 -   vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
3 -   vrep.simxFinish(-1); % just in case, close all opened connections
4 -   clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
5
6 -   if (clientID>-1)    %check if connection is successfull
7 -       disp('Connected')
8 -       [returnCode,left_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.simx_opmode_blocking)
9 -       [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0.1,vrep.simx_opmode_blocking)
10        % I use the MODE vrep.simx_opmode_blocking, since recommended simx_opmode_oneshot or simx_opmode_streaming didnt work
11        %code here
12 -       vrep.simxFinish(-1);
13 -  end
14
15 -  vrep.delete();
```

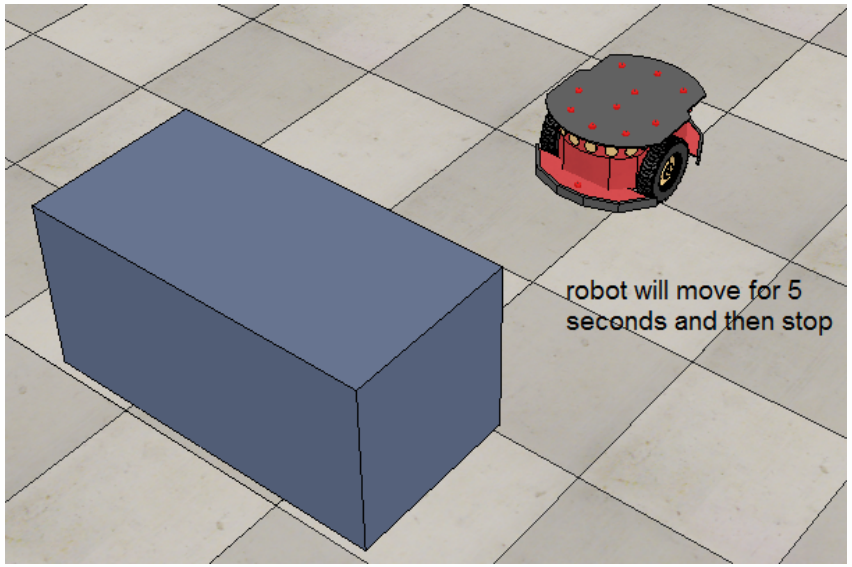
V-REP-Matlab functions info

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Editor - C:\Users\ASUS\Desktop\Clase 2 VREP\pioneer.m

simpleTest.m | pioneer.m

```
% vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

if (clientID>-1)    %check if connection is successfull
    disp('Connected')
    [returnCode,left_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.simx_opmode_blocking)
    [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0.1,vrep.simx_opmode_blocking)
    % I use the MODE vrep.simx_opmode_blocking, since recommended simx_opmode_oneshot or simx_opmode_streaming didnt work
    %code here

    pause(5)
    [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0,vrep.simx_opmode_blocking)

    vrep.simxFinish(-1);
end

vrep.delete();
```



V-REP-Matlab functions info

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

simxReadProximitySensor (regular API equivalent: sim.readProximitySensor)

Description	Reads the state of a proximity sensor. This function doesn't perform detection, it merely reads the result from a previous call to sim.handleProximitySensor (sim.handleProximitySensor is called in the default main script). See also simxGetObjectGroupData.
Matlab synopsis	[number returnCode,boolean detectionState,array detectedPoint,number detectedObjectHandle,array detectedSurfaceNormalVector]=simxReadProximitySensor(number clientID,number sensorHandle,number operationMode)
Matlab parameters	**clientID**: the client ID. refer to simxStart. **sensorHandle**: handle of the proximity sensor **operationMode**: a remote API function operation mode. Recommended operation modes for this function are simx_opmode_streaming (the first call) and simx_opmode_buffer (the following calls)
Matlab return values	**returnCode**: a remote API function return code **detectionState**: the detection state (false=no detection) **detectedPoint**: the detected point coordinates (relative to the sensor reference frame) **detectedObjectHandle**: the handle of the detected object **detectedSurfaceNormalVector**: the normal vector (normalized) of the detected surface. Relative to the sensor reference frame
Other languages	C/C++, Python, Java, Octave, Lua

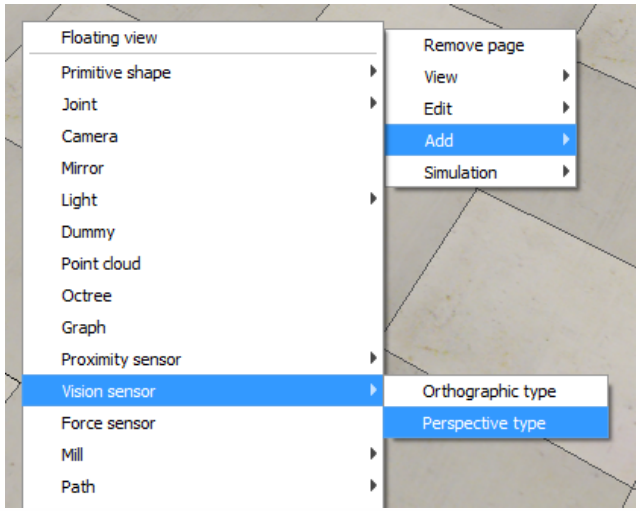
Read ultrasonic sensor matlab

Editor - C:\Users\ASUS\Desktop\Clase 2 VREP\pioneer.m

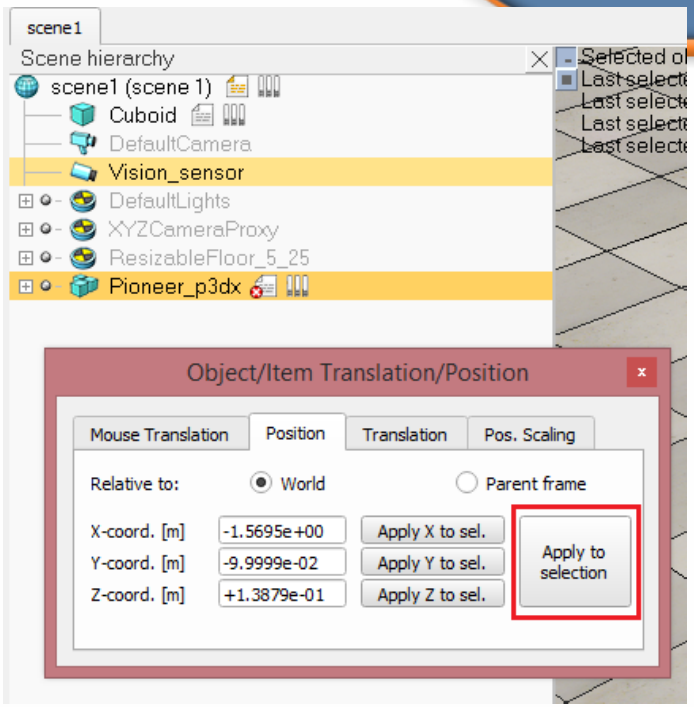
simpleTest.m | pioneer.m

```
8
9       %Recibir datos desde VREP (handle)
10      [returnCode,left_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.simx_opmode_blocking);
11      [returnCode,front_sensor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_ultrasonicSensor5',vrep.simx_opmode_blocking);
12
13      %other code
14      %Enviar dato a VREP (pioneer angular velocity)
15      [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0.1,vrep.simx_opmode_blocking);
16      %Leer dato de sensor de proximidad (Paso a.1 streaming para habilitar dato)
17      [returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(clientID,front_sensor,vrep.simx_opmode_streaming);
18
19      for i=1:50
20          %Leer dato de sensor de proximidad (Paso a.2 con streaming habilitado puedo leer buffer dato)
21          [returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(clientID,front_sensor,vrep.simx_opmode_buffer);
22          disp(norm(detectedPoint)) %quiero valor absoluto así que uso la norma
23          pause(0.1);
24      end
25
26
27      %[returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0,vrep.simx_opmode_blocking)
28
29      vrep.simxFinish(-1);
30  end
31
32  vrep.delete();
```

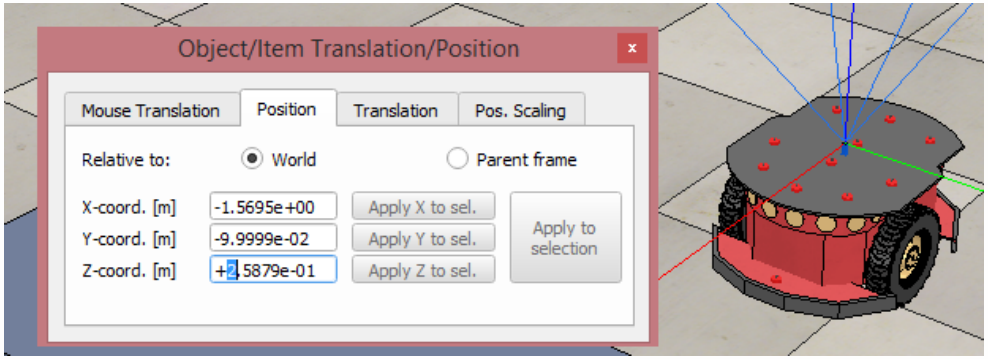
Vrep-Matlab / Vision camera



1) Create vision sensor

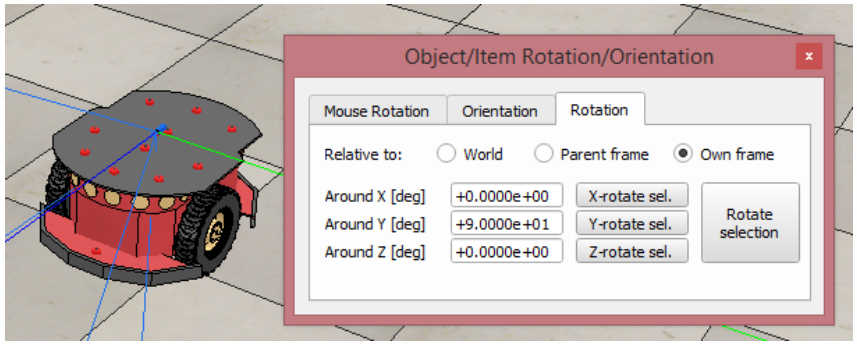


2) Move to the pioneer position

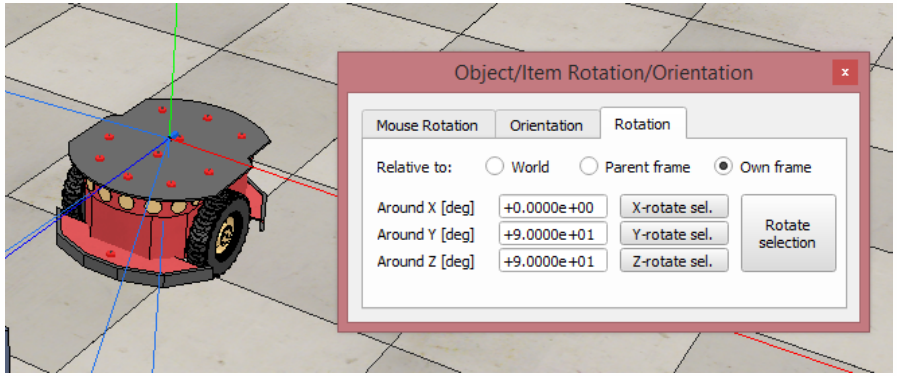


2) Move sensor +20cm in z-coord

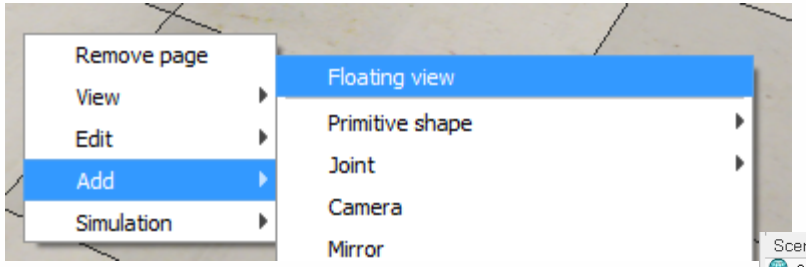
Vrep-Matlab / Vision camera



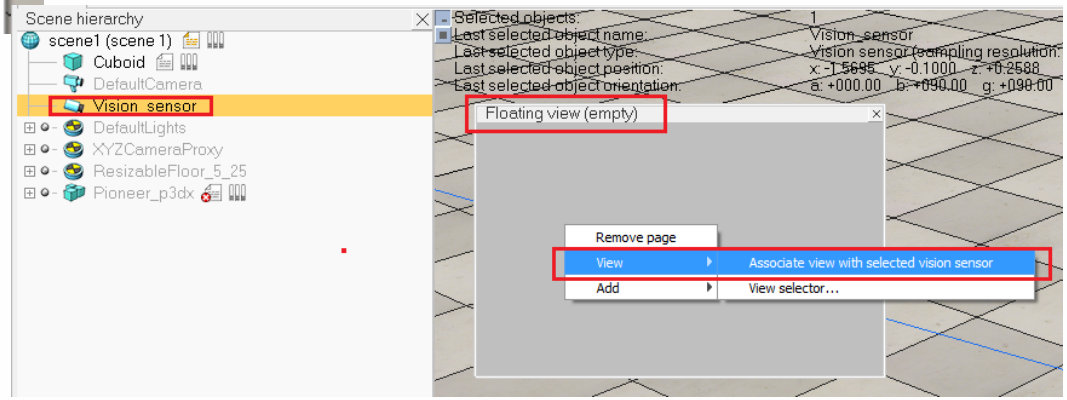
1) Rotate in “Y” by +90 degrees



2) Rotate in “Z” by +90 degrees

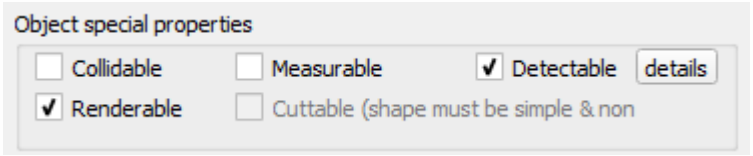


3) Add floating view

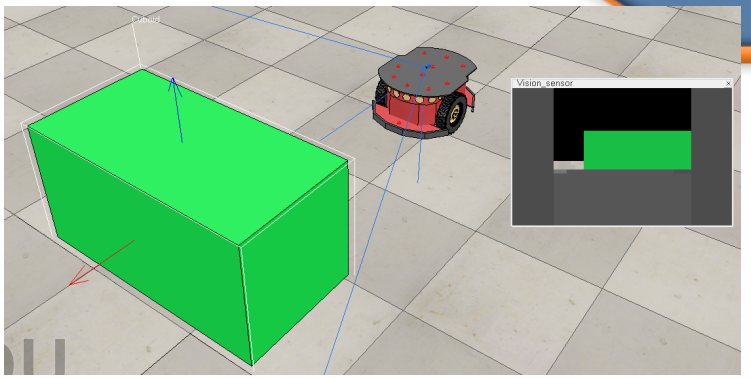


4) Associate vision sensor

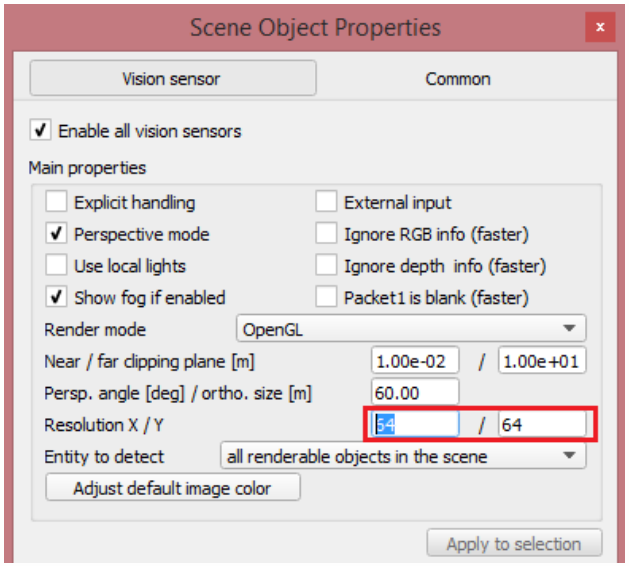
Vrep-Matlab / Vision camera



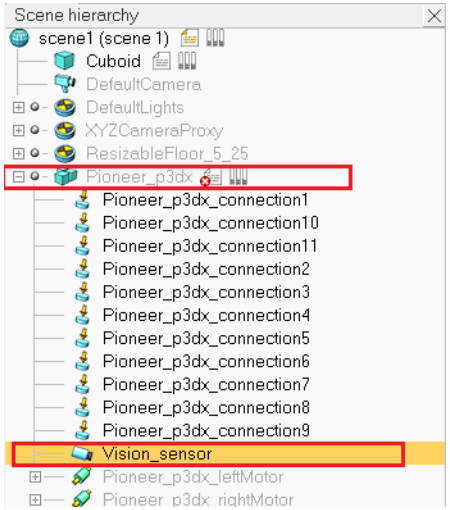
1) Object must be renderable



2) The object can be detected by the vision sensor



3) Change the image resolution



4) Move the sensor into the pioneer model

Vrep-Matlab / Vision camera

simxGetVisionSensorImage2 (regular API equivalent: sim.getVisionSensorImage)

Description	Retrieves the image of a vision sensor as an image array. The returned data doesn't make sense if sim.handleVisionSensor wasn't called previously (sim.handleVisionSensor is called by default in the main script if the vision sensor is not tagged as explicit handling). Use the simxGetLastCmdTime function to verify the *freshness* of the retrieved data. See also simxGetVisionSensorImage, simxSetVisionSensorImage, simxGetVisionSensorDepthBuffer and simxReadVisionSensor.
Matlab synopsis	[number returnCode,array resolution,matrix image]=simxGetVisionSensorImage2(number clientID,number sensorHandle,number options,number operationMode)
Matlab parameters	**clientID**: the client ID. refer to simxStart. **sensorHandle**: handle of the vision sensor **options**: image options, bit-coded: bit0 set: each image pixel is a byte (greyscale image), otherwise each image pixel is a rgb byte-triplet **operationMode**: a remote API function operation mode. Recommended operation modes for this function are simx_opmode_streaming (the first call) and simx_opmode_buffer (the following calls)
Matlab return values	**returnCode**: a remote API function return code **resolution**: 2 number values representing the resolution of the image **image**: the image data. Values are in the range of 0-255.
Other languages	C/C++, Python, Java, Octave, Lua

Vrep-Matlab / Vision camera

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Editor - C:\Users\ASUS\Desktop\Clase 2 VREP\pioneer2.m*

simpleTest.m | pioneer2.m*

```
% vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

if (clientID>-1)    %check if connection is successfull
    disp('Connected')

    %Habilitar recepcion de datos desde VREP (handle)
    [returnCode,left_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.simx_opmode_blocking);
    [returnCode,front_sensor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_ultrasonicSensor5',vrep.simx_opmode_blocking);
    [returnCode,camera]=vrep.simxGetObjectHandle(clientID,'Vision_sensor',vrep.simx_opmode_blocking);

    %other code
    %Enviar dato a VREP (pioneer angular velocity)
    [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0.1,vrep.simx_opmode_blocking);
    %Leer dato de sensor de proximidad (Paso a.1 streaming para habilitar dato)
    [returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(clientID,front_sensor,vrep.simx_opmode_streaming);
    %LEer dato de imagen como matriz de valores
    [returnCode,resolution,image]=vrep.simxGetVisionSensorImage2(clientID,camera,0,vrep.simx_opmode_streaming); %0 for RGB image
    for i=1:50
       %Leer dato de sensor de proximidad (Paso a.2 con streaming habilitado puedo leer buffer dato)
       [returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(clientID,front_sensor,vrep.simx_opmode_buffer);
       disp(norm(detectedPoint)) %quiero valor absoluto así que uso la norma

       %Leer dato de sensor de proximidad (Paso a.2 con streaming habilitado puedo leer buffer dato)
       [returnCode,resolution,image]=vrep.simxGetVisionSensorImage2(clientID,camera,0,vrep.simx_opmode_buffer);
       imshow(image)


       pause(0.1);
    end
    [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0,vrep.simx_opmode_blocking)
    vrep.simxFinish(-1);
end
vrep.delete();
```

Fundamentals of Robotics

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

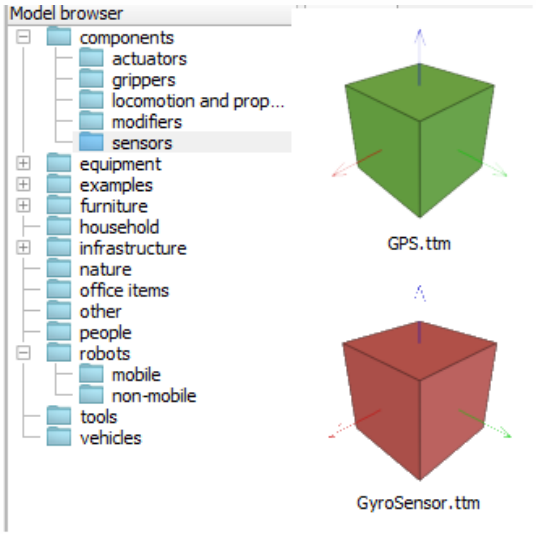
Virtual robot experimentation platform
V-REP

Class 3 - Features and sensors (Part2)

DEPARTAMENTO DE
ELECTRONICA

Valparaíso, Oct 2021

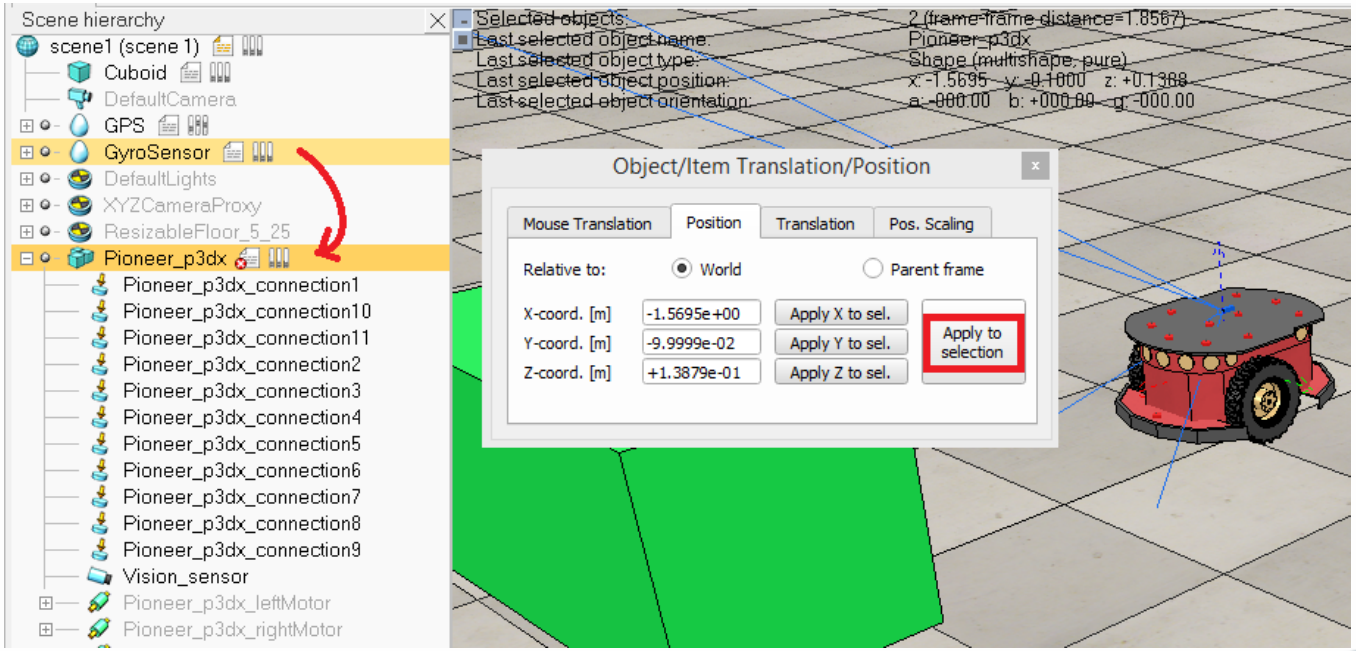
Vrep-Matlab / GPS and Gyroscope sensor



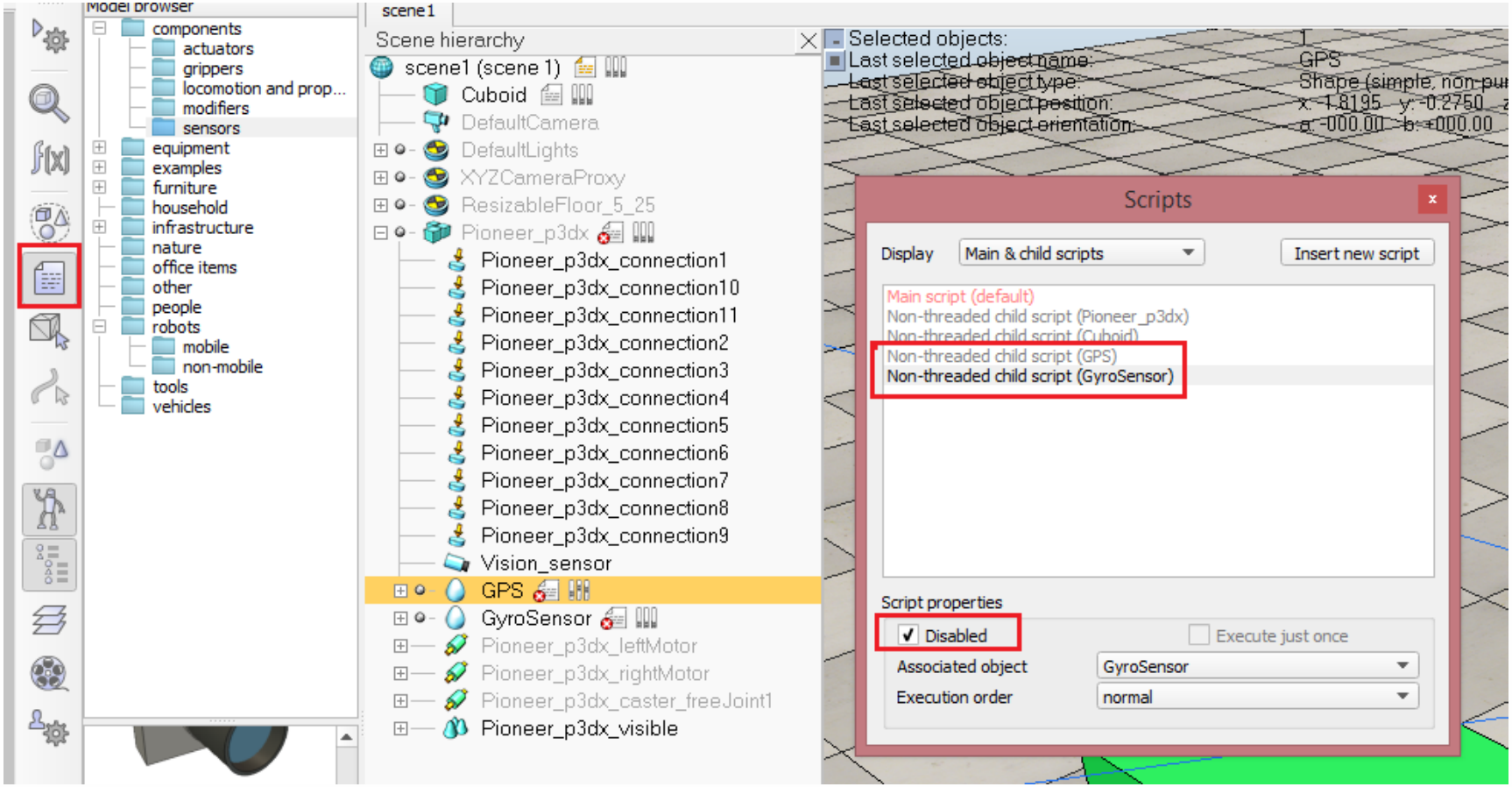
GPS-> position measure

Gyroscope -> Orientation

Robot Pose



Vrep-Matlab / GPS and Gyroscope sensor



Vrep-Matlab / GPS and Gyroscope sensor

simxGetObjectPosition (regular API equivalent: sim.getObjectPosition)

Description	Retrieves the position of an object. See also simxSetObjectPosition, simxGetObjectOrientation, simxGetObjectQuaternion and simxGetObjectGroupData.
Matlab synopsis	[number returnCode,array position]=simxGetObjectPosition(number clientID,number objectHandle,number relativeToObjectHandle,number operationMode)
Matlab parameters	**clientID**: the client ID. refer to simxStart. **objectHandle**: handle of the object **relativeToObjectHandle**: indicates relative to which reference frame we want the position. Specify -1 to retrieve the absolute position, sim_handle_parent to retrieve the position relative to the object's parent, or an object handle relative to whose reference frame you want the position **operationMode**: a remote API function operation mode. Recommended operation modes for this function are simx_opmode_streaming (the first call) and simx_opmode_buffer (the following calls)
Matlab return values	**returnCode**: a remote API function return code **position**: 3 values representing the position
Other languages	C/C++, Python, Java, Octave, Lua

simxGetObjectOrientation (regular API equivalent: sim.getObjectOrientation)

Description	Retrieves the orientation (Euler angles) of an object. See also simxSetObjectOrientation, simxGetObjectQuaternion, simxGetObjectPosition and simxGetObjectGroupData.
Matlab synopsis	[number returnCode,array eulerAngles]=simxGetObjectOrientation(number clientID,number objectHandle,number relativeToObjectHandle,number operationMode)
Matlab parameters	**clientID**: the client ID. refer to simxStart. **objectHandle**: handle of the object **relativeToObjectHandle**: indicates relative to which reference frame we want the orientation. Specify -1 to retrieve the absolute orientation, sim_handle_parent to retrieve the orientation relative to the object's parent, or an object handle relative to whose reference frame you want the orientation **operationMode**: a remote API function operation mode. Recommended operation modes for this function are simx_opmode_streaming (the first call) and simx_opmode_buffer (the following calls)
Matlab return values	**returnCode**: a remote API function return code **eulerAngles**: 3 values representing the Euler angles (alpha, beta and gamma)
Other languages	C/C++, Python, Java, Octave, Lua

Vrep-Matlab / GPS and Gyroscope sensor

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Editor - C:\Users\ASUS\Desktop\VREP Clases\pioneer3.m

Mainv2.m | pioneer3.m

```
% vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

if (clientID>-1)    %check if connection is successfull
    disp('Connected')

    %Habilitar recepcion de datos desde VREP (handle) y obtener tags de
    %variables
    [returnCode,left_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.simx_opmode_blocking);
    [returnCode,right_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_rightMotor',vrep.simx_opmode_blocking);
    [returnCode,front_sensor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_ultrasonicSensor5',vrep.simx_opmode_blocking);
    [returnCode,camera]=vrep.simxGetObjectHandle(clientID,'Vision_sensor',vrep.simx_opmode_blocking);
    [returnCode,robot_gps]    = vrep.simxGetObjectHandle(clientID,'GPS',vrep.simx_opmode_blocking);
    [returnCode,robot_gyro]   = vrep.simxGetObjectHandle(clientID,'GyroSensor',vrep.simx_opmode_blocking);

    %MOTOR
    %Enviar dato a VREP (pioneer angular velocity)
    [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0.1,vrep.simx_opmode_blocking);
    %[returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,0.1,vrep.simx_opmode_blocking);

    %PROXIMITY SENSOR
    %Leer dato de sensor de proximidad (Paso a.1 streaming para habilitar dato)
    [returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(clientID,front_sensor,vrep.simx_opmode_streaming);

    %VISION SENSOR
    %LEer dato de imagen como matriz de valores
    [returnCode,resolution,image]=vrep.simxGetVisionSensorImage2(clientID,camera,0,vrep.simx_opmode_streaming); %0 for RGB image

    % GPS SENSOR
    %To obtain robot position (reference purposes
    [returnCode,robot_position]=vrep.simxGetObjectPosition(clientID,robot_gps,-1,vrep.simx_opmode_oneshot);

    % GYROSCOPE SENSOR
    %To obtain orientation angle of robot
    [returnCode,robot_theta] = vrep.simxGetObjectOrientation(clientID,robot_gyro,-1,vrep.simx_opmode_streaming);
    [returnCode,robot_theta] = vrep.simxGetObjectOrientation(clientID,robot_gyro,-1,vrep.simx_opmode_buffer);
```

Vrep-Matlab / GPS and Gyroscope sensor

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Editor - C:\Users\ASUS\Desktop\VREP Clases\pioneer3.m

Mainv2.m | pioneer3.m

```
39
40        %x = 0:pi/100:200*pi;
41
42        for i=1:600
43
44           %-----------------------------distance-----------------------------------
45           %Leer dato de sensor de proximidad (Paso a.2 con streaming habilitado puedo leer buffer dato)
46           [returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(clientID,front_sensor,vrep.simx_opmode_buffer);
47           %disp(norm(detectedPoint)); %quiero valor absoluto así que uso la norma
48
49           %-----------------------------vision-------------------------------------
50           %Leer dato de sensor de proximidad (Paso a.2 con streaming habilitado puedo leer buffer dato)
51           [returnCode,resolution,image]=vrep.simxGetVisionSensorImage2(clientID,camera,0,vrep.simx_opmode_buffer);
52           %imshow(image)
53
54           %----------------------gyroscope-----------------------------------
55
56           %Extract theta angles
57           [returnCode,robot_theta] = vrep.simxGetObjectOrientation(clientID,robot_gyro,-1,vrep.simx_opmode_buffer);
58           theta_robot=double(robot_theta(3));
59           %Fixing theta
60            if theta_robot<0
61                theta_robot=theta_robot+2*pi;
62            end
63
64           disp(theta_robot);
65
66           %----------------------GPS-----------------------------------------
67           %Extract GPS position
68           [returnCode,robot_position]=vrep.simxGetObjectPosition(clientID,robot_gps,-1,vrep.simx_opmode_oneshot);
69           xy_robot=double([robot_position(1);robot_position(2)]);
70           %disp(xy_robot);
71
72           pause(0.1);
73        end
74        [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0,vrep.simx_opmode_blocking)
75        vrep.simxFinish(-1);
76     end
77     vrep.delete();
```

Joystick in Matlab - Example

Editor - C:\Users\ASUS\Desktop\VREP Clases\joystick.m

joystick.m | pioneer4.m

```
1  -    joy = vrjoystick(1)
2
3  -    for cont = 1:300
4  -    disp('medicion')
5  -    w = -axis(joy, 1);
6  -    disp(w)
7
8  -    X = button(joy, 1);
9  -    disp(X)
10
11 -    O = button(joy, 3);
12 -    disp(O)
13
14
15 -    pause(0.1)
16
17 -    end
18
19 -    close(joy) % cierro el objeto joystick
```

Joystick VREP / Matlab - Example

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Editor - C:\Users\ASUS\Desktop\VREP Clases\pioneer4.m

joystick.m | pioneer4.m

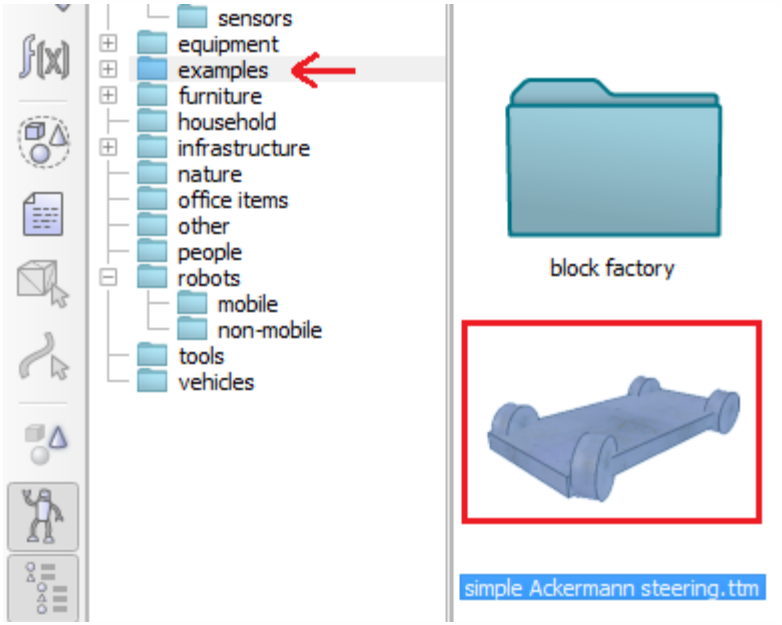
```
% vrep=remApi('remoteApi','extApi.h'); % using the header (requires a compiler)
vrep=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
vrep.simxFinish(-1); % just in case, close all opened connections
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

joy = vrjoystick(1) ;  %Inicializo Joystick

vel_M_izq=0.1;
vel_M_der=0.3;

if (clientID>-1)    %check if connection is successfull
    disp('Connected')

    %Habilitar recepcion de datos desde VREP (handle) y obtener tags de
    %variables
    [returnCode,left_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMotor',vrep.simx_opmode_blocking);
    [returnCode,right_Motor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_rightMotor',vrep.simx_opmode_blocking);
    [returnCode,front_sensor]=vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx_ultrasonicSensor5',vrep.simx_opmode_blocking);
    [returnCode,camera]=vrep.simxGetObjectHandle(clientID,'Vision_sensor',vrep.simx_opmode_blocking);
    [returnCode,robot_gps]    = vrep.simxGetObjectHandle(clientID,'GPS',vrep.simx_opmode_blocking);
    [returnCode,robot_gyro]   = vrep.simxGetObjectHandle(clientID,'GyroSensor',vrep.simx_opmode_blocking);

    %MOTOR
    %Enviar dato a VREP (pioneer angular velocity)
    [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,vel_M_izq,vrep.simx_opmode_blocking);
    [returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,vel_M_der,vrep.simx_opmode_blocking);

    %PROXIMITY SENSOR
    %Leer dato de sensor de proximidad (Paso a.1 streaming para habilitar dato)
    [returnCode,detectionState,detectedPoint,~,~]=vrep.simxReadProximitySensor(clientID,front_sensor,vrep.simx_opmode_streaming);

    %VISION SENSOR
    %LEer dato de imagen como matriz de valores
    [returnCode,resolution,image]=vrep.simxGetVisionSensorImage2(clientID,camera,0,vrep.simx_opmode_streaming); %0 for RGB image
```

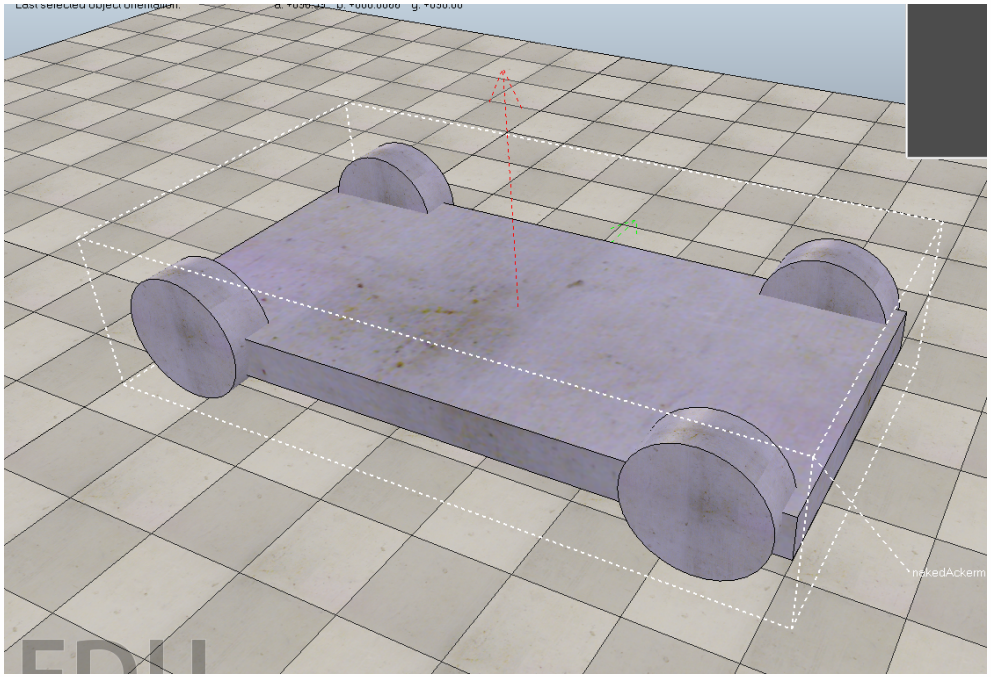
Joystick VREP / Matlab - Example

joystick.m | pioneer4.m*

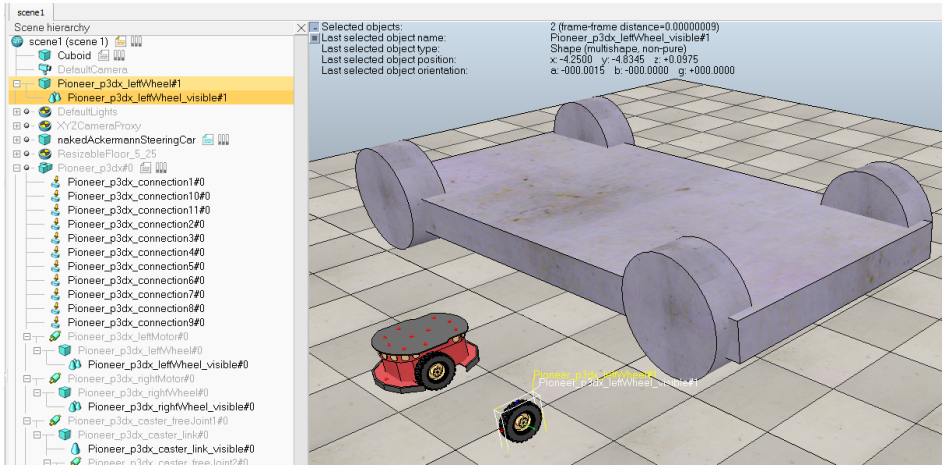
```
77          %---------------Robot movement using joystick----------------------
78
79 -        w = -axis(joy, 1);
80 -        X = button(joy, 1);
81 -        O = button(joy, 3);
82 -        P = button(joy, 2);
83 -        T = button(joy, 4);
84
85 -        if (w >= 0) && (w <= 0.3)
86 -             vel_M_der=0.3;
87 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,vel_M_der,vrep.simx_opmode_blocking)
88 -        elseif (w > 0.3) && (w <= 0.6)
89 -            vel_M_der=0.6;
90 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,vel_M_der,vrep.simx_opmode_blocking)
91 -        elseif (w > 0.6) && (w <= 1)
92 -            vel_M_der=0.99;
93 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,vel_M_der,vrep.simx_opmode_blocking)
94 -        elseif (w < 0) && (w >= -0.3)
95 -             vel_M_der=0.3;
96 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,vel_M_der,vrep.simx_opmode_blocking);
97 -        elseif (w < -0.3) && (w >= -0.6)
98 -            vel_M_der=0.6;
99 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,vel_M_der,vrep.simx_opmode_blocking);
100 -       elseif (w < -0.6) && (w >= -0.99)
101 -           vel_M_der=0.99;
102 -           [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,vel_M_der,vrep.simx_opmode_blocking);
103 -       end
104
105 -       if X==1
106 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,1,vrep.simx_opmode_blocking);
107 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,1,vrep.simx_opmode_blocking);
108 -       elseif O==1
109 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,0,vrep.simx_opmode_blocking);
110 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,0,vrep.simx_opmode_blocking);
111 -       elseif P==1
112 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,-1,vrep.simx_opmode_blocking);
113 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,-1,vrep.simx_opmode_blocking);
114 -       elseif T==1
115 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,left_Motor,-0.5,vrep.simx_opmode_blocking);
116 -            [returnCode]=vrep.simxSetJointTargetVelocity(clientID,right_Motor,-0.5,vrep.simx_opmode_blocking);
117 -       end
```

Vrep-Matlab / Car-like vehicle

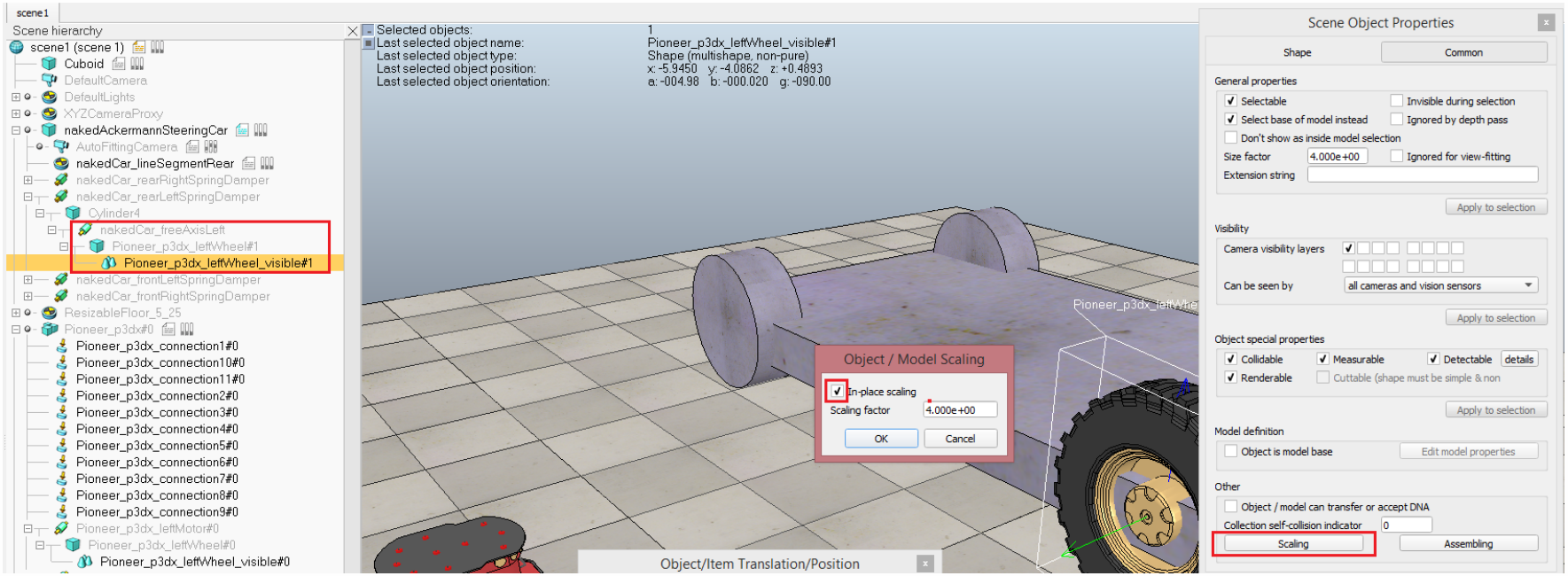




Vrep-Matlab / Car-like vehicle



- Copy Pioneer wheel
- Rotate 90 degrees
- Move to car-like wheel position
- Erase car-like wheel
- Scale pioneer wheel (in-place)



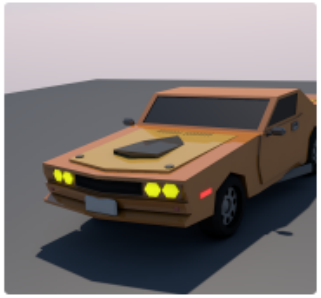
Importing 3D models - .obj

Free3D | car | Free 3D Models | Premium 3D Models

Lexus
.obj .dds
FREE 9,729

Car Scrap
.obj
FREE 6,672

VW Bus
.obj .dae
FREE 4,391



Low Poly Car
.3ds .obj .c4d .fbx
FREE 6,317

Citroen 2CV
.obj .max
FREE 6,930

Lamborghini Aventador Sport
.obj .dae
FREE 12,344

Low Poly Car
.obj .blend .fbx
FREE 3,147

Camaro 2009
.obj .lwo .lws
FREE 23,877

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

AC3E
Centro Avanzado de Ingeniería
Eléctrica y Electrónica

CONICYT
Ministerio de Educación
Gobierno de Chile

Thanks.

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA