Lab Number 03
CoppeliaSim EDU – Translation, Rotation and Transformation

Introduction:

CoppeliaSim uses different languages such as Lua, python or MATLAB etc. In this lab we will study Lua programming. Since Lua programming is used in many industrial grade robots.

Software Used:

CoppeliaSim EDU v4.7

Programming Language:

LUA
Basics can be learnt from the website: www.tutorialspoint.com/lua/

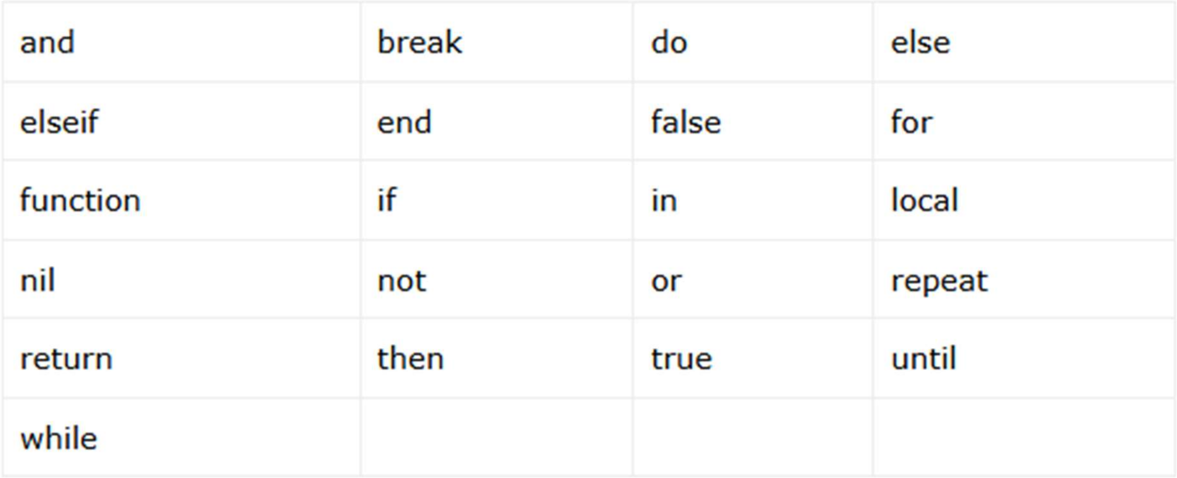
Objectives of the Lab:

- To learn basics of LUA programming
- To use revolute and prismatic joints

Programming basics:

Syntax is almost same as C language.
Data Types → nil, Boolean, number, string, tables
Variable declaration → assignment
Function call with variable arguments
print("test") displays test in the output console.

Keywords:



and	break	do	else
elseif	end	false	for
function	if	in	local
nil	not	or	repeat
return	then	true	until
while			

Example:

```
-- Variable definition:
local a, b

-- Initialization
a = 10
b = 30

print("value of a:", a)

print("value of b:", b)

-- Swapping of variables
b, a = a, b

print("value of a:", a)

print("value of b:", b)

f = 70.0/3.0
print("value of f", f)
```

Loops:

```
while( true )
do
  print("This loop will run forever.")
end
```

Functions:

```
function max(num1, num2)

  if (num1 > num2) then
    result = num1;
  else
    result = num2;
  end

  return result;
end
-- calling a function
print("The maximum of the two numbers is ",max(10,4))
print("The maximum of the two numbers is ",max(5,6))
```

Strings:

```
string1 = "Lua"
print("\"String 1 is\"",string1)

string2 = 'Tutorial'
print("String 2 is",string2)

string3 = [["Lua Tutorial"]]
print("String 3 is",string3)
```

Arrays:

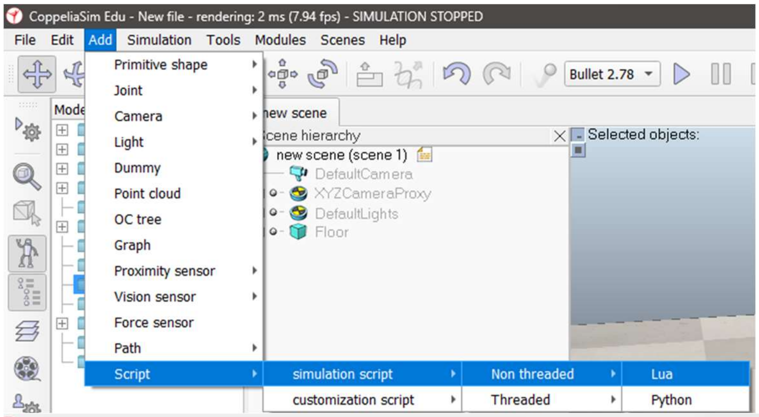
```
array = {"Lua", "Tutorial"}

for i = 0, 2 do
  print(array[i])
end
```

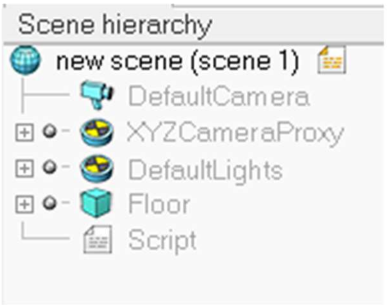
Modules:

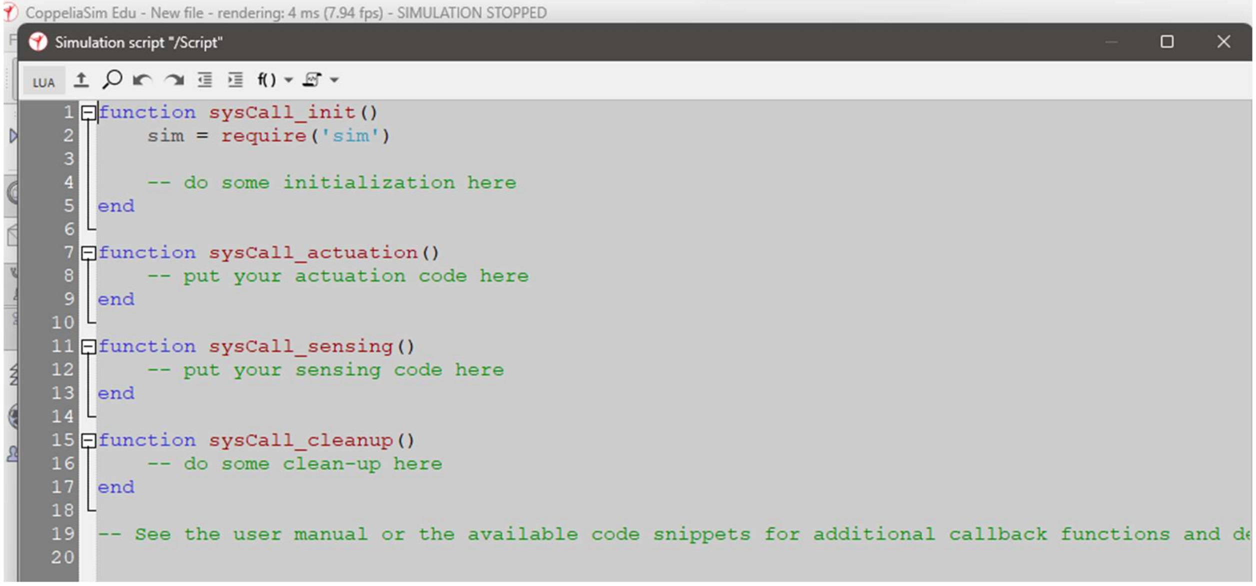
```
-- Assuming we have a module printFormatter
-- Also printFormatter has a funtion simpleFormat(arg)
-- Method 1
require "printFormatter"
printFormatter.simpleFormat("test")

-- Method 2
local formatter = require "printFormatter"
formatter.simpleFormat("test")

-- Method 3
require "printFormatter"
local formatterFunction = printFormatter.simpleFormat
formatterFunction("test")
```

Matrices and Vectors will be discussed in the script made in CoppeliaSim as they provide major role in transformations. First of all create a new script in the scene.

ADDING A SCRIPT IN CoppeliaSim EDU:






CoppeliaSim Edu - New file - rendering: 4 ms (7.94 fps) - SIMULATION STOPPED
Simulation script "/Script"
LUA

```lua
function sysCall_init()
    sim = require('sim')

    -- do some initialization here
end

function sysCall_actuation()
    -- put your actuation code here
end

function sysCall_sensing()
    -- put your sensing code here
end

function sysCall_cleanup()
    -- do some clean-up here
end

-- See the user manual or the available code snippets for additional callback functions and de
```

Function sysCall_init is used to initialize the handles (robots or objects) that are to be used. Function sysCall_actuation is used when you have to actuate something in the scene. Such as actuating a revolute joint as motors.

ROTATION MATRIX:

Write the code inside sysCall_init ( )

```
function sysCall_init()
   sim = require('sim')

A  = Matrix3x3:rotz(math.pi/2)    - - rotation matrix (rotated around z axis by angle pi/2)
Matrix.print(A)                   - - Print the matrix
   -- do some initialization here
end
```

TRANSLATIONAL VECTOR:

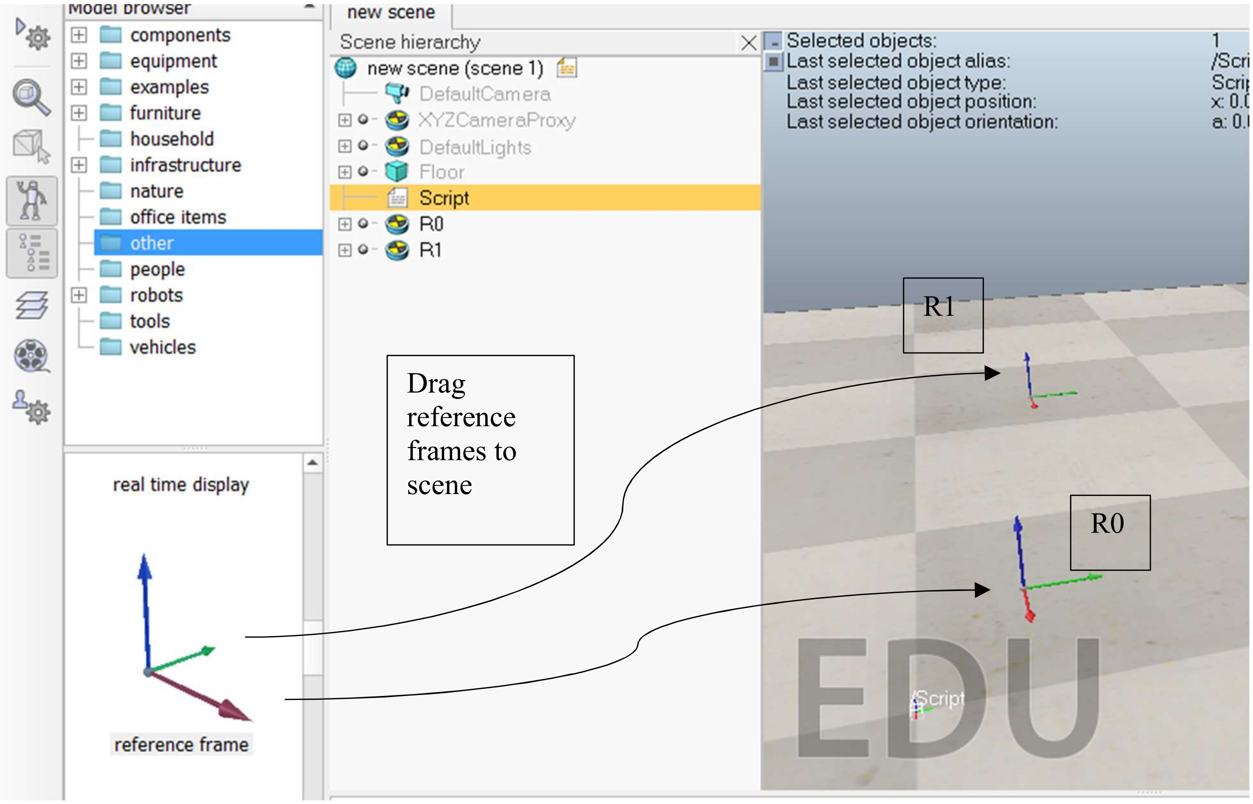
Now translational vector will be printed as follows

```
t = vector3({2,3,4})
Matrix.print(t)
```

TRANSFORMATION MATRIX:

```
A  = Matrix4x4({1,0,0, 1,          - - Add a transformation Matrix
              0,2,0, -2,
              1,2,0, 3,
              0,0,0, 1})

Matrix.print(A)                        - - print the matrix A
print(Matrix4x4:torotation(A))         - - print the rotation from transformation matrix
print(Matrix4x4:toeuler(A))            - - print the euler angles
print(Matrix4x4:toposition(A))         - - print the translation of the transformation matrix
```

Now visualizing the frames in CoppeliaSim.
Add the two reference frames from model browser
Model browser → other → reference frame
Name them as R0 and R1



Selected objects: 1
Last selected object alias: /Scri
Last selected object type: Scrip
Last selected object position: x: 0.0
Last selected object orientation: a: 0.

R1

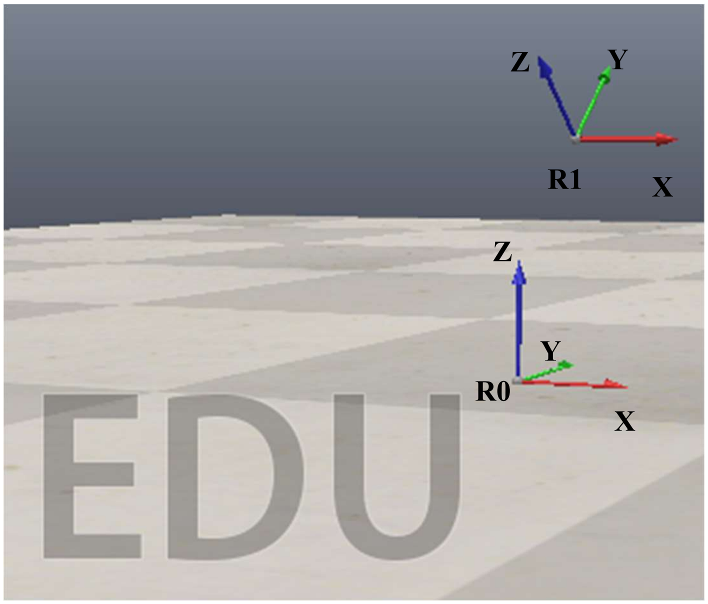
Drag reference frames to scene

R0

HOMOGENOUS TRANSFORM:

Write down the code in the script’s sysCall_init ( )

```
function sysCall_init()
   sim = require('sim')                  - - module sim

R0 = sim.getObjectHandle('/R0')   - - initialize an object R0 (reference frame)
R1 = sim.getObjectHandle('/R1')   - - initialize an object R1
R = Matrix3x3:rotx(math.pi/4)       - - Make a rotation matrix to rotate the frame R1 by
                                          pi/4 relative to R0
t = Vector3({0, 0.1, 0.2})              - - Make a vector such that to translate R1 by t relative
                                          to R0
A1 = Matrix4x4:fromrt(R,t)            - - Make the transformation matrix by concatenating R
                                          and t to make a 4x4 matrix
sim.setObjectMatrix(R1,R0,A1:data())     - - set object position and orientation

end
```



Now we will print a homogeneous transformation matrix of R1 relative to R0. Adding the following line in the code already done will make homogeneous transformation matrix as shown below:

```
A1m = Matrix (3,4, sim.getObjectMatrix(R1,R0))
A2 = Matrix:vertcat(A1m,Matrix(1,4,{0,0,0,1}))
Matrix.print(A2)
```

USING THE AXIS METHOD:

Using the Axis method to compute the rotation matrix R from the given transformation matrix such that it is rotated by angle “Y”.

```
Q = A2:axis("y")
T = Matrix3x3:fromaxisangle(Q,math.pi/4)
Matrix.print(T)
```

By adding the above code we get the rotation matrix when rotated by 45 degrees around Y axis.

Matrix.print(T:inv( )) - - takes the inverse of the matrixs