

**INTRODUCTION TO ROBOTICS**

**(MTS -417)**

**DE-44 Mechatronics Syndicate– C**

**Lab Report 2**

**Name of members:**

**NC** M. Ahmed Hussain Babar (**427781)**

**PC** Abdul Haseeb Zahid (**432219)**

**PC** Rehan Shahid (**432283)**

**Submitted to**: LE Hamza Sohail

**Task 1:**

* **Indirect Placement of R2 using R1:**
* **Lua Script:**

|  |
| --- |
| function sysCall\_init()  sim= require('sim') -- Core API  matrix= require('matrix') -- For matrix ops (CoppeliaSim's built-in)    -- Get handles (assume objects named /R0, /R1, /R2 in scene)  R0 = sim.getObjectHandle('/R0')  R1 = sim.getObjectHandle('/R1')  R2 = sim.getObjectHandle('/R2')    -- Initial positions: Ensure at origin (optional, for reset)  sim.setObjectPosition(R1, -1, {0, 0, 0})  sim.setObjectPosition(R2, -1, {0, 0, 0})  sim.setObjectPosition(R0, -1, {0, 0, 0})    theta1 = math.pi / 4 -- 45° in radians  Rz1 = Matrix3x3:rotz(theta1) -- Rotation matrix  t1 = Vector3({0.432, 0, 0}) -- Translation vector  T1 = Matrix4x4:fromrt(Rz1, t1)    theta2 = 54 \* math.pi / 180 --  Ry2 = Matrix3x3:roty(theta2)  t2 = Vector3({0, 0, 0.219})  T2 = Matrix4x4:fromrt(Ry2, t2)  sim.setObjectMatrix(R1, R0, T1:data())  sim.setObjectMatrix(R2, R1, T2:data())  local R2\_world\_pos = sim.getObjectPosition(R2, -1)  print(string.format("R2 Global Position: x=%.3f, y=%.3f, z=%.3f",  R2\_world\_pos[1], R2\_world\_pos[2], R2\_world\_pos[3]))  end  function sysCall\_actuation()  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 details |

* **Simulation Snippet:**

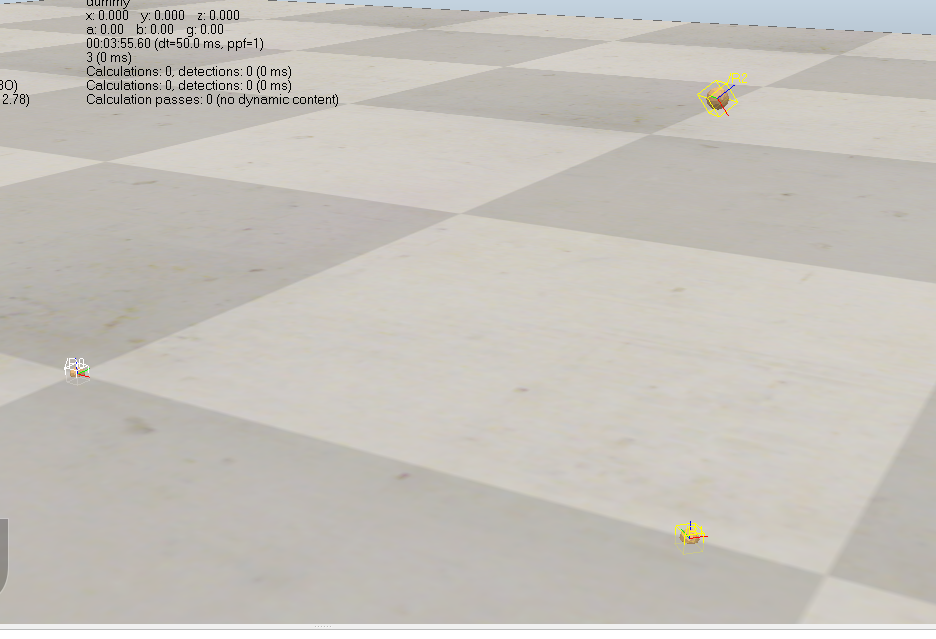


Figure 1

**Task 2:**

* **Direct Placement of R2 using Transformations product:**
* **Lua Script:**

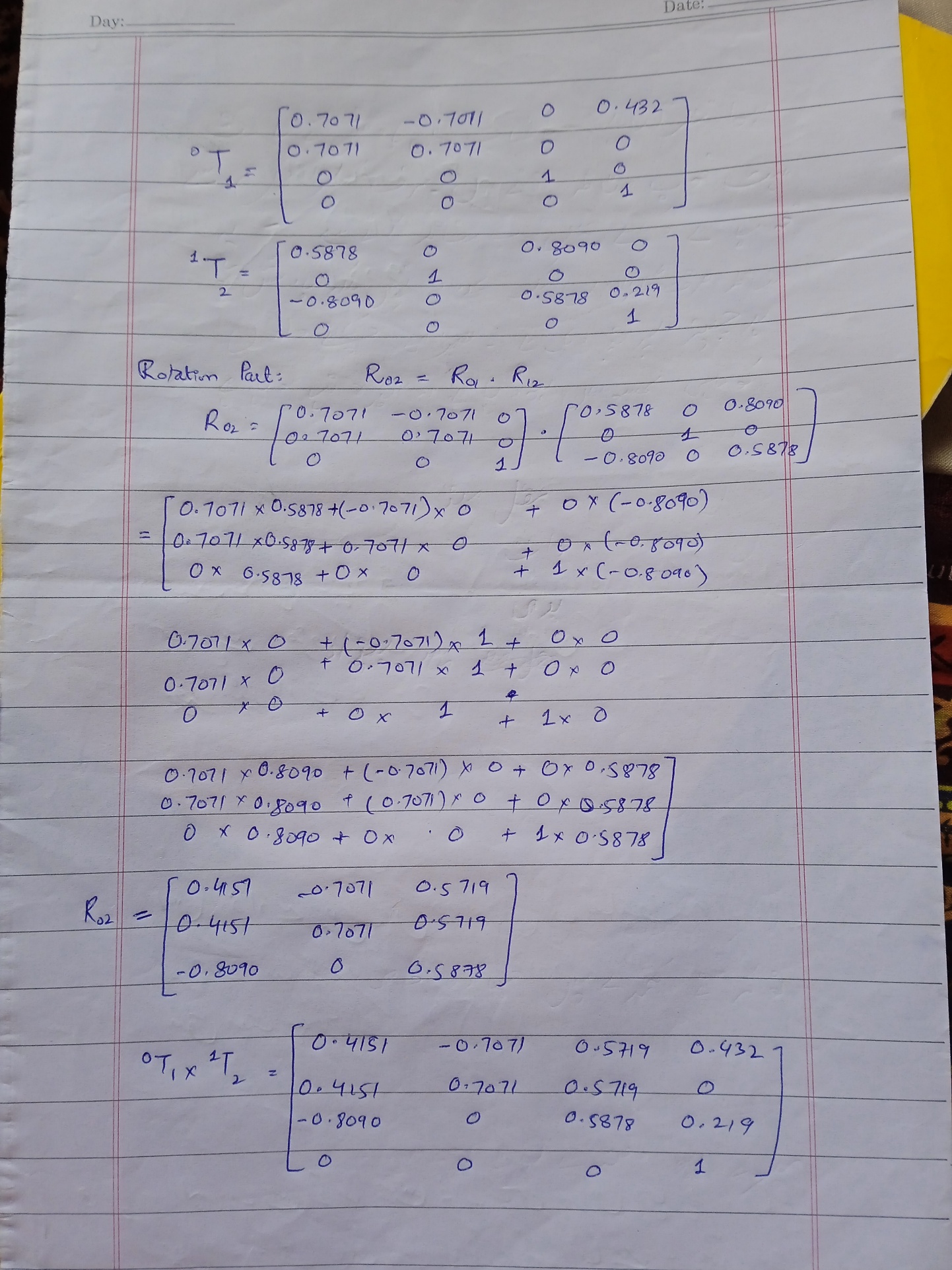
|  |
| --- |
| function sysCall\_init()  sim= require('sim') -- Core API  matrix= require('matrix') -- For matrix ops (CoppeliaSim's built-in)    -- Get handles (assume objects named /R0, /R1, /R2 in scene)  R0 = sim.getObjectHandle('/R0')  R1 = sim.getObjectHandle('/R1')  R2 = sim.getObjectHandle('/R2')    -- Initial positions: Ensure at origin (optional, for reset)  sim.setObjectPosition(R1, -1, {0, 0, 0})  sim.setObjectPosition(R2, -1, {0, 0, 0})  sim.setObjectPosition(R0, -1, {0, 0, 0})    theta1 = math.pi / 4 -- 45° in radians  Rz1 = Matrix3x3:rotz(theta1) -- Rotation matrix  t1 = Vector3({0.432, 0, 0}) -- Translation vector  T1 = Matrix4x4:fromrt(Rz1, t1)    theta2 = 54 \* math.pi / 180 --  Ry2 = Matrix3x3:roty(theta2)  t2 = Vector3({0, 0, 0.219})  T2 = Matrix4x4:fromrt(Ry2, t2)  T3 = T1\*T2  sim.setObjectMatrix(R2, R0, T3:data())  local R2\_world\_pos = sim.getObjectPosition(R2, -1)  print(string.format("R2 Global Position: x=%.3f, y=%.3f, z=%.3f",  R2\_world\_pos[1], R2\_world\_pos[2], R2\_world\_pos[3]))  end  function sysCall\_actuation()  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 details |

* **Simulation Snippet:**

****

Figure 2

**T1.T2 Multiplication:**

****

**Discussion**:

Chaining transformations through an intermediate frame (R1) is equivalent to applying the composite transformation directly because homogeneous transformations are composable via matrix multiplication, which is associative. This means ​represents the same overall rotation and translation. In robotics, this allows efficient forward kinematics by multiplying link transformations.

**Task 2:**

#### Answer: 45° Explanation:

The answer is correct because sim.setObjectMatrix in CoppeliaSim sets the transformation matrix of an object directly relative to the specified reference frame, overwriting any prior pose. In this case, the reference is R0, so the matrix T1 (which includes a 45° rotation about Z) replaces the initial 45° rotation entirely. The rotation angles do not add up; the function does not perform composition unless the reference frame is the object itself (which would apply the new transformation relative to its current pose). Here, since the reference is R0, the final orientation remains 45° about the Z-axis wrt R0.

**References:** Grok, Chatgpt