A Simulated 3D Printer in CoppeliaSim

Sudip Chakraborty[1] & P. S. Aithal[2]

[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India

OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2]ViceChancellor, Srinivas University, Mangalore, India

OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Subject Area: Information Technology.
Type of the Paper: Simulation-based Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.5897079
Google Scholar Citation: IJAEML

How to Cite this Paper:

Sudip Chakraborty & Aithal, P. S., (2022). A Simulated 3D Printer in CoppeliaSim. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *6*(1), 22-32. DOI: https://doi.org/10.5281/zenodo.5897079

International Journal of Applied Engineering and Management Letters (IJAEML)
A Refereed International Journal of Srinivas University, India.

Crossref DOI : https://doi.org/10.47992/IJAEML.2581.7000.0117

© With Authors.

This work is licensed under a Creative Commons Attribution-Non-Commercial 4.0 International License subject to proper citation to the publication source of the work.
Disclaimer: The scholarly papers as reviewed and published by the Srinivas Publications (S.P.), India are the views and opinions of their respective authors and are not the views or opinions of the S.P. The S.P. disclaims of any harm or loss caused due to the published content to any party.

A Simulated 3D Printer in CoppeliaSim

Sudip Chakraborty[1] & P. S. Aithal[2]

[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India

OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2]ViceChancellor, Srinivas University, Mangalore, India

OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *Now, 3D Printer is becoming popular. Not only used for printing toys but also to print some needful items for prototyping. We go for final production after prototype approval. So the researcher needs a 3D printer for their research. A good 3D printer is not only costly but cumbersome also. Carrying a 3D printer is too much problem. We can use the simulator to continue our research work in this scenario. Here we provide the procedure for building a 3D printer in the simulator. We used Coppeliasim to make it. It is an excellent open-source simulator. It has a good user interface. Following the step, the researcher can build it quickly. All project code and simulator files are available for the researcher in Github. They can download, use and integrate into their project without any issues.*

Design/Methodology/Approach: *In the CoppeliaSim simulator, we created a 3D printer model. We adopted the Ender 3D printer specification, a popular and economical 3D printer. After completing the model, we add the LUA script associated with the base. Some code is kept as a module inside the project folder for better understanding. After running the simulation, it will draw a square on the work area for the working demonstration.*

Findings/Result: *We can create a virtual 3D printer through this research work. So without having an actual printer, we can test our written algorithm, try the 3D model file, and understand the functionality of the 3D Printer. It can be integrated into the researcher's project by customizing some physical parameters or making a different 3D printer model following the procedure described here.*

Originality/Value: *This research work may add value to the researcher's work. We provide the information on creating a workable 3D printer as simple as possible in the CoppeliaSim robot simulator. Customizing some physical parameters can be a tool to help the research work rapidly.*

Paper Type: Simulation-based Research.

Keywords: 3D printer simulator, Virtual 3D printer, 3D Printer in CoppeliaSim

1. INTRODUCTION :

Nowadays, 3D printers are used in different domains. The various branded 3D printers are available in the market—the researchers are adding more features to it day by day. Starting from tiny toys to large boats are also tested for manufacturability in the researcher's lab. The researchers are trying to cover more domains. So it is now the market window for the 3D Printer. Sometimes researchers need a simulated 3D printer for their experiment. Not only testing their algorithm, to demonstrate their developed algorithm. The actual 3D Printer is not possible to carry most of the time. Keeping in our mind, we started to search for a workable virtual 3D printer that could act as a basic 3D printer. But lots of searching, we didn't get an excellent simulated 3D printer. So we researched it and created our virtual 3D Printer. We selected CoppeliaSim, which is a good simulator as of now. Lots of experiments we completed using this simulator. So we were confident about that. In this paper, we demonstrate our research work. We explain how we developed a 3D printer for research purposes. We present the most common functionality here using a simple square drawing example. Using this template, the researcher can add more functions according to their needs.

2. RELATED WORKS :

Tursynbek, I. et al. presents a new methodology for modeling and simulating spherical parallel manipulators (SPM) in CoppeliaSim (V-REP) robot simulator software [1]. J. Braker et al. presents a prototype of an innovative robotic cart using radio frequency (RF) signal strength for navigation. The proposed automated cart is cost-effective since it utilizes analog signal strengths to localize and follow a customer [2]. C. E. Magrin et al. design a mobile robot; the various development and testing stages, project costs, and multidisciplinary knowledge discourage students and researchers from starting on such tasks [3]. A. Vivas et al., in their paper, shows the implementation of control of a real UR5 robot from Matlab/Simulink using ROS. The desired trajectories are designed in Matlab tested in Simulink. The ROS Toolbox sends the trajectories from Simulink to ROS, using a Python file to make the connection. The robot simulator CoppeliaSim is used to test the robot's movement before sending it to the real manipulator [4]. F. Ulurrasyadi et. al. Presents a fast and simple learning algorithm for humanoid robot walking. They propose a rule-based learning method that has never been used in this walking gait learning case [5]. G. A. Al et. Al. Their paper presents a multimodal sensor interface capable of recognizing hand gestures for human-robot interaction. The proposed system is composed of an array of proximity and gesture sensors, which have been mounted on a 3D printed bracelet [6]. H. İ. Şahin et. Al. An autonomously controlled intelligent wheelchair is an electric wheelchair system that can reach the desired position by finding paths without user intervention. Pathfinding can be performed in outdoor areas with the support of a global positioning system (GPS), while interior mapping is required for indoor regions [7]. S. Zhu et al. focus on the control of a simulated robot hand using a data glove, which gathers information from resistors that measure the flexion of the user's fingers [8]. J. Buzzatto et al., in their paper, propose a novel design of an agile, coaxial, omnidirectional rotor module that can move and exert forces in all directions without limitations on orientation and wiring. This independent rotor unit presents high mobility and manoeuvrability, high fight autonomy, and offers high potential for all-terrain robotic manipulation applications [9]. A. Chico et al. focus on solving the trajectory tracking problem for the UR5 robot using two different control strategies: a minimum norm PID and a controller based on linear algebra. Both controllers are designed from the UR5 kinematic model [10]. CoppeliaSim is used for demonstration of 6DF Robot [11], demonstration of Custom Robotic ARM [12], demonstration of Custom Robot using C# [13], and demonstration of forward and inverse kinematics [14].

3. OBJECTIVES :

The objective of the research work is to provide some reference information to the robot or 3D printer researcher so they can create a virtual 3D printer for their research purpose. Using this model, they can test or simulate the 3D Printer inside the virtual environment. Usually, creating a model is a time-consuming task. Need to learn software, search for suitable tools for the work, etc. Using the CoppeliaSim, we made this 3D printer model. We can build it using a little effort through the step-by-step procedure. We demonstrate the functionality of the 3D Printer drawing a square box inside the work area. This template may help them to add more functionality to their research need. We created it as simple as possible so it could be built quickly. We skip some details drawings that are present in the actual Printer. This model may be used for the functionality demonstration.

The 3D Printer is the movement of 3 Axis, that is, X, Y, Z. in the actual scenario, the model file presents the three axes' different positions. Reading the file, the axis movement send to the joint handler. Generally, the stepper motor is used for precision position navigation in the actual Printer. Here we didn’t create a standard handler in detail. Instead, we arranged the target partition to reach the same as natural. That makes us simple. The new researcher trying to understand 3D printers can help them create a new one or modify it and develop a custom model better.

Figure 1 depicts the block diagram of our research work. The state machine handler manages and coordinates all entities when the simulation runs in the LUA main thread. X-Axis, Y-Axis, and Z-axis modules are responsible for navigating home. After the home position is reached, the finite state machine starts drawing or creating the object. Here we made the box using the programming. Different approaches are available to do the same. In the actual scenario, from 3D creator software, create an object file which is consists of G-code. The G-code is the 3D printer command list. Reading the file,

the system sends it to the associated joint movement handler. Figure 2 is the block diagram of the different used objects in a hierarchical manner. The same object hierarchy is depicted in figure 3 in CoppeliaSim simulator scene hierarchy window. When creating the model, we need to correctly care about object parent-child relation. The code only works if all objects are aligned perfectly.

4. APPROACH AND METHODOLOGY :

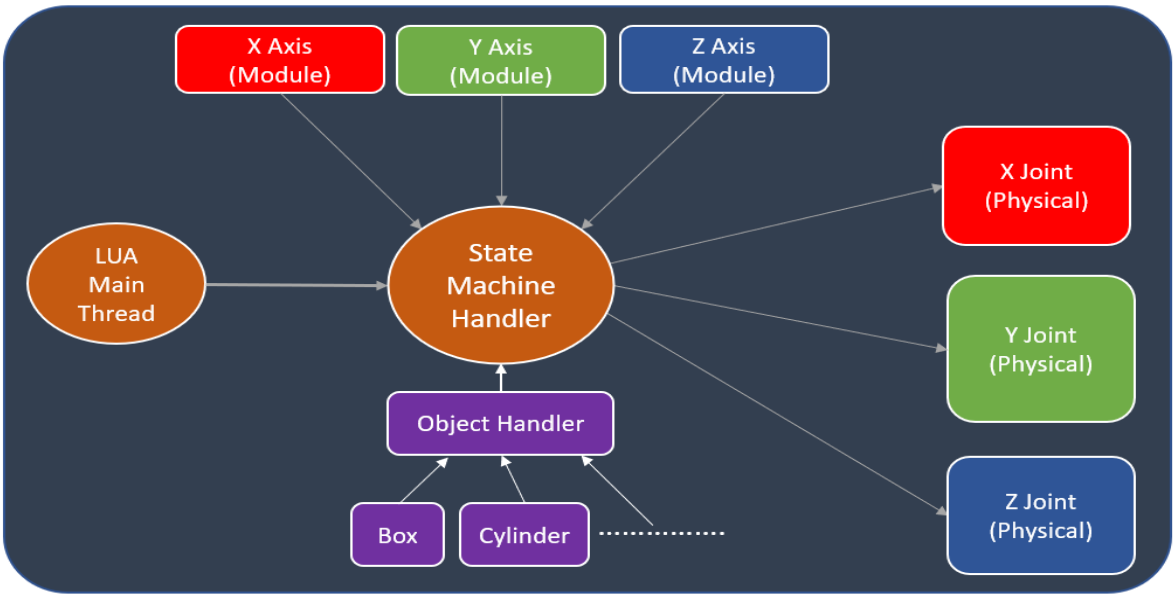

Fig. 1: Block diagram of virtual 3D Printer

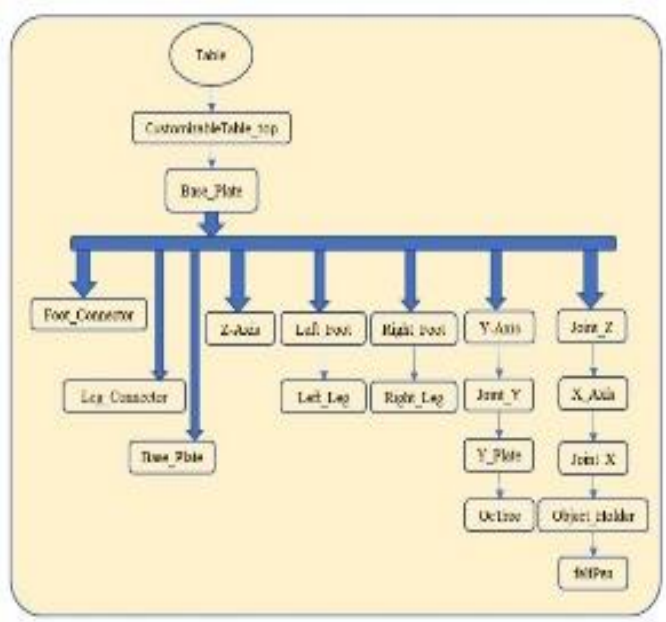

Fig. 2: Block diagram of used object

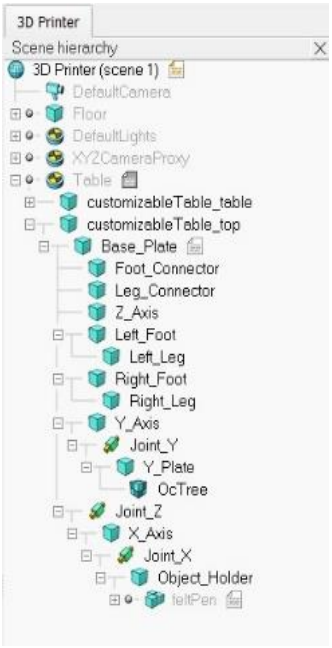

Fig. 3: Object hirerarchy

5. EXPERIMENT :

Let us do some experiments. We can follow as below:

1) Download and install the CoppeliaSim from https://www.coppeliarobotics.com/downloads.
2) Download project from GitHub. The download link is https://github.com/sudipchakraborty/A-simulated-3D-Printer-In-CoppeliaSim. Unzip the folder. We can see the 3D Printer.ttt file inside the folder—double-click on it. The file will open using the CoppeliaSim application depicted in figure 4.
3) A "RUN" button (triangle icon) is available inside the top toolbar. Press the button. Our 3D Printer will draw a box.
4) We can experiment with different designs or shapes.
5) Now we will see how we can create a new one from scratch, as figure 4.
6) Create a folder on the desktop. Rename "Test."
7) open CoppeliaSim from program/start menu. A blank scene will appear, as depicted in figure 5. Save the scene first. File>Save scene>select "Test" folder from the desktop > set filename : "test_scene"> press "Save" button.
8) First, we need a table to keep the Printer. On the left side of the window,> Model browser>furniture>click on tables> at the bottom, some table picture appears. Drag "Customizable table" to the workspace. Inside the "Scene hierarchy" window, double-click on "Customizable Table" and Rename it "Table." Click on the "Object/item shift" button from the top toolbar. Select "position" Tab>select "World" radio button> set value **X-cord[m]=0.00, Y-cord[m]=0.00, Z-cord[m]= 0.7.** Now the table is placed in the center of the workspace.
9) From Top menu bar > "Add" > "Primitive Shape" > click on "Cuboid" > press "OK". Rename "Base_Plate." Keep the "Base_Plate" selected. From left-side toolbar> Click "Scene object properties" button (Lense Icon) > click "View/modify geometry" button. Uncheck "Keep proportions" and modify the value. **X[m]=0.5, Y[m]=0.5, Z[m]=0.02**. Click on "Apply." Close the two recently opened windows. Expand the "Table" Object in the "Scene hierarchy" window. Drag the "Base_Plate" and drop on "customizable_Table_top." Now, the "customizable_Table_top" is the parent of the "Base_Plate" object. Keep "Base_Plate" selected, using "Object/item shift" button, select "Parent frame" radio button and set the value **X-cord[m]=0.00, Y-cord[m]=0.00, Z-cord[m]= 0.05**. close the window. We can change the color of the "Base_Plate." Click on "Scene object properties" button> click on "Adjust color" button> click on "Ambient/diffuse component" > and set the value **R=0.69, G=0.35, B= 0.28**. close three small opened windows. Our base plate is on the table. Now we will create the next part.
10) Add a cuboid. Rename "**Left_Foot**". Set parameter as below:
    Size : X[m]=0.04, Y[m]=0.29, Z[m]=0.04
    Parent frame: Base_Plate
    Position: X-cord[m]=-0.125, Y-cord[m]=0.00, Z-cord[m]= 0.02
    Colour: R=0.1, G=0.1, B=0.1
11) Add a cuboid. Rename "**Right_Foot**". Set parameter as below:
    Size : X[m]=0.04, Y[m]=0.29, Z[m]=0.04
    Parent frame : Base_Plate
    Position: X-cord[m]=0.125, Y-cord[m]=0.00, Z-cord[m]= 0.02
    Colour: R=0.1, G=0.1, B=0.1
12) Add a cuboid. Rename "**Foot_Connector**". Set parameter as below:
    Size : X[m]=0.25, Y[m]=0.04, Z[m]=0.04
    Parent frame: Base_Plate
    Position: X-cord[m]=0.0, Y-cord[m]=0.00, Z-cord[m]= 0.02
    Colour: R=0.1, G=0.1, B=0.1
13) Add a cuboid. Rename "**Left_Leg**". Set parameter as below:

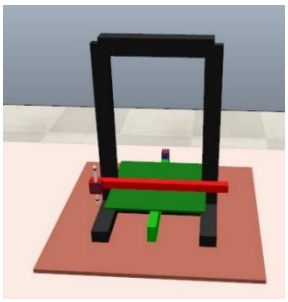

Fig. 4: 3D printer

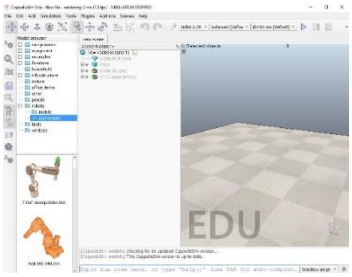

Fig. 5: Blank Workspace

Size : X[m]=0.04, Y[m]=0.02, Z[m]=0.4
Parent frame: Left_Foot
Position: X-cord[m]=0.0, Y-cord[m]=0.0, Z-cord[m]= 0.22
Colour: R=0.1, G=0.1, B=0.1

14) Add a cuboid. Rename "**Right_Leg**". Set parameter as below:
Size : X[m]=0.04, Y[m]=0.02, Z[m]=0.4
Parent frame: Right_Foot
Position: X-cord[m]=0.0, Y-cord[m]=0.00, Z-cord[m]= 0.22
Colour: R=0.1, G=0.1, B=0.1
15) Add a cuboid. Rename "**Leg_Connector**". Set parameter as below:
Size : X[m]=0.25, Y[m]=0.02, Z[m]=0.04
Parent frame: : Base_Plate
Position: X-cord[m]=0.0, Y-cord[m]=0.00, Z-cord[m]= 0.435
Colour: R=0.1, G=0.1, B=0.1
16) Add Joint>Prismatic. Rename "**Joint-Z**". Set parameter as below:
Size : Length[m]=.025, Diameter[m]=0.02,
Pos. min.[m]=0.0, Pos. range[m]=1.00, Position[m]=0.00
Parent frame: : Base_Plate
Position: X-cord[m]=-0.125, Y-cord[m]=0.03, Z-cord[m]= 0.05
Colour A: R=.41, G=0.17, B=0.25
17) Add a cuboid. Rename "**Z-Axis**". Set parameter as below:
Size : X[m]=0.025, Y[m]=0.025, Z[m]=0.34
Parent frame: : Base_Plate
Position: X-cord[m]=-0.125, Y-cord[m]=0.03, Z-cord[m]= 0.235
Colour: R=0, G=0, B=0.66
18) Add a cuboid. Rename "**X-Axis**". Set parameter as below:
Size : X[m]=0.34, Y[m]=0.02, Z[m]=0.02
Parent frame: : Joint_Z
Position: X-cord[m]=0.125, Y-cord[m]=-0.052, Z-cord[m]= 0.048
Colour: R=0.9, G=0.0, B=0.0
19) Add Joint>Prismatic. Rename "**Joint-X**". Set parameter as below:
Size : Length[m]=.025, Diameter[m]=0.02,
Pos. min.[m]=0.0, Pos. range[m]=1.00, Position[m]=0.00
Parent frame: : X-Axis
Position: X-cord[m]=-0.15, Y-cord[m]=-0.02, Z-cord[m]= 0.0
Colour A: R=.41, G=0.17, B=0.25
20) Add a cuboid. Rename "**Object-Holder**". Set parameter as below:
Size : X[m]=0.03, Y[m]=0.02, Z[m]=0.03
Parent frame: : Joint-X
Position : X-cord[m]=0.0, Y-cord[m]=0.0, Z-cord[m]= 0.0
Colour: R=0.57, G=0.0, B=0.0
21) Model browser>components>modifier>felt pen. rename "**felt-Pen**". Set parameter as below:
Parent frame: : Object-Holder
Position: X-cord[m]=0.0, Y-cord[m]=0.00, Z-cord[m]= 0.0056
22) Add a cuboid. Rename "**Y-Axis**". Set parameter as below:
Size : X[m]=0.02, Y[m]=0.35, Z[m]=0.04
Parent frame: : Base_Plate
Position: X-cord[m]=0.0, Y-cord[m]=0.00, Z-cord[m]= 0.06
Colour: R=0.0, G=.7, B=0
23) Add Joint>Prismatic. Rename "**Joint-Y**". Set parameter as below:
Size : Length[m]=.025, Diameter[m]=0.02,
Pos. min.[m]=0.0, Pos. range[m]=1.00, Position[m]=0.00
Parent frame: : Y-Axis
Position : X-cord[m]=0.00, Y-cord[m]=0.16, Z-cord[m]= 0.03
Colour A: R=.41, G=0.17, B=0.25
24) Add a cuboid. Rename "**Y-Plate**". Set parameter as below:

Size : X[m]=0.233, Y[m]=0.233, Z[m]=0.005
Parent frame: : Joint-Y
Position: X-cord[m]=0.00, Y-cord[m]=-0.005, Z-cord[m]= 0.000
Colour: R=0.0, G=0.4, B=0.0

25) Right click on workspace>Add> OC tree. Set parameter as below:
Parent frame: : Y-Plate
Position: X-cord[m]=0.0, Y-cord[m]=0.00, Z-cord[m]= 0.00

26) Now we need to set rotation configuration of the below objects. Select object. from the top toolbar click on "Object/item rotate." Select the "Orientation" Tab. Select "Parent frame." Set value for Alpha, Beta, and Gamma. As below:
Joint-X: Alpha[deg.]=**00.0**, Beta[deg.]=**90.0**,Gamma[deg.] =**00.0**.
Joint-Y: Alpha[deg.]=**90.0**, Beta[deg.]=**00.0**,Gamma[deg.] =**00.0**.
Y-Plate: Alpha[deg.]=-**90.0**, Beta[deg.]=**00.0**,Gamma[deg.] =**00.0**.
Joint-Z: Alpha[deg.]=**00.0**, Beta[deg.]=**00.0**,Gamma[deg.] =**0.00**.
Object_Holder: Alpha[deg.]=**00.0**, Beta[deg.]=-**90.0**,Gamma[deg.] =**0.00**.

27) If we run by pressing the run button we, will observe that all components have broken. By default, all objects are set as dynamic. Now we need to disable the dynamic behavior of everything. Whatever we created an object, Select object one by one> "Scene object properties" > press "Show dynamic properties dialog" button> uncheck "Body is dynamic."

28) We need to add the LUA scripts for the operational 3D Printer. Select Base_Plate. Right-click on Base_Plate > "Add"> "Associated child script"> "Non threaded." On the right side of the name, a script file is added—the path to adding the script depicted in figure 6.

29) Open Base_Plate.Lua(project download folder) in note pad or Visual studio code and copies the complete code. Paste the code inside the newly created script file associated with Base_Plate.

30) Copy the Axis_X. Lua, Axis_Y.Lua, Axis_Z.Lua, and box.Lua files (available in download folder) and paste inside the current simulator folder(\Desktop\Test). Now run the simulator. We can observe that our 3D Printer is printing a square box.

31) We can add an image on our Y-plate for better understanding, depicted in figure 7 to add an image need to follow the step. Select Y-Plate> click on "Scene object properties" button> click on "Adjust texture" button> click on "Load new texture" browse project folder > select "CAL_Pattern_Square.jpg >" Open"> select texture scale max. 2048 x 2048 radio button> "OK".

32. Using this pen, we can draw on the plate also. For this, we need the below steps. Select "Y-Plate"> click on "Scene object properties"> click on "Common" Tab > under the "Object special properties" group box > checked on "Detectable" checkbox. In the nib of the felt pen, one

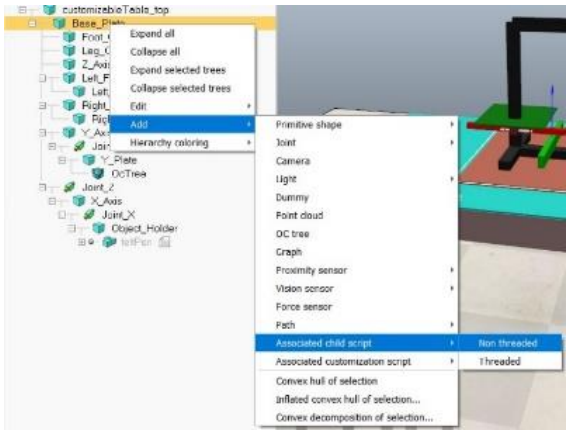

Fig. 6: Script addition

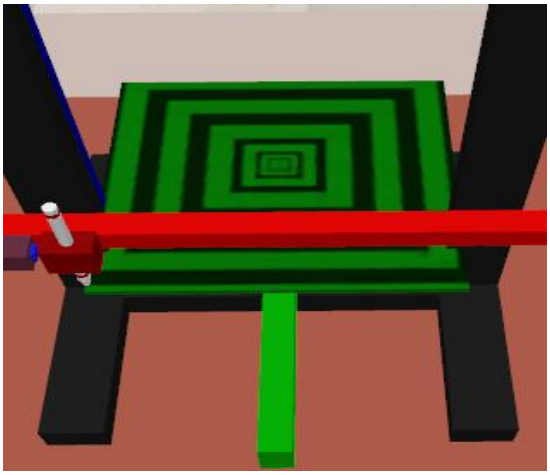

Fig. 7: Pattern addition

proximity sensor is attached. It will draw when any detectable object is found near the proximity sensor. So, when the felt pen reaches near the Y-plate, it will mark the line.

Sl. No.	Object Type	Object Name	Size		
			X/Length[m]	Y/Diameter[m]	Z[m]
1	Customizable Table	Table	Default		
2	Cuboid	Base_Plate	0.5	0.5	0.02
3	Cuboid	Left_Foot	0.04	0.29	0.04
4	Cuboid	Right_Foot			
5	Cuboid	Foot_Connector	0.25	0.04	0.04
6	Cuboid	Left_Leg	0.04	0.02	0.4
7	Cuboid	Right_Leg			
8	Cuboid	Leg_Connector	0.25	0.02	0.04
9	Joint>Prismatic	Joint_Z	0.025	0.02	
10	Cuboid	Z-Axis	0.025	0.025	0.34
11	Cuboid	X-Axis	0.34	0.02	0.02
12	Joint>Prismatic	Joint-X	0.025	0.02	
13	Cuboid	Object-Holder	0.03	0.02	0.03
14	Model browser> modifier> felt pen	felt-Pen			
15	Cuboid	Y-Axis	0.02	0.35	0.04
16	Joint>Prismatic	Joint_Y	0.025	0.02	
17	Cuboid	Y-Plate	0.233	0.233	0.005
18	Add>OC tree	OcTree			

Fig. 8: List of used object's Size

Sl. No.	Object Name	Position			Relative to
		X-cord.[m]	Y-cord.[m]	Z-cord.[m]	
1	Table	0.00	0.00	0.7	World
2	Base_Plate	0.00	0.00	0.05	Table_top
3	Left_Foot	-0.125	0.000	0.02	Base_Plate
4	Right_Foot	0.125			
5	Foot_Connector	0.00	0.000		Base_Plate
6	Left_Leg	0.00	0.000	0.22	Left_Foot
7	Right_Leg	0.00	0.000	0.22	Right_Foot
8	Leg_Connector	0.00	0.000	0.435	Base_Plate
9	Joint_Z	-0.125	0.03	0.05	Base_Plate
10	Z-Axis	-0.125	0.03	0.235	Base_Plate
11	X-Axis	0.125	-0.052	0.103	Joint_Z
12	Joint_X	-0.15	-0.02	0.00	X_Axis
13	Object_Holder	0.00	0.00	0.035	Joint_X
14	feltPen	0.0000	0.0000	0.0056	Object_Holder
15	Y-Axis	0.000	0.000	0.06	Base_Plate
16	Joint_Y	0.00	0.16	0.03	Y-Axis
17	Y-Plate	0.000	0.01	0.092	Joint-Y
18	OcTree	0.000	0.000	0.000	Y_Plate

Fig. 9: List of used object position and relative

Child script (Base_Plate)

```
local x= require("Axis_X")
local y = require("Axis_Y")
local z=require("Axis_Z")
local box=require("box")

FSM="Wait_For_Home"
x_val=0
y_val=0
z_val=0
count_idle=0;
start_drawing=0
-----------------------------------------
function sysCall_init()

   Joint_X=sim.getObjectHandle('Joint_X')
   Joint_Y=sim.getObjectHandle('Joint_Y')
   Joint_Z=sim.getObjectHandle('Joint_Z')
   octree=sim.getObjectHandle("OcTree")

   x.set_default(0.04,.25,0.0005)
   y.set_default(0.07,.25,0.0005)
   z.set_default(0.0545,.25,0.0005)
   box.init(.2,.1,.1)
end
```

Fig. 10: Base_Plate script , part-1

The advanced researcher, who knows CoppeliaSim, can use figures 8 and 9 for rapid prototype development. Whatever we follow, the procedure in descriptive form is also available in tabular form. Figures 10 and 11 are the code for Base_Plate scripts. The same code is available in the project folder. The researcher needs to copy and paste inside the script file. We can change the filament/line width. The below procedure helps us to set the OcTree properties settings. Select "OcTree" from the "Scene hierarchy" window> click "Scene object properties" button> One window will open as depicted in figure 12. under the "Main properties" group box, there is a "Voxel size[m]" textbox. According to our need, we can set the size of the filament. In some experiments, if we need the random color of the filament from the "visual properties group box, we need to check on "Voxels have random colors." Whatever we draw into the "OC tree" is saved even if the simulation stop or closes. We can clear by force. The "Clear octree" removes the voxel content of the OC tree. Another feature is that if we want to draw the point instead of the voxel, we checked on "Show points instead of voxels" and set the "point size" as our need. Now our experiment is complete.

```
function sysCall_actuation()

  -----------------------------
  if(FSM=="Wait_For_Home") then
      x_val=x.pos()
      y_val=y.pos()
      z_val=z.pos()
      print("System is Homing...")
          if((x.idle()) or (y.idle()) or (z.idle())) then
              print("System is Now in Home")
              sim.removeVoxelsFromOctree(octree,0,nil)
              start_drawing=1
              box.set_pos(x_val,y_val,z_val)
              FSM="trigger"
          end
  -----------------------------
  elseif(FSM=="trigger") then
  x_val,y_val,z_val=box.pos()
   --  x.target(0.25)
  --  x.set_command("go")
   --   FSM="Waiting_For_Completion"
  -----------------------------
   elseif(FSM=="Waiting_For_Completion") then
          print("Waiting for completion...")
          if(x.idle()) then
              FSM="xx"
          end
  -----------------------------
  else
  print("System is in Idle State>",count_idle)
  count_idle=count_idle+1
  end
  ---------------------------------
  print("x=",x_val,"Y=",y_val,"Z=",z_val)

  sim.setJointPosition(Joint_X,x_val)
  sim.setJointPosition(Joint_Y,y_val)
  sim.setJointPosition(Joint_Z,z_val)
  p={x_val-.14,y_val-.05,z_val+.7}

  if(start_drawing==1)then
    sim.insertVoxelsIntoOctree(octree,0,p)
    end
end
-------------------------------------------------------------
function sysCall_sensing()
end
-------------------------------------------------------------
function sysCall_cleanup()
   sim.removeVoxelsFromOctree(octree,0,nil)

end
```

Fig. 11: Base_Plate script , part-2

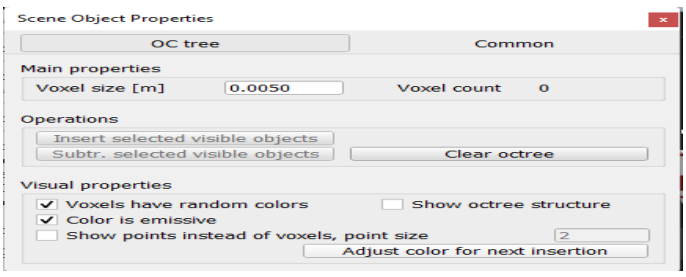

Fig. 12: OC Tree Properties Window

6. RECOMMENDATIONS :

- Our research paper on CoppeliaSim, "A Custom Robot" [12] can help basic coppeliaSim UI element introduction in the project folder.
- We can test and implement the 3D Printer G-Code. One good information link is https://www.simplify3d.com/support/articles/3d-printing-gcode-tutorial/
- The complete 3D printer command list is not integrated here. We can implement it on our needs.
- The researcher can control this Printer from outside using G-Code or custom code through TCP/IP socket communication.
- All parameters are not calibrated. We need to calibrate for a more accurate result.
- Using this project, our researcher can create digital twins.
- This application is not entirely bug-free. We debugged what we got at experiment time. May persists several bugs, which can be debugged under more tests.
- We developed core functions only. The researcher can add additional functionality for better usability.
- We can easily create a different simulated model to demonstrate some algorithm or processes using the same procedure.

7. CONCLUSION :

Nowadays, 3D Printer is common to all. We demonstrated how to create a virtual 3D printer that can be used for research purposes or some functionality demonstration without purchasing an actual one through the step-by-step procedure. A good 3D printer is costly and takes lots of physical space. Also, lots of issues to carry anywhere. There is no justification for buying a brand new 3D printer, only to study the functionality or demonstration. So through this research work, we can demonstrate the functionality of a 3D printer without having real hardware. Instead of carrying an actual 3D printer, we can use this virtual 3D printer for demonstration.

REFERENCES :

[1] Tursynbek, I. and Shintemirov, A. (2020). Modeling and Simulation of Spherical Parallel Manipulators in CoppeliaSim (V-REP) Robot Simulator Software. International Conference Nonlinearity, Information and Robotics (NIR), 2020, pp. 1-6, DOI: 10.1109/NIR50484.2020.9290227.
Google Scholar↗

[2] Braker, J., Beebe, D., Allen, K., Shastry, P. and Miah, M. S. (2021). A Smart Robotic Cart Prototype using RF Signal Strength. IEEE International Symposium on Robotic and Sensors Environments (ROSE), 2021, pp. 1-6, DOI: 10.1109/ROSE52750.2021.9611760.
Google Scholar↗

[3] Magrin, C. E., Del Conte, G. and Todt, E. (2021). Creating a Digital Twin as an Open Source Learning Tool for Mobile Robotics. Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE), 2021, pp. 13-18, DOI: 10.1109/LARS/SBR/WRE54079.2021.9605457.
Google Scholar↗

[4] Vivas, A. and Sabater, J. M. (2021). UR5 Robot Manipulation using Matlab/Simulink and ROS. IEEE International Conference on Mechatronics and Automation (ICMA), 2021, pp. 338-343, DOI: 10.1109/ICMA52036.2021.9512650.
Google Scholar↗

[5] Ulurrasyadi, F., Dewanto, R. S., Barakbah, A. and Pramadihanto, D. (2021). Walking Gait Learning for "T-FLoW" Humanoid Robot Using Rule-Based Learning. International Electronics Symposium (IES), 2021, pp. 527-531, DOI: 10.1109/IES53407.2021.9593960.
Google Scholar↗

[6] Al, G. A. Estrela, P. and Martinez-Hernandez, U. (2020). Towards an intuitive human-robot interaction based on hand gesture recognition and proximity sensors. IEEE International

Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2020, pp. 330-335, DOI: 10.1109/MFI49285.2020.9235264.
Google Scholar↗

[7] Şahin H. İ. and Kavsaoğlu, A. R. (2021). Autonomously Controlled Intelligent Wheelchair System for Indoor Areas. 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2021, pp. 1-6, DOI: 10.1109/HORA52670.2021.9461335.
Google Scholar↗

[8] Zhu, S. Stuttaford-Fowler, A. Fahmy, A. Li, C. and Saenz, J. (2021). Development of a Low-cost Data Glove using Flex Sensors for the Robot Hand Teleoperation. 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT), 2021, pp. 47-51, DOI: 10.1109/ISRIMT53730.2021.9596972.
Google Scholar↗

[9] Buzzatto J. and Liarokapis, M. (2020). An Agile, Coaxial, Omnidirectional Rotor Module: On the Development of Hybrid, All Terrain Robotic Rotorcrafts. IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2020, pp. 162-168, DOI: 10.1109/SSRR50563.2020.9292586.
Google Scholar↗

[10] Chico, A. et al., (2021). Hand Gesture Recognition and Tracking Control for a Virtual UR5 Robot Manipulator. IEEE Fifth Ecuador Technical Chapters Meeting (ETCM), 2021, pp. 1-6, DOI: 10.1109/ETCM53643.2021.9590677.
Google Scholar↗

[11] Chakraborty, S., & Aithal, P. S. (2021). Forward Kinematics Demonstration of 6DF Robot using CoppeliaSim and C#. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 29-37.
Google Scholar↗

[12] Chakraborty, S., & Aithal, P. S. (2021). A Custom Robotic ARM in CoppeliaSim. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 38-50.
Google Scholar↗

[13] Chakraborty, S., & Aithal, P. S. (2021). An Inverse Kinematics Demonstration of a Custom Robot using C# and CoppeliaSim. *International Journal of Case Studies in Business, IT, and Education (IJCSBE)*, *5*(1), 78-87.
Google Scholar↗

[14] Chakraborty, S., & Aithal, P. S. (2021). Forward and Inverse Kinematics Demonstration using RoboDK and C#. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 97-105.
Google Scholar↗

****