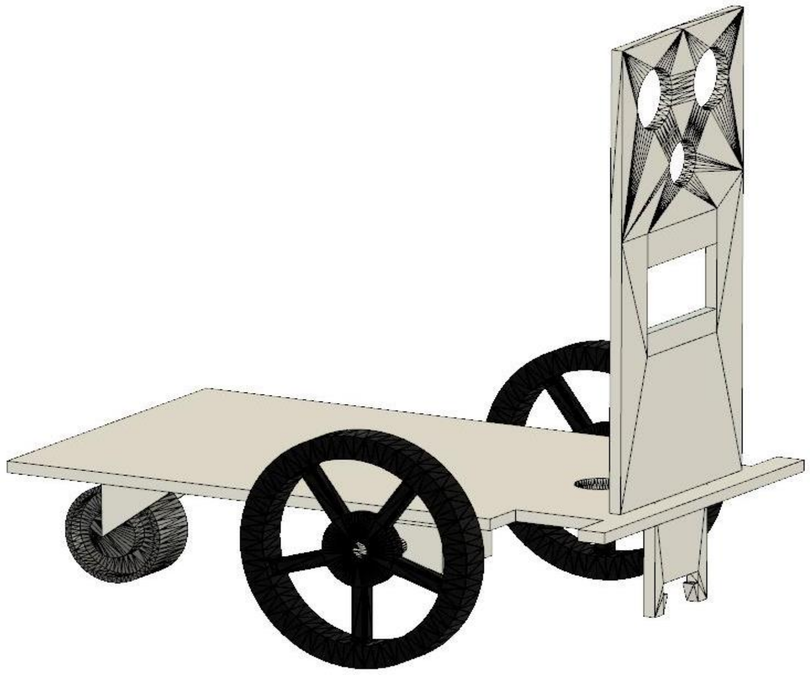
Project # 01

3D Modeling and Simulation of a Mobile Robot



Submitted to: Dr. Yasar Ayaz

Submitted by: Usman Zaheer

Registration Number: 327700

Subject: Mobile Robotics

Contents

1. Problem Statement:

Submit a report of your project describing what sort of robot simulation you will develop and using which tools. Highlight key deliverables and Novel ideas and give references from literature.

The scope of this document is to give information about 3D modelling of a mobile robot (wheeled robot) and its movement simulation using modelling software Autodesk Fusion 360 and simulation software CopppleliaSim (V-rep).

2. Selection and Design of Wheeled Robot:

Wheeled robots are widely used all around the world based on their applications. Wheeled robots are robots that navigate around the ground using motorized wheels to propel themselves. They are easier to design, build, and program for movement in flat and not-so-rugged terrain.

The wheeled robot chosen for this project is composed of following simple parts:

- Two standard wheels
- One Caster wheel.
- Base plate (Chasis)
- Front Plate.
- Support for caster wheel.
- Caster wheel cover..
- Servos for wheels.

3. Modeling of Components:

The components are designed in Adobe Fusion 360 which is cloud based modelling software and very user friendly.

➢ Base plate design (Chassis):

The base plate is a simple plate consisting of a slot and circle for adjusting other components like front plate and caster wheel. It is simple design with a lot of space on it, so that other components can be installed on it. The dimensions of a base plate are:

- **Width:** 920mm
- **Height:** 1580mm
- **Thickness:** 30mm

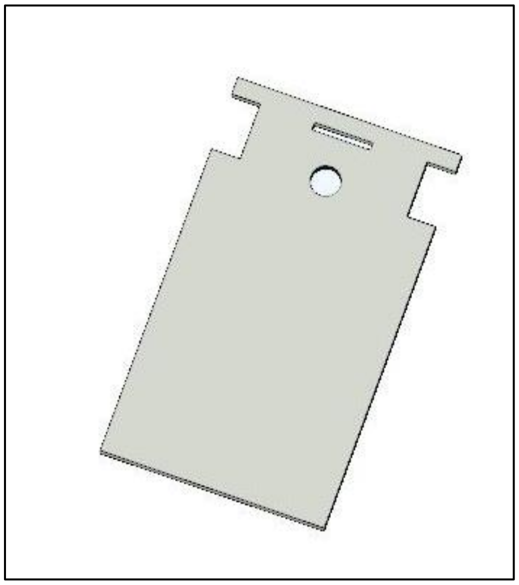


Figure 1: ***Base Plate(Chassis)***

➢ Front plate design:

The base plate is a simple plate consisting of a slot and circle for adjusting other components. It is simple design with cutouts, so that other components can be installed on it.

The dimensions of a front plate are:

- **Width:** 5600mm
- **Height:** 10mm
- **Thickness:** 30mm

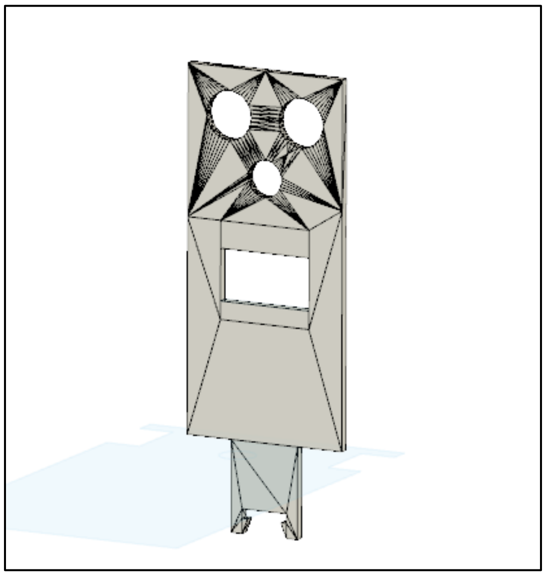


Figure 2: Front Plate

➢ Standard Wheels Design:

The standard wheel used in this project are simple wheels with cut outs for giving aesthetic look. The wheels used on both sides are same and are rotated using servo's.

The dimensions of a wheels are:

- **Dia:** 600 mm
- **Thickness:** 80mm

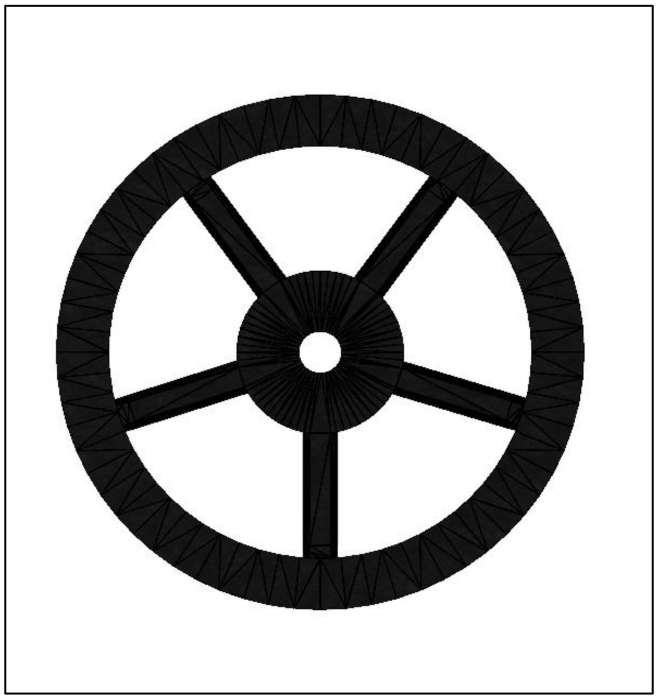


Figure 3: Wheel Design

➢ Castor Wheels Design:

The castor wheel used in this project is solid wheel. It is free to move around the caster axis jointed to body by caster support.

The dimensions of a wheels are:

- **Dia:** 250 mm
- **Thickness:** 130mm

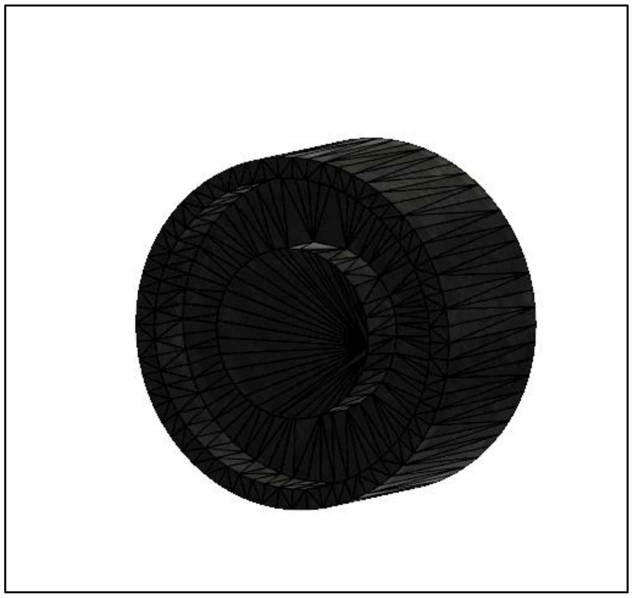


Figure 4: Caster Wheel

➢ Castor Wheels Support:

The castor wheel support is used to attach the caster wheel with the body. It allows the caster wheel to move around the caster axis.

The dimensions are:

- **Length:** 380mm
- **Height:** 380mm mm
- **Thickness:** 10mm

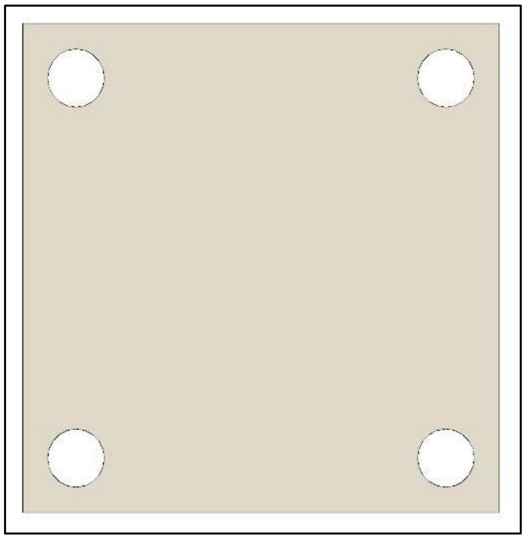


Figure 5: Caster Wheel Support

➢ Castor Wheel Side Cover:

The castor wheel cover is used to protect the caster wheel. It allows the caster wheel to attach the wheel around the caster axis and the centre axis.

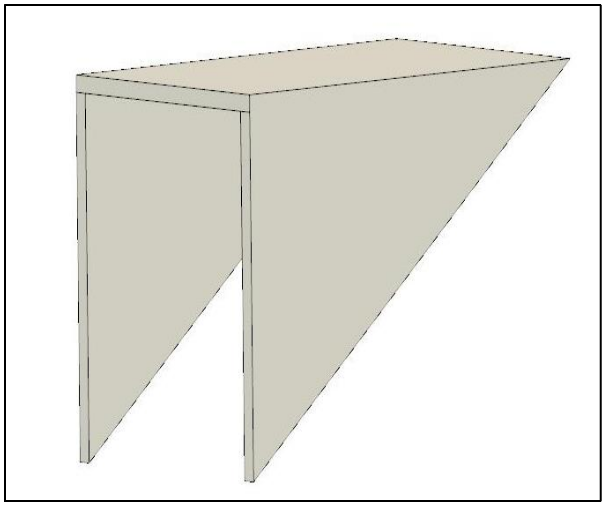


Figure 6: Caster Wheel Side Cover

4. Assembling Components:

All the components designed in the fusion 360 are assembled to get the complete assembly of the robot. First all the individual components are imported one by one and then joints are made to get the complete assembly.

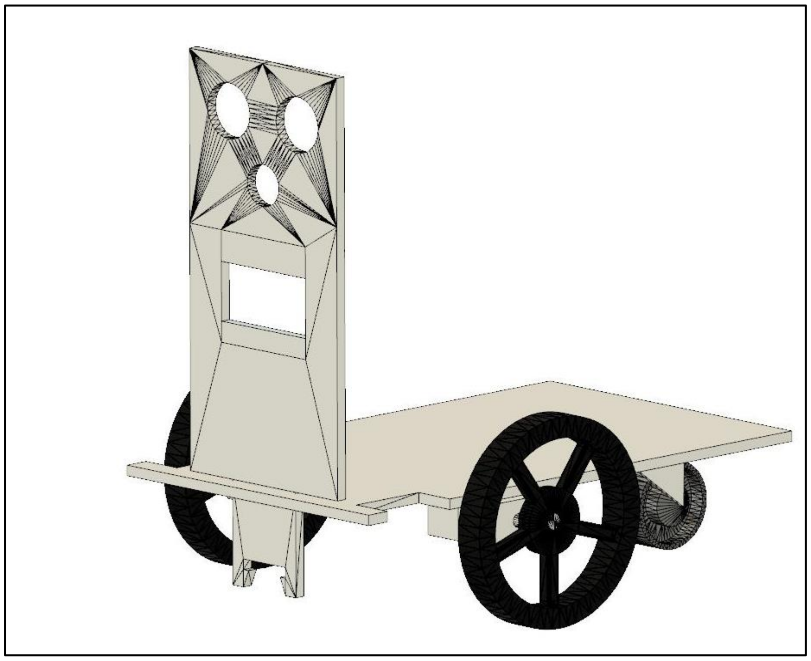


Figure 7: Complete Assembly

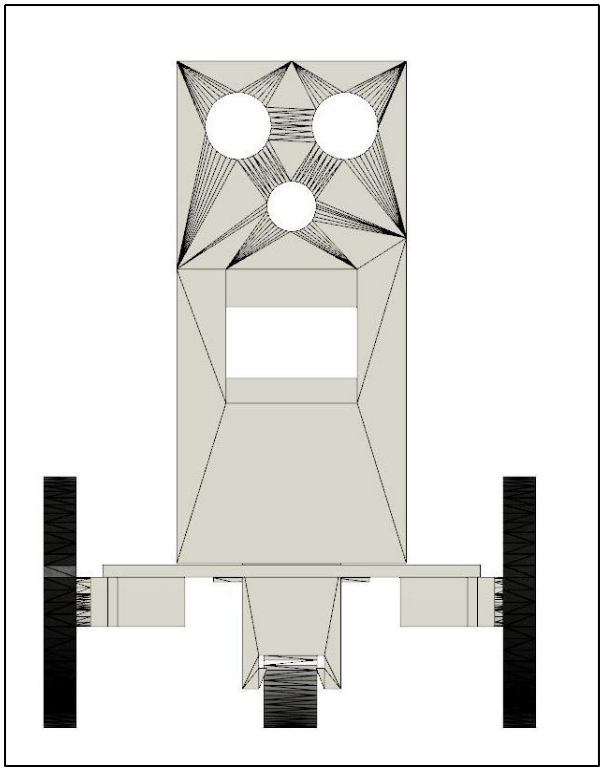


Figure 8: Front View

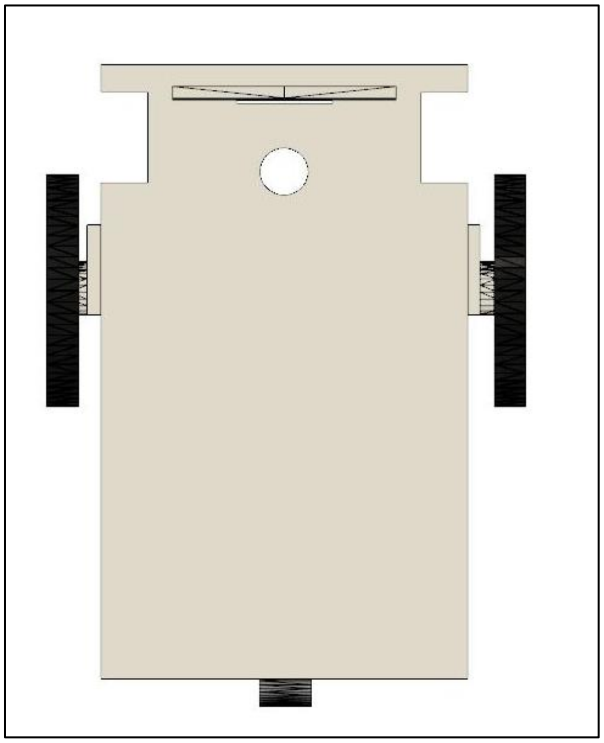


Figure 9: Top View

5. Conversion of components:

The individual components are then converted into STL files also know as Mesh files to be imported into the simulation software also known as CoppeliaSim or V-rep.

The mesh files are then uploaded into the CoppeliaSim to perform the simulation of this robot.

6. Simulation of Robot:

The following steps will be done in the CoppeliaSim to complete the simulation of this robot:

1) Importing mesh files.
2) Creating pure bodies from mesh files.
3) Grouping the components after making pure bodies.
4) Making revolute joints to the components like front wheels and caster wheel.
5) Setting hierarchy of components.
6) Writing Kinematics of robot in LUA script and simulating the robot using this script.

➢ Importing mesh files:

The individual converted STL file of each converted is then uploaded into the CoppeliaSim to perform the simulation of this robot. The CoppeliaSim also know as Vrep is the robotics simulation software that is used in this project. The assembly of the model in the CoppeliaSim environment is given below:

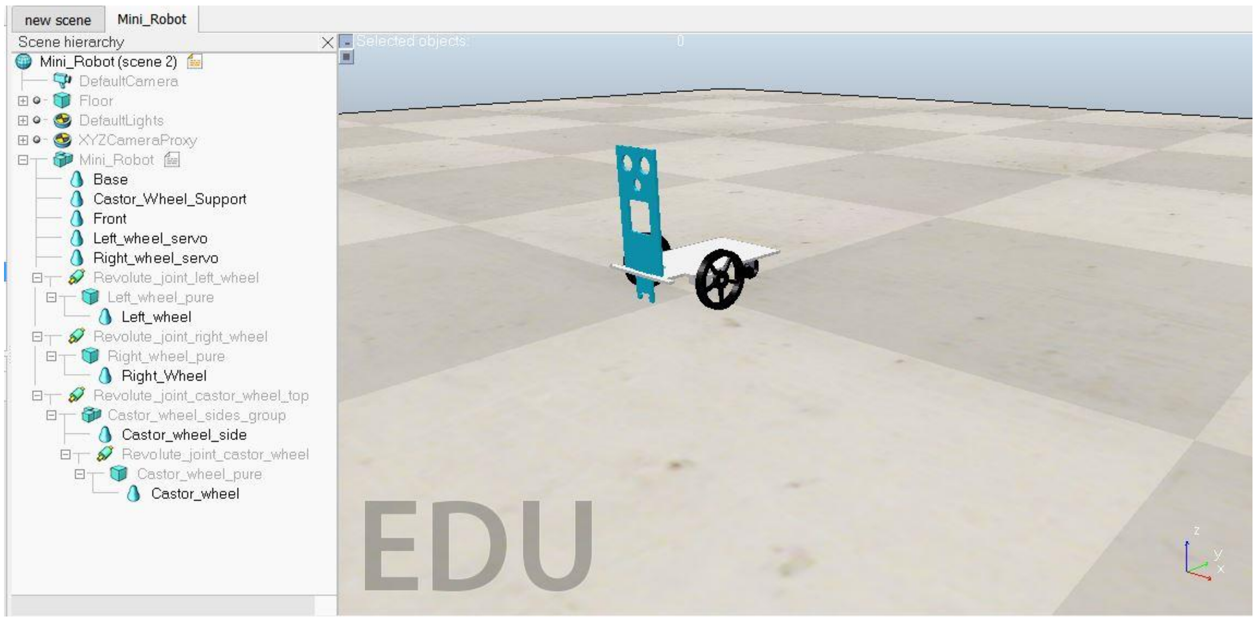


Figure 10: Mini Robot in CoppeliaSim

➢ Converting mesh files into pure shapes:

As the files imported in the simulation software environment are stl or mesh files, therefore for the simulation purpose the individual components are needed to be converted into pure shapes like cuboid and cylinder so that the simulation can be done successfully. Because mesh files are only for the import and representation purposes.

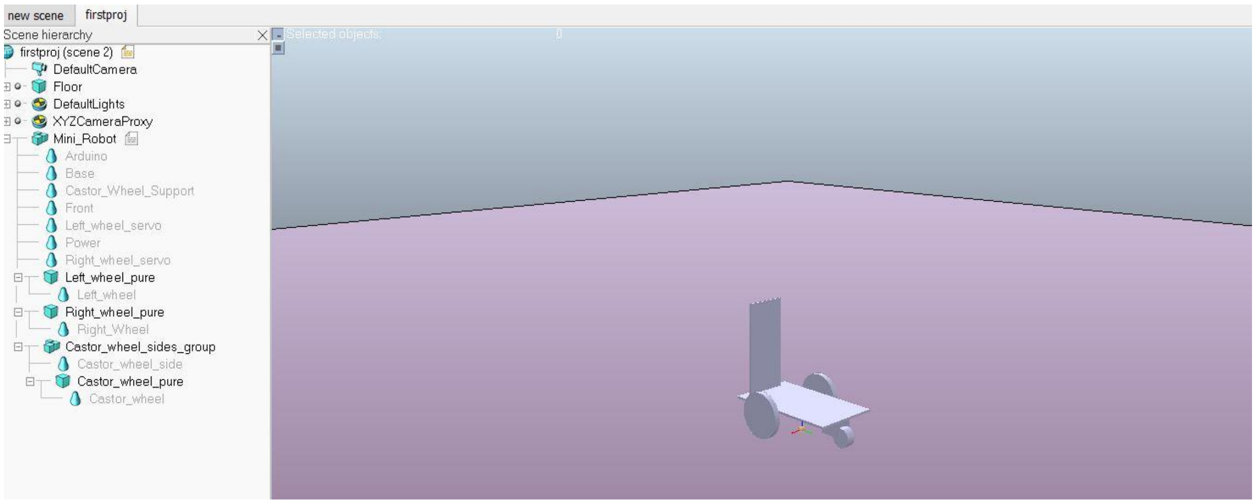


Figure 11: Pure Shapes of components in CoppeliaSim

➢ Making joints of front wheels and caster wheel:

After converting the all the components into pure shapes for the simulation purpose, the next step is to make the joints for the front wheels and the caster wheel. For this purpose, the components were selected to make the revolute joint between the wheels and the body. Four revolute joints were made in this model, 02 for the front wheels and 2 for the caster wheel (1 with caster wheel and 1 with caster wheel support bracket with chasis).

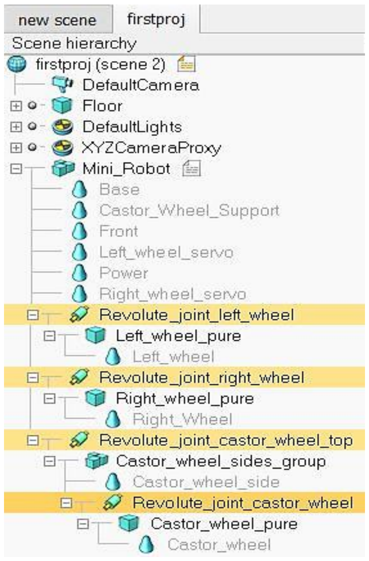


Figure 12: Revolute Joints

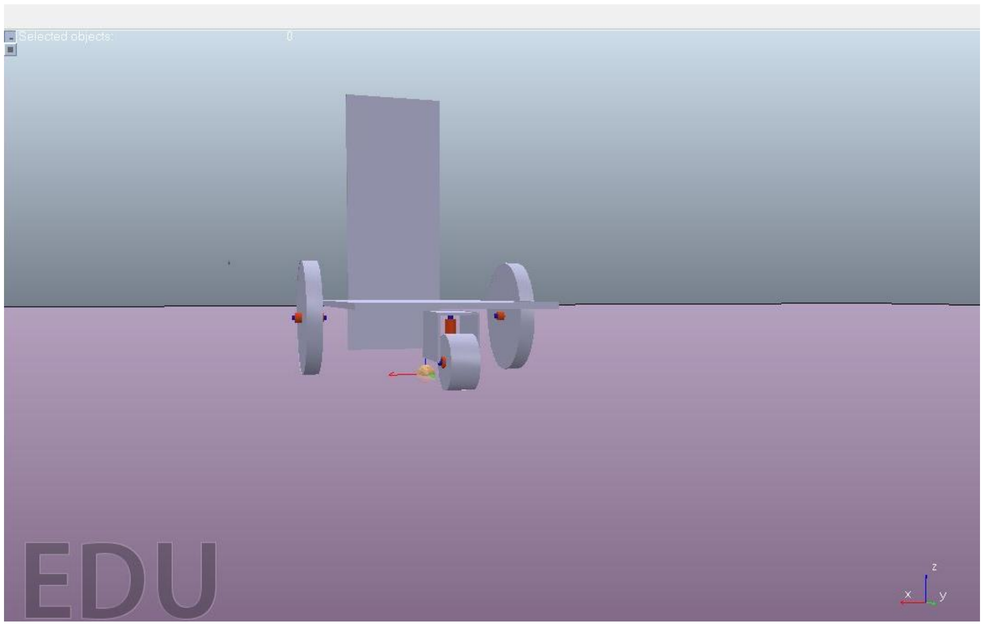
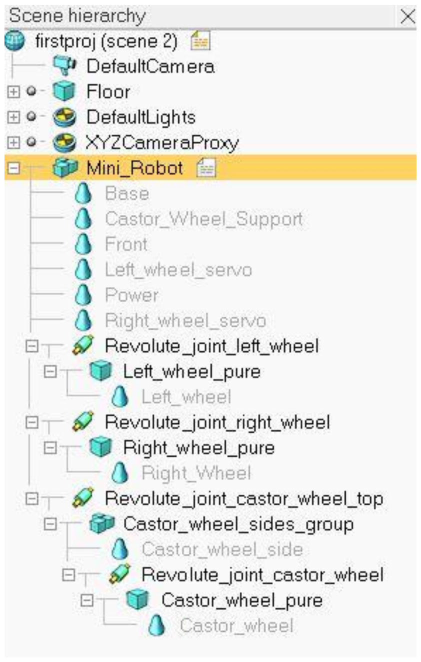


Figure 13: Revolute Joints between components.

➢ Setting the hierarchy of components:

After creating the revolute joints and setting the dynamic properties of all revolute joints, the hierarchy of components is maintained so that the simulation can run successfully. If the hierarchy of components is not maintained, then the simulation will fail. The hierarchy is set by making the parent and child components linked to each other i.e. all components are the child of parent component Mini_Robot as shown in fig below:



```
Scene hierarchy
firstproj (scene 2)
  DefaultCamera
  Floor
  DefaultLights
  XYZCameraProxy
  Mini_Robot
    Base
    Castor_Wheel_Support
    Front
    Left_wheel_servo
    Power
    Right_wheel_servo
    Revolute_joint_left_wheel
      Left_wheel_pure
        Left_wheel
    Revolute_joint_right_wheel
      Right_wheel_pure
        Right_Wheel
    Revolute_joint_castor_wheel_top
      Castor_wheel_sides_group
        Castor_wheel_side
        Revolute_joint_castor_wheel
          Castor_wheel_pure
            Castor_wheel
```

Figure 14: Hierarchy of Components

➢ Writing Kinematics equations of robot in LUA script:

The final step before running the simulation is to write the Kinematics of wheeled robot in LUA script of the CoppeliaSim software. The wheel equations are written in the script and attached to the robot for the simulation. The script can be added using the associated customized script option of CoppeliaSim.

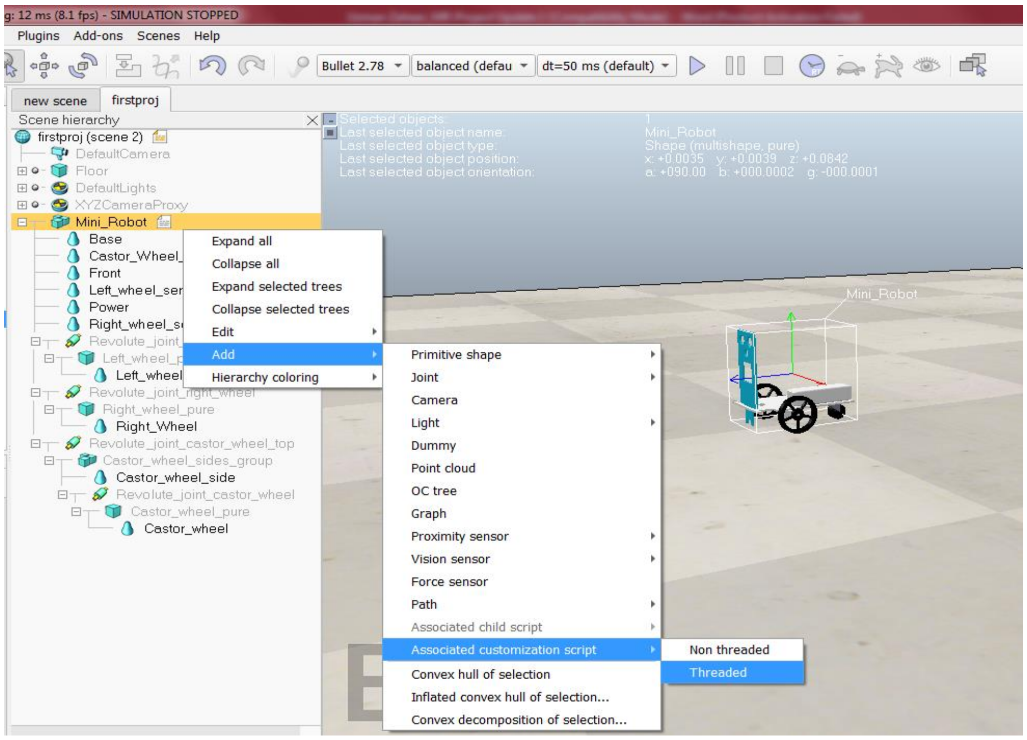


Figure 15: Adding LUA child script to robot.

The values of parameters used in the simulation are given below:

- wheel_radius= 0.03,
- max_speed=0.2,
- max_turn=0.2,
- actual speed=0 (initially zero),
- actual turn=0 (initially zero rad/s),
- b=0.0565 (separation/distance of wheels from centre of robot)

The kinematic equation for the wheels in mobile robotics is given below:

wheel velocity = 1/2*(max_speed/wheel_radius)

wheel target velocity = (v-b*w/wheel_radius)

here,

v= linear velocity

b= distance of wheels from centre of robot (wheel basis)

w= angular velocity (Omega)

Some important functions written for the movement of mini_robot are given below:

To move robot using wheel kinematics equation:

```
function move(v,w)
   sim.setJointTargetVelocity(left_wheel,(v-b*w)/wheel_radius)
   sim.setJointTargetVelocity(right_wheel,(v+b*w)/wheel_radius)
end
```

To move robot in forward direction:

```
function moveForward()
   sim.setJointTargetVelocity(left_wheel,0.5*max_speed/wheel_radius)
   sim.setJointTargetVelocity(right_wheel,0.5*max_speed/wheel_radius)
end
```

To turn robot in left direction:

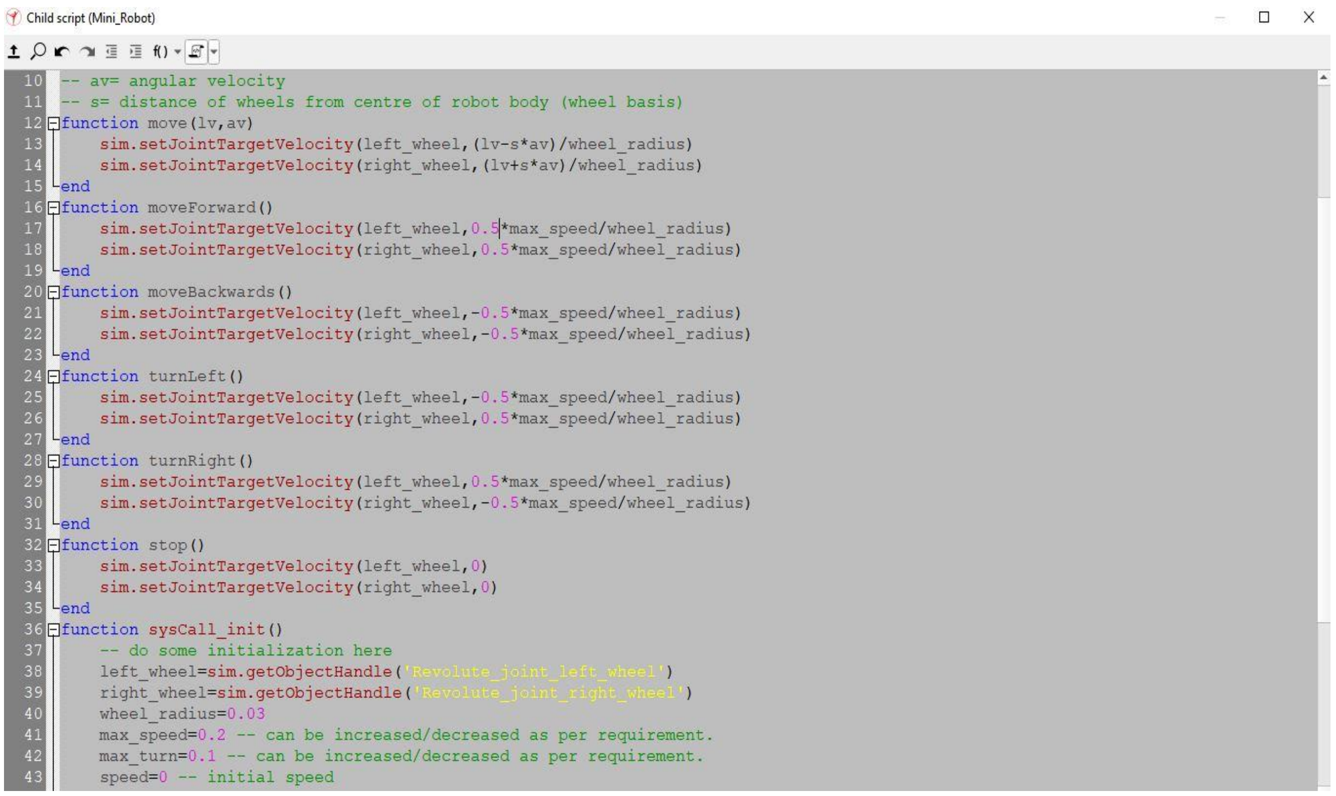
```
function turnLeft()
   sim.setJointTargetVelocity(left_wheel,-0.5*max_speed/wheel_radius)
   sim.setJointTargetVelocity(right_wheel,0.5*max_speed/wheel_radius)
end
```

To stop the robot:

function stop()

sim.setJointTargetVelocity(left_wheel,0)

sim.setJointTargetVelocity(right_wheel,0)



```
-- av= angular velocity
-- s= distance of wheels from centre of robot body (wheel basis)
function move(lv,av)
    sim.setJointTargetVelocity(left_wheel,(lv-s*av)/wheel_radius)
    sim.setJointTargetVelocity(right_wheel,(lv+s*av)/wheel_radius)
end
function moveForward()
    sim.setJointTargetVelocity(left_wheel,0.5*max_speed/wheel_radius)
    sim.setJointTargetVelocity(right_wheel,0.5*max_speed/wheel_radius)
end
function moveBackwards()
    sim.setJointTargetVelocity(left_wheel,-0.5*max_speed/wheel_radius)
    sim.setJointTargetVelocity(right_wheel,-0.5*max_speed/wheel_radius)
end
function turnLeft()
    sim.setJointTargetVelocity(left_wheel,-0.5*max_speed/wheel_radius)
    sim.setJointTargetVelocity(right_wheel,0.5*max_speed/wheel_radius)
end
function turnRight()
    sim.setJointTargetVelocity(left_wheel,0.5*max_speed/wheel_radius)
    sim.setJointTargetVelocity(right_wheel,-0.5*max_speed/wheel_radius)
end
function stop()
    sim.setJointTargetVelocity(left_wheel,0)
    sim.setJointTargetVelocity(right_wheel,0)
end
function sysCall_init()
    -- do some initialization here
    left_wheel=sim.getObjectHandle('Revolute_joint_left_wheel')
    right_wheel=sim.getObjectHandle('Revolute_joint_right_wheel')
    wheel_radius=0.03
    max_speed=0.2 -- can be increased/decreased as per requirement.
    max_turn=0.1 -- can be increased/decreased as per requirement.
    speed=0 -- initial speed
```

Figure 16: LUA script in CoppeliaSim

The Simui.create command in LUA script is used for creating the user interface for the mini robot so that it can be controlled easily. It takes a string and creates a buttons and slider as a user interface for controlling the robot.

➢ Running the simulation:

When the simulation is run using the written LUA script for the Mini_Robot, a user interface menu will pop up through which robot can be controlled. It can me moved in following directions:

- It can be moved in Forward & Backward direction
- It can be Rotated Right & Left.
- It can be stopped.

The slider control can also be used for controlling the movement of the robot.

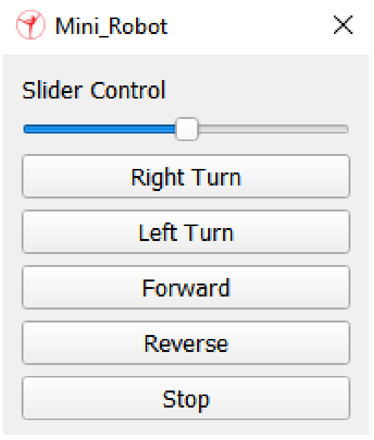
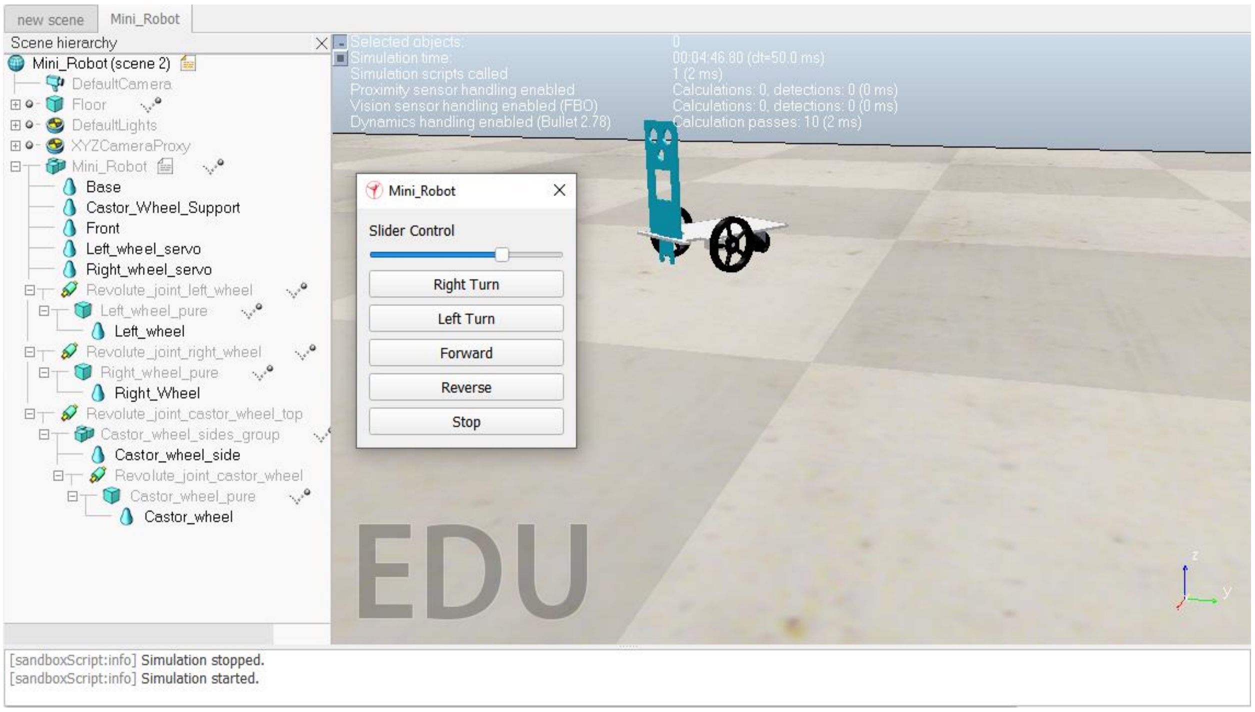


Figure 17: User interface buttons for controlling robot.



```
Selected objects:                        0
Simulation time:                         00:04:46.80 (dt=50.0 ms)
Simulation scripts called                1 (2 ms)
Proximity sensor handling enabled        Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)     Calculations: 0, detections: 0 (0 ms)
Dynamics handling enabled (Bullet 2.78)  Calculation passes: 10 (2 ms)
```

```
[sandboxScript:info] Simulation stopped.
[sandboxScript:info] Simulation started.
```

Figure 18: Simulation running with user interface buttons for controlling robot.

7. Conclusion:

The 3D modeling and simulation of a wheeled mobile is successfully completed in this project. The 3D modeling of different parts and their assembly was done in Autodesk Fusion 360, which is a very friendly and a cloud-based platform used for the designing and simulation of components. The simulation of the robot is done in CoppeliaSim also known as Vrep. The control functions and kinematic wheel equations of mobile robot are written using the LUA script of the software and the user interface for control buttons is created using the Sim.createUI API of the software.

8. Instructions on running the simulation:

The scene of the CoppeliaSim environment with all the relevant parameters and settings is provided along with LUA script code is provided. One just has to import the provided scene (CoppeliaSim file named “Mini_Robot.ttt”) in the CoppeliaSim software and run the simulation. A user interface as described above will pop up and any one can move the robot using it.

9. References:

1. Jamil, O., Jamil, M., Ayaz, Y., & Ahmad, K. (2014). Modeling, control of a two-wheeled self-balancing robot. 2014 International Conference On Robotics And Emerging Allied Technologies In Engineering (Icreate). doi: 10.1109/icreate.2014.6828364
2. Pandey, A., Jha, S., & Chakravarty, D. (2017). Modeling and Control of an Autonomous Three Wheeled Mobile Robot with Front Steer. 2017 First IEEE International Conference On Robotic Computing (IRC). doi: 10.1109/IRC.2017.67
3. Fusion 360 | 3D CAD, CAM, CAE & PCB Cloud-Based Software | Autodesk. (2021). Retrieved 20 June 2021, from https://www.autodesk.com/products/fusion-360/overview?term=1-YEAR
4. Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics. (2021). Retrieved 20 June 2021, from https://www.coppeliarobotics.com/
5. Solar2D Documentation — Developer Guides | Start. (2021). Retrieved 20 June 2021, from https://docs.coronalabs.com/guide/start/introLua/index.html#:~:text=A%20comment%20starts%20with%20a,in%20%2D%2D%2D%5B%5B%20.
6. CustomUI Plugin - UI XML Syntax. (2021). Retrieved 20 June 2021, from https://coppeliarobotics.com/helpFiles/en/simUI-widgets.html