| Topic | Planet Exploration Robot - 3 | |
|---|---|---|
| Class Description | **Students will be introduced to distance sensors. Students will learn to toggle between auto mode and manual mode. The rover detects obstacles in the path and changes its direction.** | |
| Class | **PRO C286** | |
| Class time | **45 mins** | |
| Goal | • Move the rover automatically<br>• Add distance sensors to the rover.<br>• Turn the rover to the right if an obstacle is detected | |
| Resources Required | • weTeacher Resources:<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>○ Smartphone<br><br>• Student Resources:<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen | |
| Class structure | **Warm-Up**<br>**Teacher-Led Activity 1**<br>**Student-Led Activity 1**<br>**Wrap-Up** | **10 mins**<br>**10 mins**<br>**20 mins**<br>**05 mins** |
| Credit & Permissions: | **This project uses [Webots](), an open-source mobile robot simulation software developed by Cyberbotics Ltd.**<br>[License]() | |
| **WARM-UP SESSION - 10 mins** | | |

<table>
<tr><th>Teacher Action</th><th>Student Action</th></tr>
<tr><td>Hey <student's name>. How was your day?<br><br>Are you excited to learn something new today?<br><br>Following are the WARM-UP session deliverables:<ul><li>Greet the student.</li><li>Revision of previous class activities.</li><li>Quizzes.</li></ul></td><td>ESR: Hi, varied<br><br>ESR: Yes.<br><br>Click on the slide show tab and present the slides</td></tr>
<tr><td colspan="2">WARM-UP QUIZ<br>Click on In-Class Quiz</td></tr>
<tr><td colspan="2">Activity Details<br><br>Following are the session deliverables:<ul><li>Appreciate the student.</li><li>Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</li></ul></td></tr>
<tr><td colspan="2">TEACHER-LED ACTIVITY - 10 mins</td></tr>
<tr><td colspan="2">Teacher Initiates Screen Share</td></tr>
<tr><td colspan="2"><ul><li>Add automation to the planet rover</li></ul></td></tr>
<tr><th>Teacher Action</th><th>Student Action</th></tr>
<tr><td>Do you remember what we did in the last class?<br><br><br><br><br>Yes, we added keyboard control to the Planet Rover. Wouldn’t it be exciting if the rover could detect obstacles</td><td>ESR: We added a controller to the Robot and made it move when we press WSAD keys on the keyboard.</td></tr>
</table>

on its own and avoid them by moving in a different direction?

First we will add automation to the  Planet Exploration Robot so that it moves around on its own. Then we will add two distance sensors at the front of the rover so that it can detect and avoid obstacles while moving. When an obstacle is detected the rover will change its direction of movement.

*Teacher opens [Teacher Activity 1](#) and starts coding.*

First we will set a flag or an indicator that determines whether the Robot is currently in auto mode or not.
We will declare a variable **autoMode** and set it to **False**.

```
robot=Robot()
keyboard = Keyboard()
timestep=64

autoMode= False
```

We will switch to auto mode when the key “**o**” is pressed. The auto mode will switch off and manual mode will start when “**o**” is pressed again.

The keyCode for “**o**” is 79. So let us write an if condition that checks whether the key pressed is “**o**”. If “**o**” is pressed, invert the value of **autoMode**. That is if the value is **True** make it **False** and if the value is **False** make it **True**. This can be achieved using the **not** keyword.

So the mode will keep switching from auto mode and back whenever “**o**” is pressed.

**CODE:**

```
while (robot.step(timestep) !=-1):

    key_pressed = keyboard.getKey()

    # 79 is "o" key
    if(key_pressed==79):
        autoMode= not autoMode
        print(autoMode)
```

**OUTPUT:**

```
Console - All
True
False
True
False
True
False
True
False
True
False
True
False
True
False
True
False
```

But in the print statements we can note that there is quick switching between **True** and **False**. The value of timestep is 64ms, but we cannot press and release a key in such a short span of time. So the program reads the value of **key_pressed** multiple times. That is, since the value of timestep is 64ms, the value of **key_pressed** keeps toggling every 64 ms. But we would want the value to toggle just once when we press it one time.

To achieve this we will add one more condition to our if statement. We will also check the previous key that was

| | |
|---|---|
| pressed. First we will store the previous key pressed in a variable **prev_key**. Now we will rewrite the if condition that if no key was pressed previously and the current key pressed is “**o**”, then switch the mode.<br><br>Now note that after pressing “**o**”, the **autoMode** stays **True** or **False** for a considerable amount of time | |

**CODE:**

```
prev_key = 0
key_pressed = -1
while (robot.step(timestep) !=-1):
    prev_key = key_pressed
    key_pressed = keyboard.getKey()
    # 79 is "o" key
    if(prev_key == -1 and key_pressed==79):
        autoMode= not autoMode
        print(autoMode)
```

**OUTPUT:**

```
Console - All
True
True
True
True
True
True
True
False
False
False
False
False
False
False
```

| | |
|---|---|
| Now that we can switch between auto mode and manual mode, can you tell me what we should do when the Rover is in autoMode?<br><br>That is correct. We should move the rover using the | **ESR:** We should start moving the rover. |

| | |
|---|---|
| **setVelocity()** function. We will move the rover forward by passing **speed** as the parameter.<br><br>The code for keyboard control that we wrote in the last class should be placed inside an if condition that checks whether the mode is manual mode. | |

**CODE:** NOTE : Intend the previous class code of keyboard control to the if block of **if (not autoMode):**

```
if(autoMode):
    wheel1_left.setVelocity(speed)
    wheel1_right.setVelocity(speed)
    wheel2_left.setVelocity(speed)
    wheel2_right.setVelocity(speed)

if(not autoMode):
    # 65 is "a" key
    if(key_pressed== 65):
        wheel1_left.setVelocity(-speed)
        wheel1_right.setVelocity(speed)
        wheel2_left.setVelocity(-speed)
        wheel2_right.setVelocity(speed)

    # 68 is "d" key
    if(key_pressed== 68):
        wheel1_left.setVelocity(speed)
        wheel1_right.setVelocity(-speed)
        wheel2_left.setVelocity(speed)
        wheel2_right.setVelocity(-speed)

    # 87 is "w" key
    if(key_pressed== 87):
        wheel1_left.setVelocity(speed)
        wheel1_right.setVelocity(speed)
        wheel2_left.setVelocity(speed)
        wheel2_right.setVelocity(speed)

    # 83 is "s" key
    if(key_pressed== 83):
        wheel1_left.setVelocity(-speed)
        wheel1_right.setVelocity(-speed)
        wheel2_left.setVelocity(-speed)
        wheel2_right.setVelocity(-speed)

    # if nothing is pressed
    if(key_pressed== -1):
        wheel1_left.setVelocity(0)
        wheel1_right.setVelocity(0)
        wheel2_left.setVelocity(0)
        wheel2_right.setVelocity(0)
```

**OUTPUT:**

Isn’t this wonderful, now the rover moves automatically when the “**o**” key is pressed.

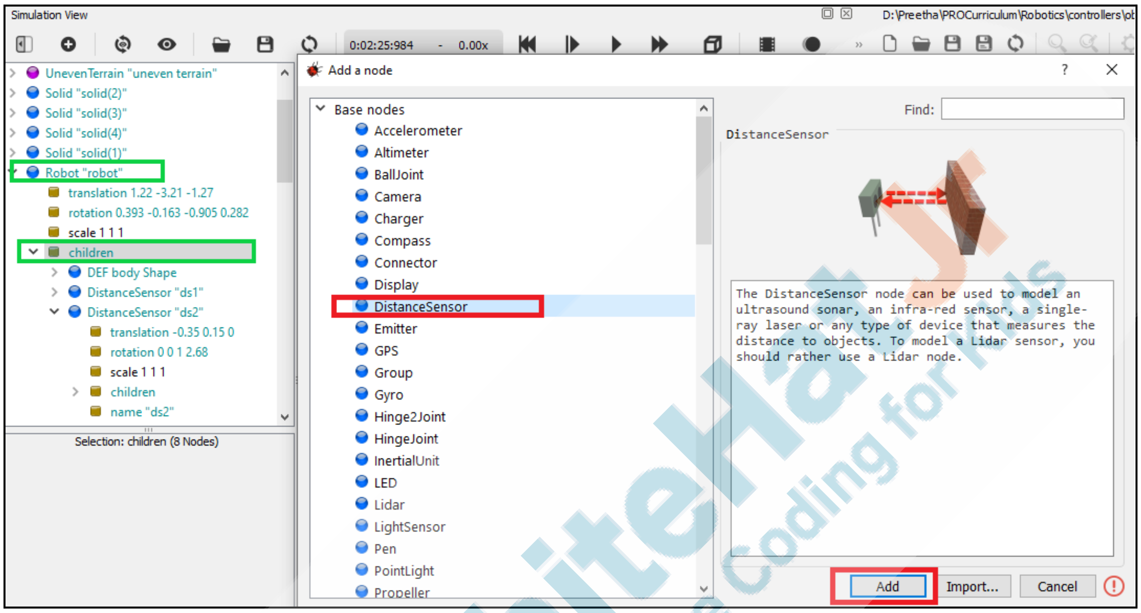
Now let us get on to the next task for the day, obstacle detection. There are some obstacles placed in the way of the rover.

Obstacle detection is an integral part of any distant planet exploration since it is much easier and much more efficient than manual control.

A distance sensor node can be added as a child node to a Robot. There are four types of distance sensors : a generic sensor, an infra-red sensor, a sonar sensor, or a

| | |
|---|---|
| laser range-finder. A distance sensor can detect collisions between other objects around it. In case of generic, sonar and laser collision occurs between bounding objects. That is the collision occurs between the bounding object of the Solid node of the robot and the bounding object of the solid node of the obstacle.<br><br>We will add two distance sensors at the front of the rover. Note that a distance sensor can be added anywhere (front, back, sides) on the robot. | |

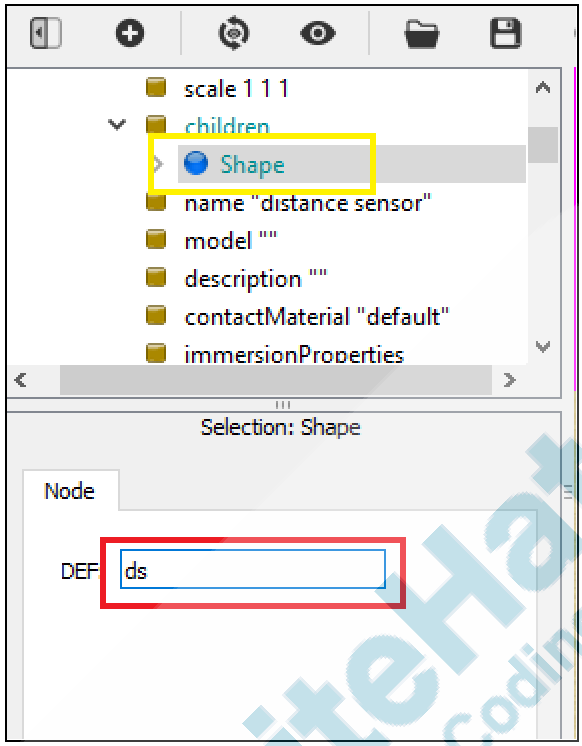
Let's add the distance sensor in our robot.
Under the children of the Robot node, add a new distance sensor.
**ROBOT > children > Add New > Base Nodes > DistanceSensor**




We have added the distance sensor. But it is invisible right now. We need to add shape and physics to it.

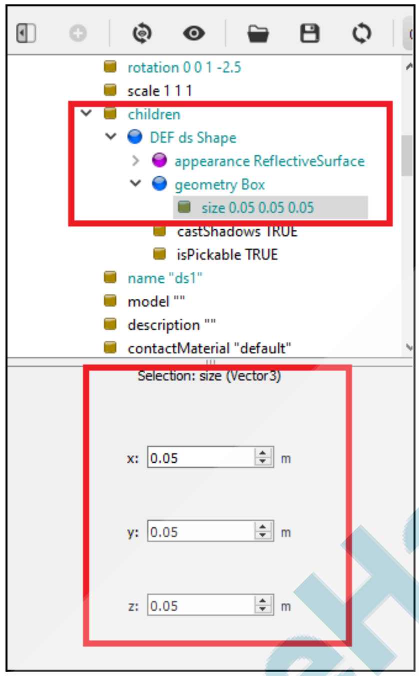
So, we will add a **Shape** "Box" under children of Distance Sensor.

1. Go to **children** node of the **distance sensor** you have just added → Right click on it → click on **Add New** → select **shapes** under **base nodes** → click on **Add**
2. Click on the newly added **shape** node and change its **DEF:** property to **ds**. This will later help us add the bounding object.




Set the following properties for the **Shape**.

1. Set **appearance** as **ReflectiveSurface.**
    - Double-click on appearance → click on **PROTO nodes (webots projects)** → **appearances** → select **ReflectiveSurface** → click on **Add**
2. Set **geometry** as **Box**.
    - Double-click on geometry→ click on **base nodes** → **Box**
    - Also change the **size** property to **x : 0.5 , y : 0.5 , z : 0.5**

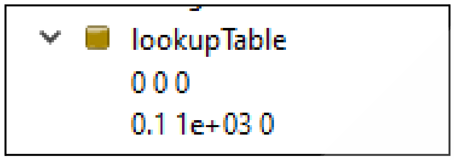



We have a shape for the bounding object now. So, let’s add physics as well. Set the following properties:

1. **name : ds1**

2. Set **boundingObject** as **ds**
3. Add **physics** node and set the **mass** as **1**
4. Now, we need to understand the **lookUp Table** property as well.

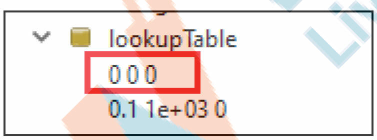
   Notice that the **lookupTable** has two rows by default-




As you can see the lookup table has three values **x, y** and **z**.

1. **x** stands for the **distance from the obstacle**.
2. **y** stands for the **value returned by the sensor**.
3. **z** stands for the **noise** (the deterioration of the signal due to other components).

Here, the first row represents that for **0 meters** (represented by x) distance from the obstacle, the distance sensor will return a value **0** (represented by y). We will ignore the noise for now.



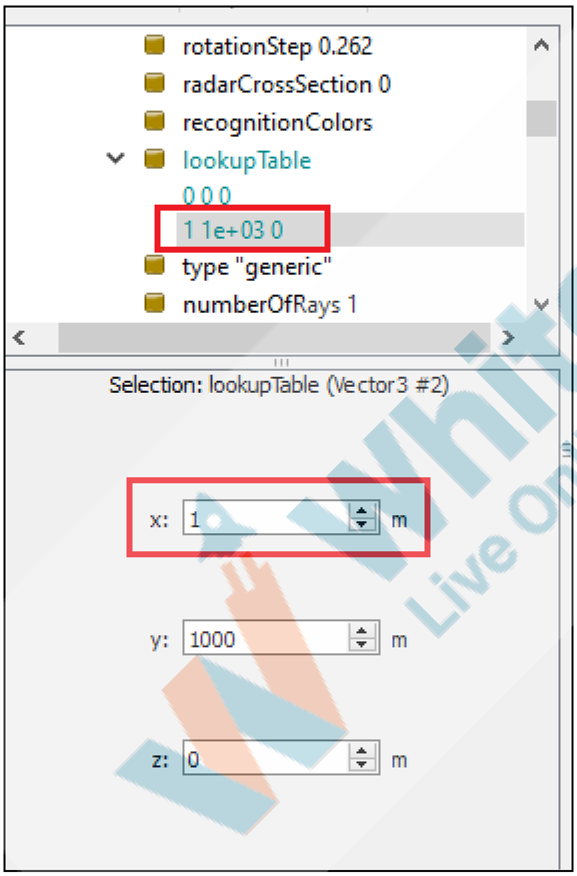

The second row of the **lookUp Table** represents that for **0.1 meters** (represented by x) distance from the obstacle, the distance sensor will return a value **1000** (represented by y).

So, for **0-0.1** meters of distance the sensor will return a value between **0-1000.**

We can add more rows to the **lookUp Table** and configure the distance sensor to return certain values depending on the distance.

For now, we will just change the second row. Instead of detecting objects from **0.1 meter** away, we want it to detect the object from a meter away.

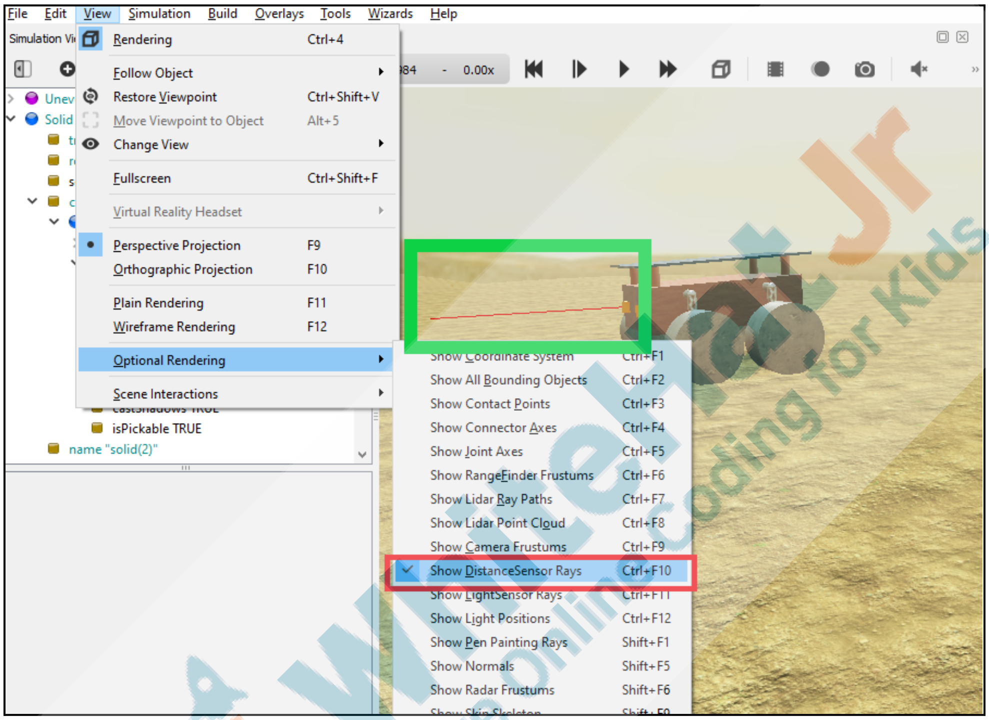
So, let's change the second row's first value to 1.

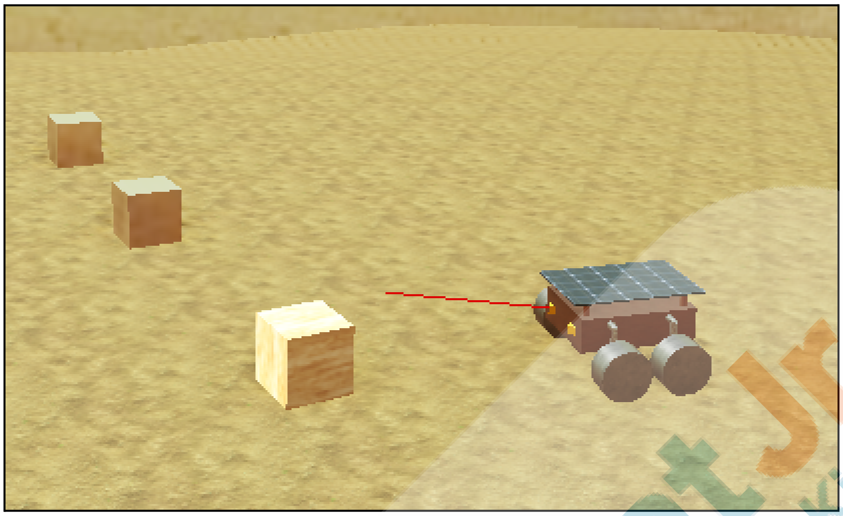



The area covered by the distance sensor can be viewed by checking the option **View > Optional Rendering > Show DistanceSensor Rays**

Increasing the **y** value increases the length of the ray and the distance covered by the sensor.

**View > Optional Rendering > Show DistanceSensor Rays**

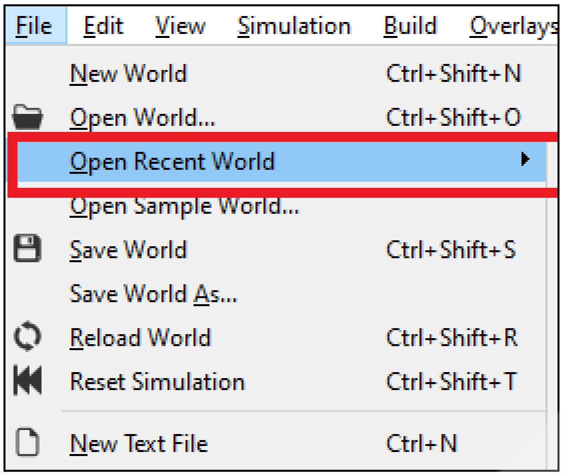


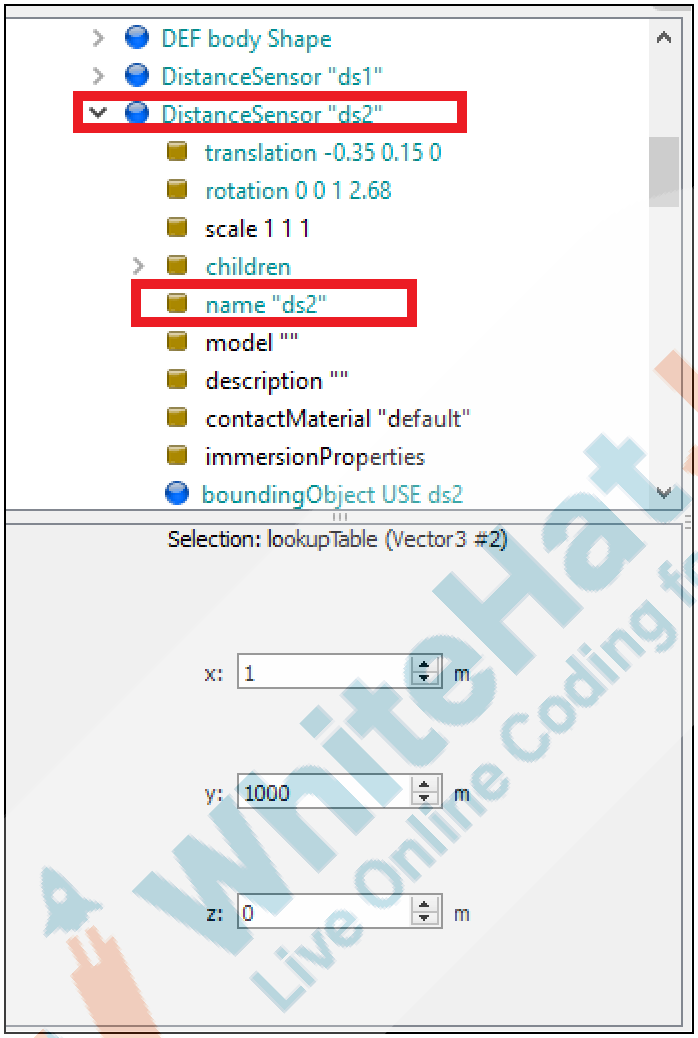


The color of the ray changes to green beyond collision point

| **Teacher Stops Screen Share** | |
|---|---|
| So now it’s your turn to code..<br>Please share your screen with me. | |
| Can you add one more distance sensor to the rover and write code in the controller so that the rover would turn in a | |

| | |
|---|---|
| different direction and when an obstacle is detected?<br><br>That's great. I will guide you through it. | **ESR:** Yes! |
| **STUDENT-LED ACTIVITY - 20 mins** | |
| • **Ask the student to press the ESC key to come back to the panel.**<br>• **Guide the student to start Screen Share.**<br>• **The teacher gets into Full Screen.** | |
| **Student Initiates Screen Share** | |
| **ACTIVITY**<br><br>• **Add a distance sensor to the rover.**<br>• **Avoid obstacles that come in the way by moving in a different direction.** | |
| **Teacher Action** | **Student Action** |
| *The teacher helps the student to download boilerplate.*<br><br>A few solids have been added as obstacles.<br><br>*Guide the student to create a controller. The student has to delete the contents of the newly created controller and paste the code given in the downloaded automode_controller.py file. Guide the student to reopen the World by clicking the menu* ***File > Open Recent World.*** | *Student downloads* **Student Activity 1** *in Webots* |

| | |
|---|---|
|  <br>*Guide the students to add a second distance sensor to the rover.* | |
| Add a second distance sensor to the rover.<br>Set the following properties<br>**Name : ds2**<br>**boundingObject > USE > ds2**<br>**physics > Physics**<br>**mass : 1**<br>**lookupTable - x: 1, y: 1000, z: 0** | |


DEF body Shape
DistanceSensor "ds1"
DistanceSensor "ds2"
translation -0.35 0.15 0
rotation 0 0 1 2.68
scale 1 1 1
children
name "ds2"
model ""
description ""
contactMaterial "default"
immersionProperties
boundingObject USE ds2
Selection: lookupTable (Vector3 #2)
x: 1 m
y: 1000 m
z: 0 m

| | |
|---|---|
| Now let us write code in the controller so that the robot would move in a different direction when it detects an obstacle.<br>Whenever we add a new component to the controller, what is the first line of code that we have to write?<br><br>That is right, let us import the **DistanceSensor** from the controller. | **ESR:** import the library to use the component. |

```
from controller import Robot
from controller import Keyboard

from controller import DistanceSensor
```

| | |
|---|---|
| Remember that in the keyboard control we had to enable the keyboard inputs. Just like that the distance sensor also needs to be enabled.<br>Each sensor will have a unique identifier which is returned by the function **getDevice()**. Once the device is obtained we can enable it using the **enable()** function. | |
| <pre>ds1= robot.getDevice("ds1")<br>ds2= robot.getDevice("ds2")<br><br>ds1.enable(timestep)<br>ds2.enable(timestep)</pre> | |
| Let us declare a variable called **number_of_turns** that holds the number of obstacles that the sensor has detected. Set the initial value of **number_of_turns to** zero. | |
| <pre>ds1= robot.getDevice("ds1")<br>ds2= robot.getDevice("ds2")<br><br>ds1.enable(timestep)<br>ds2.enable(timestep)<br><br>number_of_turns=0</pre> | |
| The **getValue()** function returns the last value measured by the specified distance sensor. By checking the value in an if condition we will know whether there are any obstacles to be avoided in the proximity of the rover. The value returned will depend upon the y value we had set in the lookupTable.<br><br>Do you remember what the y value was that we set?<br><br>So whenever the value becomes less than 1000, it means that there is an obstacle nearby. If obstacles are detected, we'll set the value of **number_of_turns** to 8.  This value | **ESR: 1000** |

| | |
|---|---|
| can be adjusted according to how much we want to turn the vehicle.<br><br>**if(ds1_value<1000 or ds2_value<1000):**<br>**number_of_turns = 8**<br><br>Finally, if there are obstacles in the way of the rover, what should we do?<br><br>Exactly, so let us write the code to move the rover to the right.<br><br>We will use the **setVelocity()** method for this. As **timeStep** is **64 milliseconds,** the **setVelocity()** runs for **64 milliseconds** for each loop.<br><br>We would want it to turn **90 degrees** approximately. For that we will run the **setVelocity()** method until the **number_of_turns** variable is greater than **0** and we will also reduce the count of **number_of_turns** by 1.<br><br>If there are obstacles, set positive velocity to the right wheels and negative velocity to the left wheels so that the rover turns away from the obstacle.<br><br>**if(number_of_turns >0):**<br>**number_of_turns =number_of_turns -1**<br>**wheel1_left.setVelocity(-speed)**<br>**wheel1_right.setVelocity(speed)**<br>**wheel2_left.setVelocity(-speed)**<br>**wheel2_right.setVelocity(speed)**<br><br>Else if there are no obstacles, then the rover can move straight ahead.<br><br>**else:** | **ESR:** We should move the rover left or right |

```
wheel1_left.setVelocity(speed)
wheel1_right.setVelocity(speed)
wheel2_left.setVelocity(speed)
wheel2_right.setVelocity(speed)
```

**CODE:**

```
if(autoMode):
    ds1_value = ds1.getValue()
    ds2_value = ds2.getValue()

    if(ds1_value<1000 or ds2_value<1000):
        number_of_turns=8

    if(number_of_turns>0):
        number_of_turns=number_of_turns-1
        wheel1_left.setVelocity(-speed)
        wheel1_right.setVelocity(speed)
        wheel2_left.setVelocity(-speed)
        wheel2_right.setVelocity(speed)
    else:
        wheel1_left.setVelocity(speed)
        wheel1_right.setVelocity(speed)
        wheel2_left.setVelocity(speed)
        wheel2_right.setVelocity(speed)
```

**OUTPUT:**

Click here to view the output video

| | |
|---|---|
| Wonderful, Now the planet explorer moves in auto mode by avoiding obstacles that come in the way. | |

**Teacher Guides Student to Stop Screen Share**

**WRAP-UP SESSION - 05 mins**

**Activity details**

**Following are the WRAP-UP session deliverables:**

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

**WRAP-UP QUIZ**
Click on In-Class Quiz

**Activity Details**

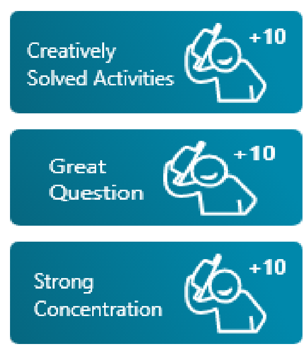
**Following are the session deliverables:**

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

**FEEDBACK**

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

| Teacher Action | Student Action |
|---|---|
| You get “hats-off” for your excellent work!<br><br>In the next class, add a camera to the rover. | *Make sure you have given at least 2 hats-off during the class for:* |




**PROJECT OVERVIEW DISCUSSION**
Refer the document below in Activity Links Sections

**Teacher Clicks**

| ACTIVITY LINKS | | |
|---|---|---|
| **Activity Name** | **Description** | **Links** |
| Teacher Activity 1 | Previous class code | https://github.com/procodingclass/PRO-C285-Reference-Code |
| Teacher Reference 1 | Reference Code | https://github.com/procodingclass/PRO-C286-Reference-Code |
| Teacher Reference 2 | Project | https://s3-whjr-curriculum-uploads.whjr.online/85d23d71-bea6-4b4a-88a5-2ae6500bee07.pdf |
| Teacher Reference 3 | Project Solution | https://github.com/procodingclass/PRO-C286-Project-Solution |
| Teacher Reference 4 | In-Class Quiz | https://s3-whjr-curriculum-uploads.whjr.online/e847ba53-d920-4a39-86fe-069439da7228.pdf |
| Student Activity 1 | Boilerplate Code | https://github.com/procodingclass/PRO-C286-Student-Boilerplate |