Topic	SELF DRIVING CAR - FINAL	
Class Description	Students will learn how to attach a Lidar to the car and create an algorithm to avoid obstacles.	
Class	PRO C300	
Class time	50 mins	
Goal	● Attaching a Lidar sensor to the car. ● Writing algorithm to avoid obstacles.	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	Warm-Up Teacher -Led-Activity 1 Student-Led Activity 1 Wrap-Up	5 mins 20 mins 20 mins 5 mins
Credit & Permissions:	This project uses Webots, an open-source mobile robot simulation software developed by Cyberbotics Ltd. License	

WARM-UP SESSION - 10 mins	
Teacher Action	Student Action

Teacher	Student
Hey <student's name>. How are you? It’s great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities. ● Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides

WARM-UP QUIZ

Click on In-Class Quiz

Activity Details

Following are the session deliverables:

- Appreciate the student.
- Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.

TEACHER-LED ACTIVITY - 15 mins

Teacher Initiates Screen Share

ACTIVITY

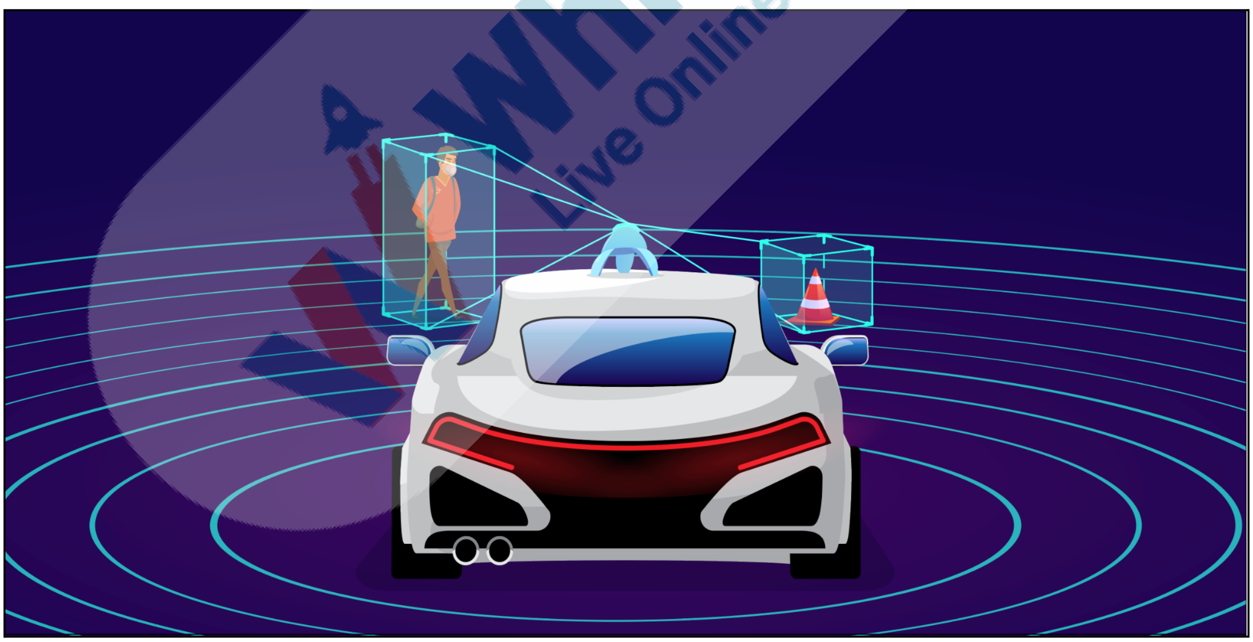
- **Adding a Lidar to the BMW X5 car node.**

Teacher Action	Student Action
Do you remember what we did in the last class? Great, if you have any doubts from the last class, please ask. *Note : Teacher will clear the doubts, if students have any.* Now that you don’t have any questions from the previous classes, let’s learn something new today.	**ESR :** Yes, we started creating a self-driving car.

In the previous class, we made a **self-driving car** where we attached a **camera node.** Using the camera the car could get information from its external surroundings. Using this information we wrote the code so that the car follows the yellow line on the road. What if there is an obstacle on this yellow line? Today, we will use a Lidar to detect any obstacle in front of the car. But before that let's understand what a Lidar is.	**ESR:** The robot should be able to avoid the obstacle.
We have heard of Radar which stands for **Radio Detection and Ranging**. Radar uses radio waves or sound energy to detect the distance, location of an object. Lidar uses light energy instead of sound energy to detect the distance of objects and map them. Lidar stands for **Light Detection and Ranging**.	
<u>How does a Lidar work?</u> A lidar sensor emits pulsed light waves into the environment. These pulses bounce off surrounding objects and return to the sensor.	
<u>Normal view:</u>	

Click here to view the reference image.

Lidar View:

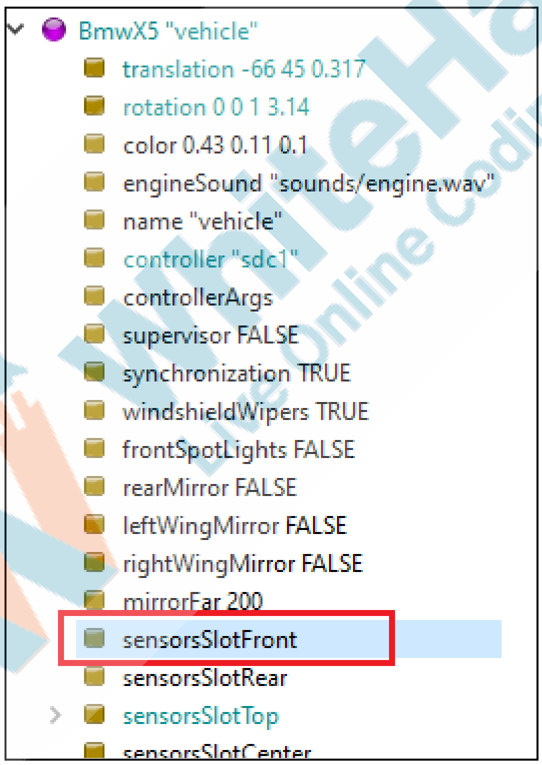


Click here to view the reference image.

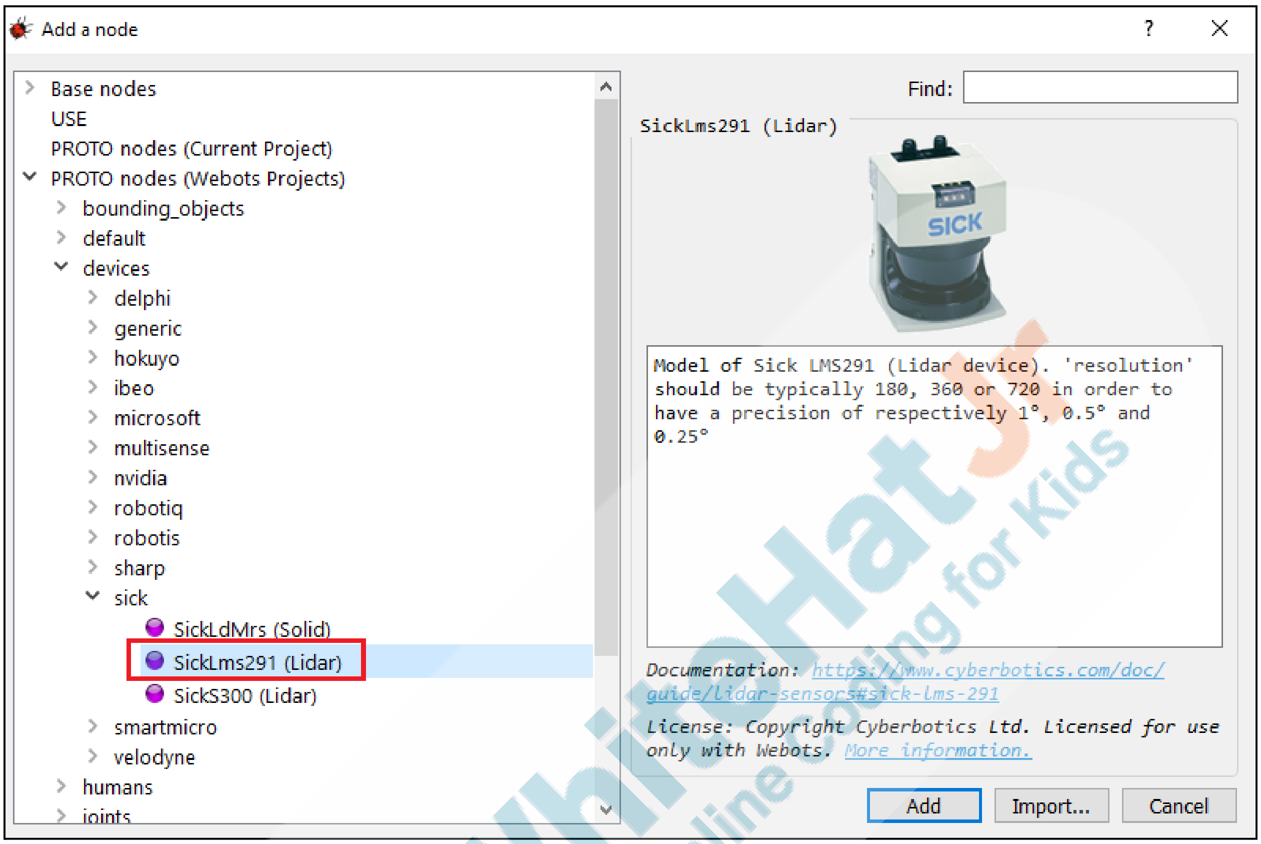
The sensor calculates the distance of the object by using the time it took to return to the source. This process can be repeated many times per second. As light travels faster than sound energy, Lidar is more efficient than a Radar. A Lidar can even create a real-time 3D map of the surrounding environment.

For that, let's first open the [Teacher Activity 1]() and download all the files from the previous class.

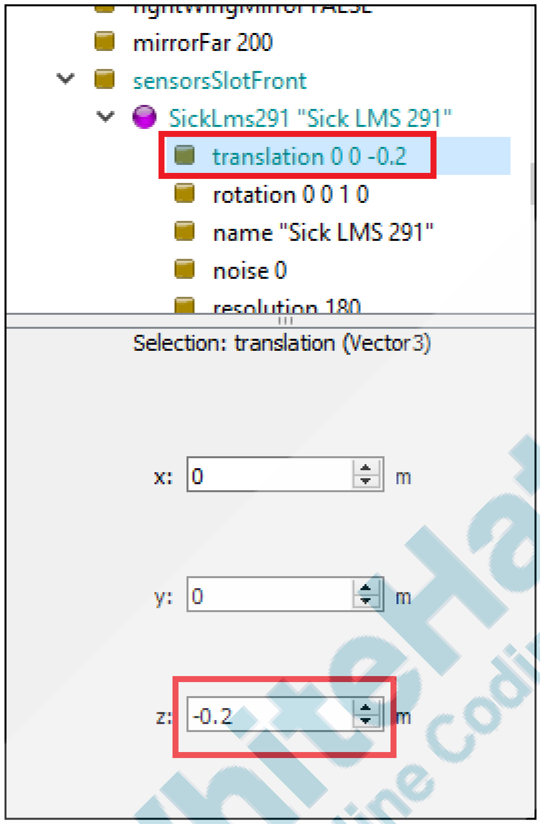
Once you have opened the downloaded files in the webots software, **expand** the **BMW X5 node** and **double click** on the **sensorSlotFront** property.



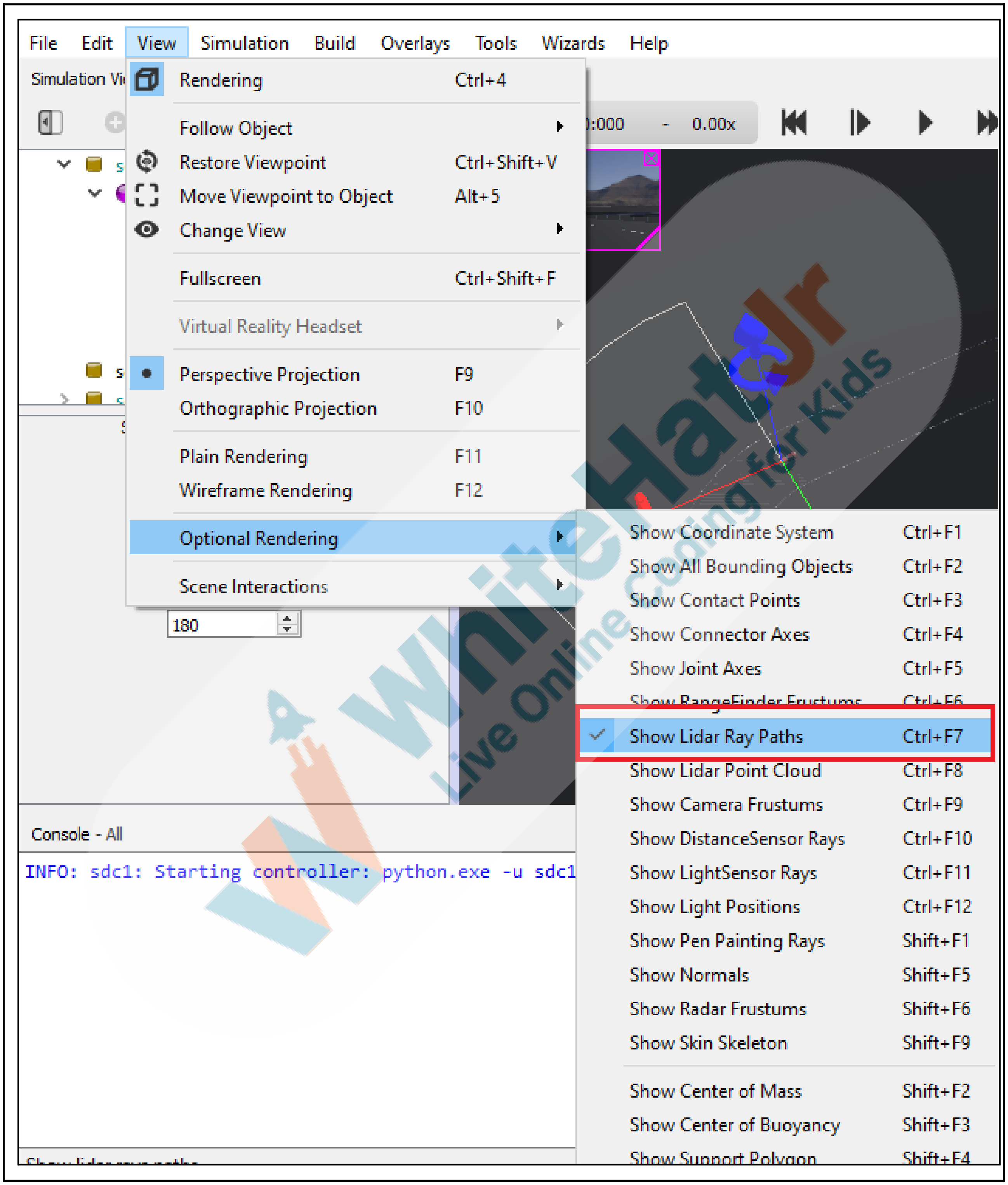
A window will open which will ask you to add a new node. **Expand** the **Proto nodes(Webots Projects)** → expand **devices** → expand **sick** → add the **SickLms291(Lidar)** node.



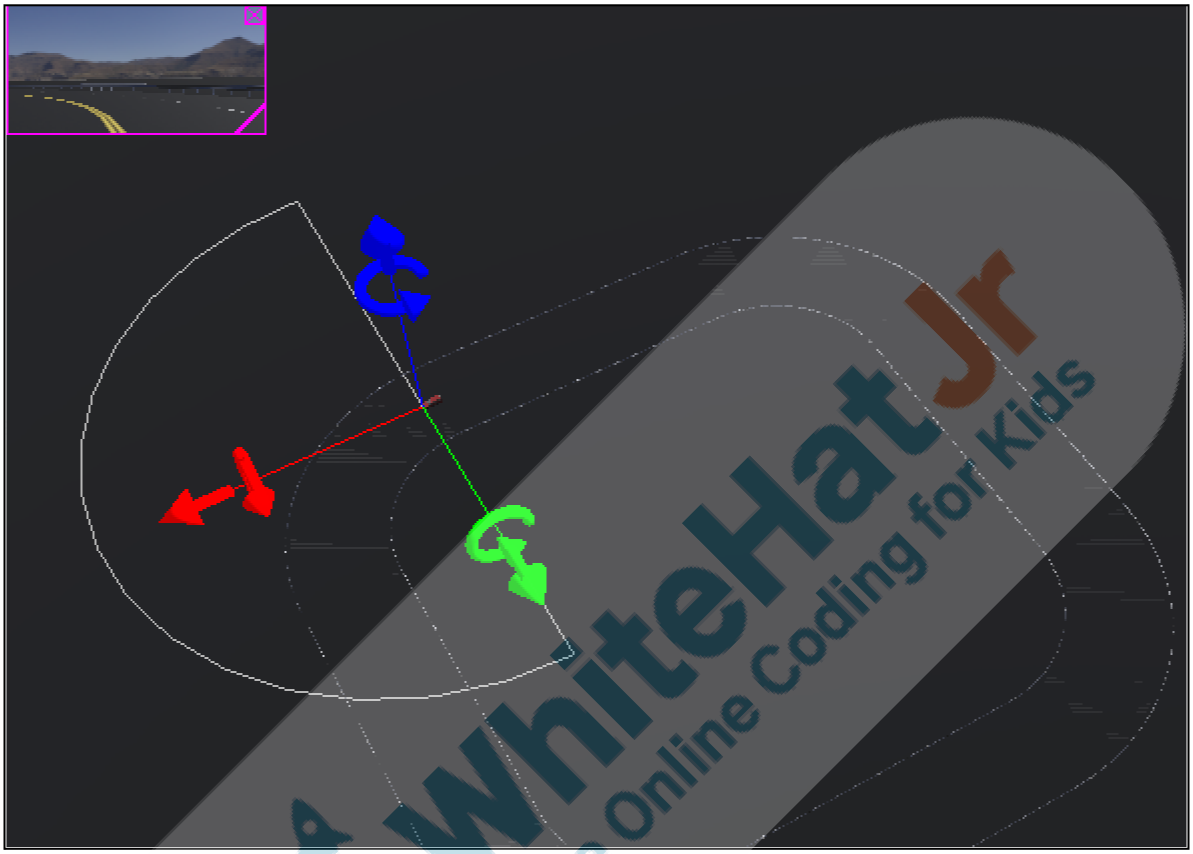
We will move the Lidar down a little. To do that we will go to **translation** and change the **z** value to **-0.2**.



Now, we need to see the range of the Lidar. Go to the **Menu Bar** → click on **View** → move your cursor to **Optional Rendering** → select **Show Lidar Ray Paths**	

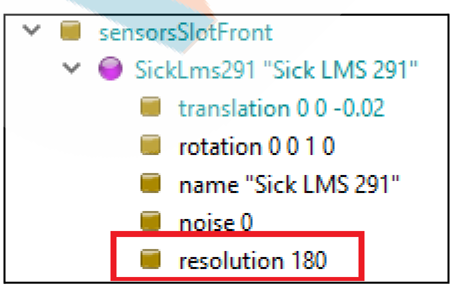

File
Edit
View
Simulation
Build
Overlays
Tools
Wizards
Help
Rendering Ctrl+4
Follow Object
Restore Viewpoint Ctrl+Shift+V
Move Viewpoint to Object Alt+5
Change View
Fullscreen Ctrl+Shift+F
Virtual Reality Headset
Perspective Projection F9
Orthographic Projection F10
Plain Rendering F11
Wireframe Rendering F12
Optional Rendering
Scene Interactions
Show Coordinate System Ctrl+F1
Show All Bounding Objects Ctrl+F2
Show Contact Points Ctrl+F3
Show Connector Axes Ctrl+F4
Show Joint Axes Ctrl+F5
Show RangeFinder Frustums Ctrl+F6
Show Lidar Ray Paths Ctrl+F7
Show Lidar Point Cloud Ctrl+F8
Show Camera Frustums Ctrl+F9
Show DistanceSensor Rays Ctrl+F10
Show LightSensor Rays Ctrl+F11
Show Light Positions Ctrl+F12
Show Pen Painting Rays Shift+F1
Show Normals Shift+F5
Show Radar Frustums Shift+F6
Show Skin Skeleton Shift+F9
Show Center of Mass Shift+F2
Show Center of Buoyancy Shift+F3
Show Support Polygon Shift+F4
Console - All
INFO: sdc1: Starting controller: python.exe -u sdc1
180
0.00x

Now, the output should look like this-



What you see here is the range of the Lidar. Basically, the Lidar should be able to detect anything within this range.

Let's look at the properties of the Lidar sensor. We should be able to see a property named **resolution**.



Here, we can see 180 written. It means there are 180 light rays in between this range that we can see. If we increase this value, the Lidar will be able detect more minute changes.

Remember that changing the resolution doesn't change the range. It only changes how many light rays are there within that range.

For now, we will keep it as 180.

Lidar is used widely in autonomous vehicles.

Let's start coding -

1. First, we need to create a unique identifier for the Lidar device. We will use the **getDevice()** method to do this.

```
from controller import Robot

bot = Robot()

timestep = 64

# getting devices
cam = bot.getDevice('camera')
left_wheel = bot.getDevice('left_front_wheel')
right_wheel = bot.getDevice('right_front_wheel')
l_steer = bot.getDevice('left_steer')
r_steer = bot.getDevice('right_steer')
lidar = bot.getDevice('Sick LMS 291')
```

2. After that we will enable the Lidar.

```
# initialisations
cam.enable(timestep)
left_wheel.setPosition(float('inf'))
right_wheel.setPosition(float('inf'))
l_steer.setPosition(0)
r_steer.setPosition(0)
left_wheel.setVelocity(0)
right_wheel.setVelocity(0)
lidar.enable(timestep)
```

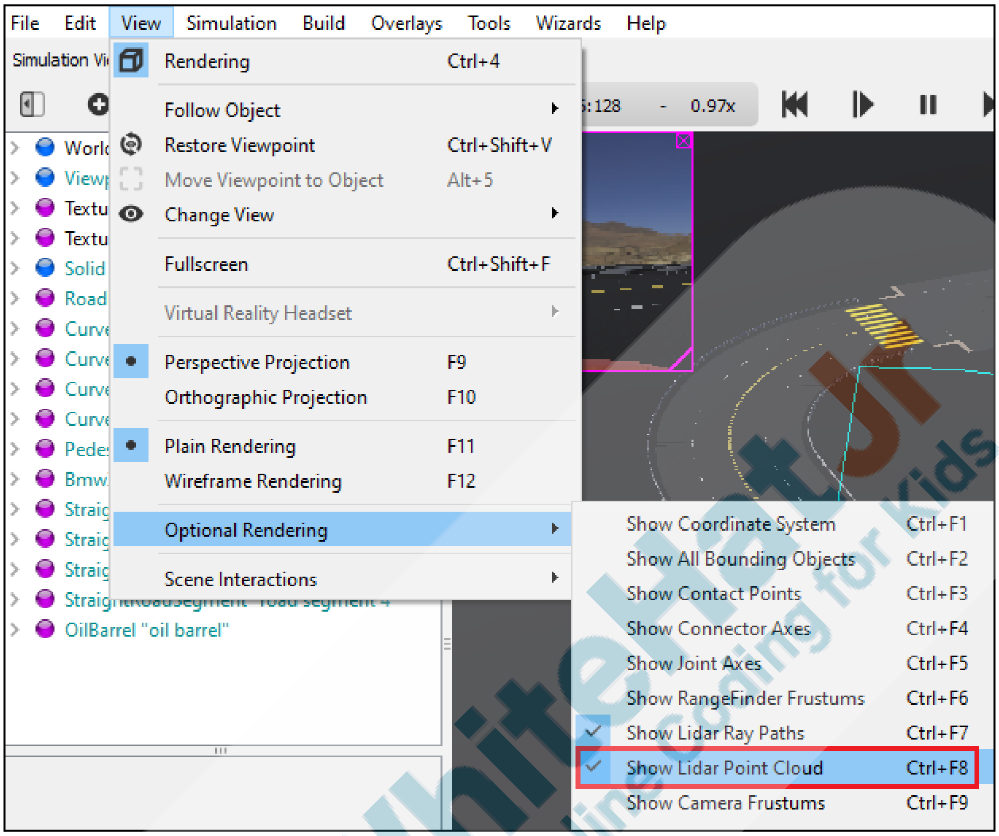
3. We can also see the point cloud for the Lidar. A point cloud is essentially a collection of tiny individual points plotted in the range of the Lidar. It's made up of a multitude of points. If you're scanning any object the light beam comes in contact with, each virtual point will represent a point on the object.

    a. To do that first we need to enable point cloud for the Lidar.

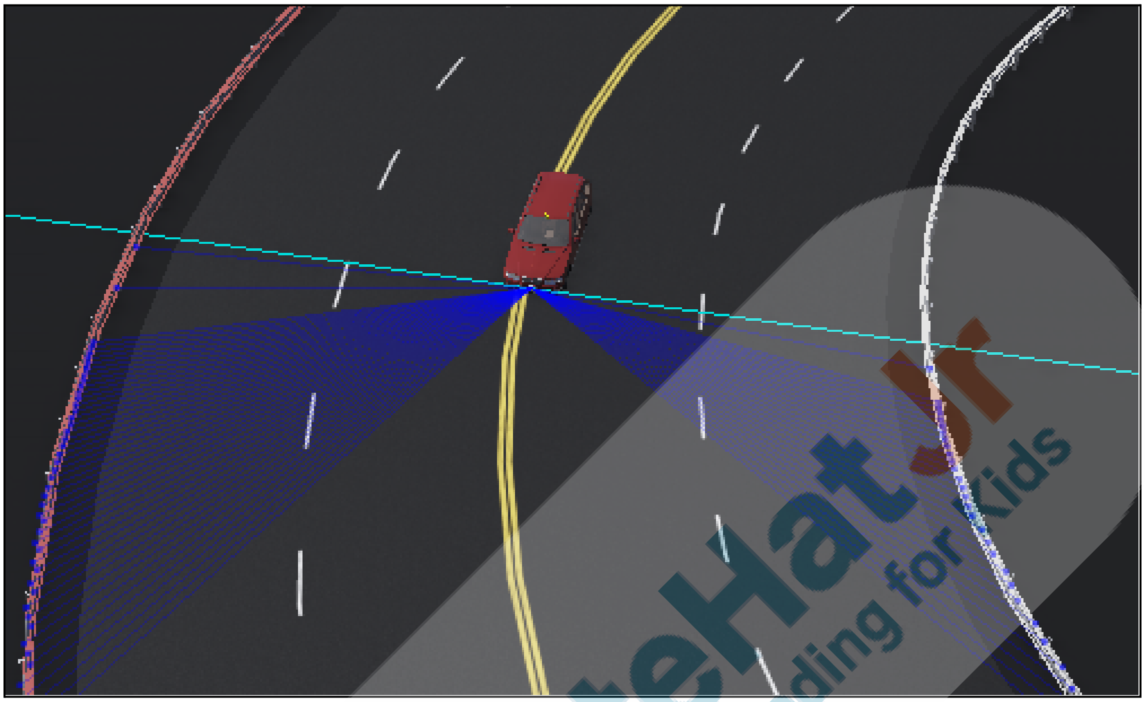
```
# initialisations
cam.enable(timestep)
left_wheel.setPosition(float('inf'))
right_wheel.setPosition(float('inf'))
l_steer.setPosition(0)
r_steer.setPosition(0)
left_wheel.setVelocity(0)
right_wheel.setVelocity(0)
lidar.enable(timestep)
lidar.enablePointCloud()
```

    b. Go to the **Menu Bar** → click on **View** → move your cursor to **Optional Rendering** → select **Show Lidar Point Cloud**.

    c. Click on the  Play button to start the simulation and view the point cloud.

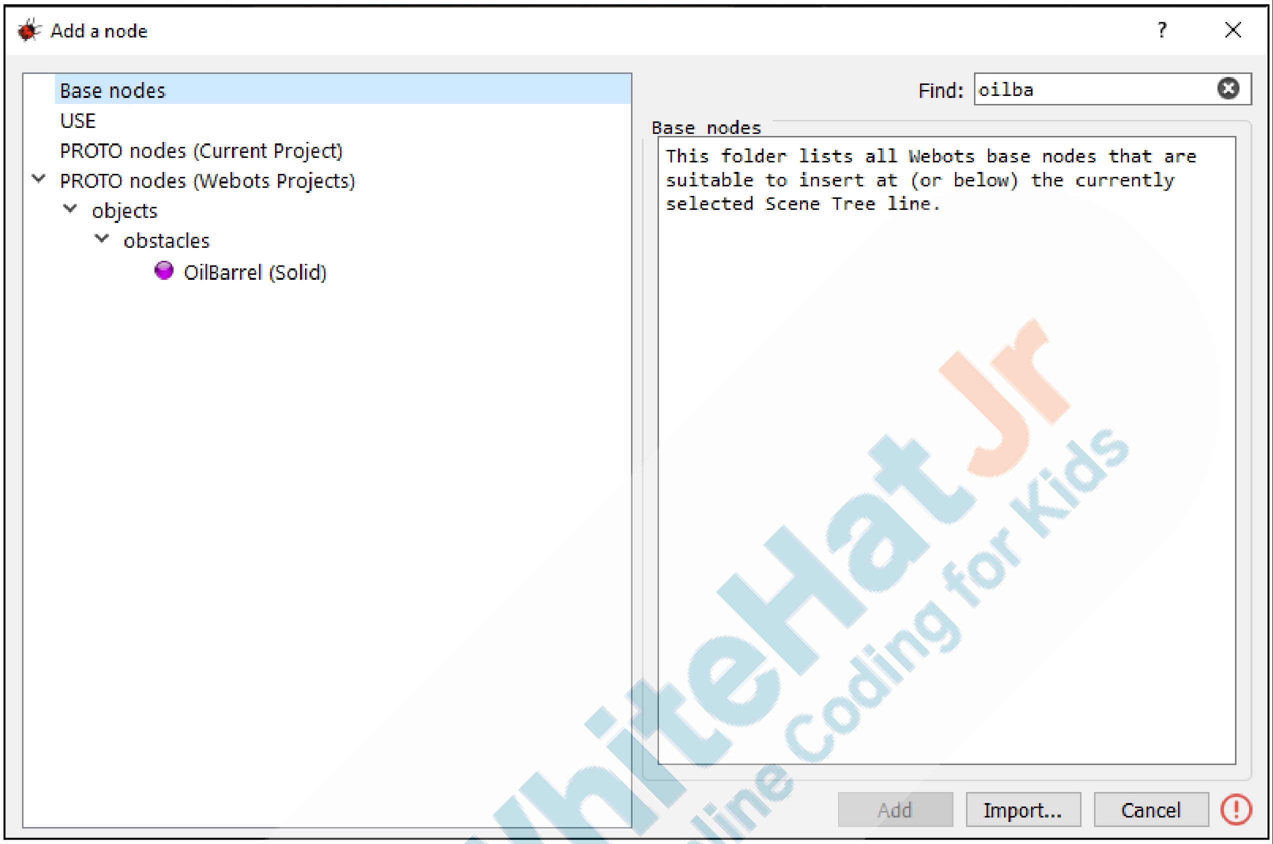


Expected Output:

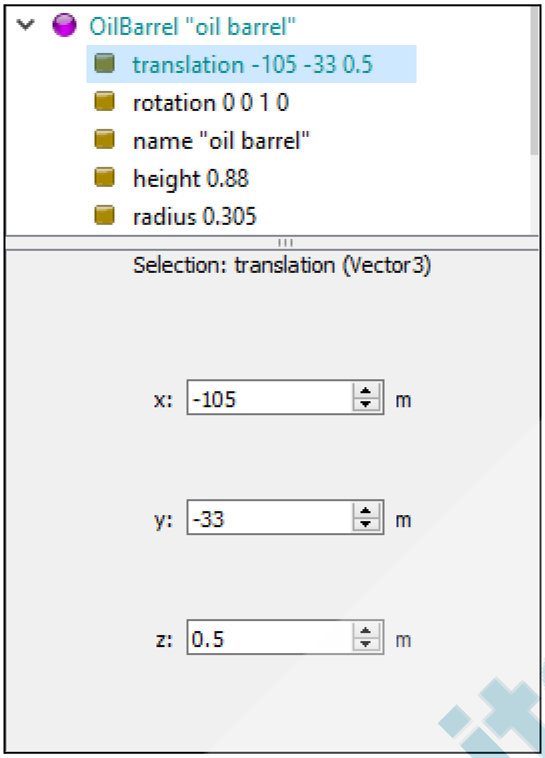


Before you start coding, let's add an obstacle as well.

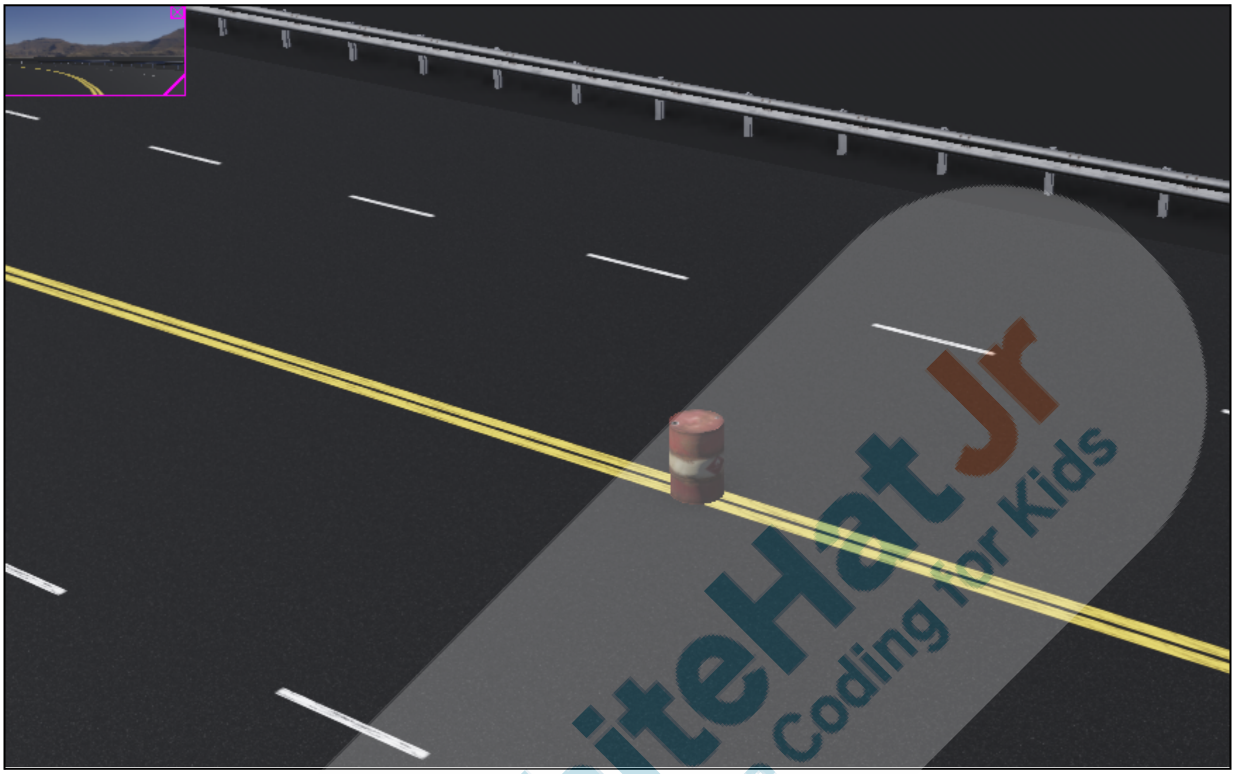
Click on the Add New  button → expand the **Proto nodes (Webots Projects)** → expand **objects** → expand **obstacles** → add **OilBarrel (Solid)**



Position the **OilBarrel** above the yellow line on the road. For reference, you can change translation property of the **OilBarrel** as following-	

OilBarrel "oil barrel"
translation -105 -33 0.5
rotation 0 0 1 0
name "oil barrel"
height 0.88
radius 0.305
Selection: translation (Vector3)
x: -105 m
y: -33 m
z: 0.5 m



Now, let's write the rest of the code obstacle detection. You will write that portion of the code. Are you excited?	**ESR:** Yes. Let's start.
So now it's your turn. Please share your screen with me.	
STUDENT-LED ACTIVITY 15 mins	
Ask the student to press the ESC key to come back to the panel. **Guide the student to start Screen Share.** **The teacher gets into Full Screen.**	
Student Initiates Screen Share	
<u>ACTIVITY</u>	

<table>
<tr><th colspan="2">• Writing an algorithm for driving a car autonomously.</th></tr>
<tr><th>Teacher Action</th><th>Student Action</th></tr>
<tr><td>Teacher guides the student to download the boilerplate code from <u>Student Activity 1</u>.</td><td>Student downloads the boilerplate code from <u>Student Activity 1</u>.</td></tr>
<tr><td>What should we do now?

Exactly! We need to write the code to get the range image of the Lidar. To do this, we will use the getRangeImage() method. The getRangeImage() method returns a one-dimensional list of floating point numbers.

If you're scanning any object the laser beam comes into contact with, each virtual point would represent a real point on the object. These points are returned by the getRangeImage() method.</td><td>ESR: We need to write code to detect an obstacle.</td></tr>
<tr><td colspan="2"><pre>
# main loop
while bot.step(timestep) != -1:

    # lidar values, affected rays and let's check if there is a obstacle or not
    val = lidar.getRangeImage()
</pre></td></tr>
<tr><td>Let's see what is returned by the getRangeImage() method.</td><td></td></tr>
<tr><td colspan="2"><pre>
# main loop
while bot.step(timestep) != -1:

    # lidar values, affected rays and let's check if there is a obstacle or not
    val = lidar.getRangeImage()
    print(val)

</pre></td></tr>
<tr><td>Now, if we observe the output, it will keep returning an array. This array will be of length 180, because the resolution of the Lidar was 180.</td><td></td></tr>
</table>

Then if you move the vehicle near the oil-barrel, the output will change. As the oil-barrel is in the center, it will affect the rays in the middle i.e. rays from index 85 to 96 in the array.

For our convenience, we will store only this portion of the array in a new variable.

```
# main loop
while bot.step(timestep) != -1:

    # lidar values, affected rays and let's c
    val = lidar.getRangeImage()
    print(val)
    affected_rays = val[85:96]
```

If there is nothing in front of the Lidar, then the value of each point will be **inf** (infinity). Otherwise, if the value decreases below 5 that means there is some object in front of the Lidar.

```
# main loop
while bot.step(timestep) != -1:

    # lidar values, affected rays and let's check if there is a obstacle or no
    val = lidar.getRangeImage()
    print(val)
    affected_rays = val[85:96]

    obstacle_detected = any([value < 5 for value in affected_rays])

    # print(obstacle_detected)
```

Reference Code:

```
# main loop
while bot.step(timestep) != -1:

    # lidar values, affected rays and let's check if there is a obstacle or not
    val = lidar.getRangeImage()
    affected_rays = val[85:96]
    obstacle_detected = any([value < 5 for value in affected_rays])
    print(obstacle_detected)

    # image data
    img = cam.getImage()
    image_width = cam.getWidth()
    image_height = cam.getHeight()
```

Now let us write the code to make a deviation if an obstacle is detected in the path.. Otherwise, we will following our previous algorithm.

1. Before writing the code for obstacle detection, let us define a wait method. We will define it before the main loop. We will use this method later, when we want to move the vehicle in a certain direction for a specified time.

```
right_wheel.setVelocity(0)
lidar.enable(timestep)
lidar.enablePointCloud()

def wait(time_steps):
    time_counter = 0
    while bot.step(timestep) != -1:
        if time_counter >= time_steps:
            break
        time_counter += 1

# main loop
while bot.step(timestep) != -1:

    # lidar values, affected rays and let's
    val = lidar.getRangeImage()
    affected_rays = val[85:96]
```

2. If an obstacle is detected, we will turn the vehicle to the left. Write this code inside the main while loop.

```python
    # if obstacle detected, take a turn
    if obstacle_detected:
        l_steer.setPosition(-0.7)
        r_steer.setPosition(-0.7)
```

3. Now, the wheel should retain this position for a few seconds so that the vehicle goes in the left direction for some time.
   Call the **wait()** method here.

```python
    # if obstacle detected, take a turn
    if obstacle_detected:
        l_steer.setPosition(-0.7)
        r_steer.setPosition(-0.7)
        wait(10)

```

4. Then, we want the car to move forward 20 timesteps.

```python
    # if obstacle detected, take a turn
    if obstacle_detected:
        l_steer.setPosition(-0.7)
        r_steer.setPosition(-0.7)
        wait(10)

        l_steer.setPosition(0)
        r_steer.setPosition(0)
        wait(20)

```

5. After that we want the car back to the right so that the car can reposition itself.

```
    # if obstacle detected, take a turn
    if obstacle_detected:
        l_steer.setPosition(-0.7)
        r_steer.setPosition(-0.7)
        wait(10)

        l_steer.setPosition(0)
        r_steer.setPosition(0)
        wait(20)

        l_steer.setPosition(0.7)
        r_steer.setPosition(0.7)
        wait(10)
```

6. What should we do if there is no obstacle detected?

ESR: We want to continue following the yellow line.

Correct! Let's write the code for that.
We will add that portion of code under the **else** section.

```
    # if obstacle detected, take a turn
    if obstacle_detected:
        l_steer.setPosition(-0.7)
        r_steer.setPosition(-0.7)
        wait(10)

        l_steer.setPosition(0)
        r_steer.setPosition(0)
        wait(20)

        l_steer.setPosition(0.7)
        r_steer.setPosition(0.7)
        wait(10)

    else:
        if x_average < x_center:
            l_steer.setPosition(-0.1)
            r_steer.setPosition(-0.1)
        elif x_average > x_center:
            l_steer.setPosition(0.1)
            r_steer.setPosition(0.1)
```

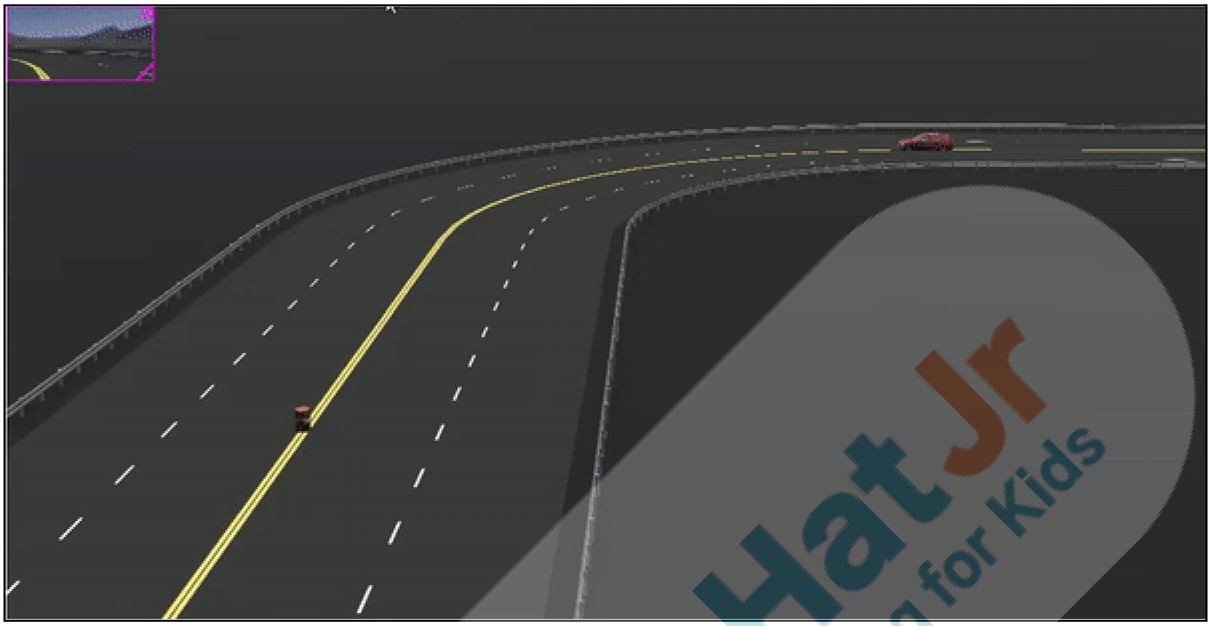
Reference Code:

```python
from controller import Robot

bot = Robot()

timestep = 64

# getting devices
cam = bot.getDevice('camera')
left_wheel = bot.getDevice('left_front_wheel')
right_wheel = bot.getDevice('right_front_wheel')
l_steer = bot.getDevice('left_steer')
r_steer = bot.getDevice('right_steer')
lidar = bot.getDevice('Sick LMS 291')

# initialisations
cam.enable(timestep)
left_wheel.setPosition(float('inf'))
right_wheel.setPosition(float('inf'))
l_steer.setPosition(0)
r_steer.setPosition(0)
left_wheel.setVelocity(0)
right_wheel.setVelocity(0)
lidar.enable(timestep)
lidar.enablePointCloud()

def wait(time_steps):
    time_counter = 0
    while bot.step(timestep) != -1:
        if time_counter >= time_steps:
            break
        time_counter += 1

```

```
# main loop
while bot.step(timestep) != -1:

    # image data
    img = cam.getImage()
    image_width = cam.getWidth()
    image_height = cam.getHeight()

    # processing image, method 1
    # getting average position of yellow pixels
    # getting total yellow pixels

    x_yellow = []
    for x in range(0, image_width):
        for y in range(0, image_height):
            red_val = cam.imageGetRed(img, image_width, x, y)
            green_val = cam.imageGetGreen(img, image_width, x, y)
            blue_val = cam.imageGetBlue(img, image_width, x, y)
            if red_val > 190 and green_val > 180 and blue_val > 90:
                x_yellow.append(x)

    # finding average of yellow pixels
    if x_yellow: # if there are any yellow pixels
        x_total = 0
        for x in x_yellow:
            x_total = x_total + x
        x_average = x_total / len(x_yellow)

    # rotating steering angle so that yellow lane remains in the center
    x_center = image_width / 2

```

```python

    # if obstacle detected, take a turn
    if obstacle_detected:
        l_steer.setPosition(-0.7)
        r_steer.setPosition(-0.7)
        wait(10)

        l_steer.setPosition(0)
        r_steer.setPosition(0)
        wait(20)

        l_steer.setPosition(0.7)
        r_steer.setPosition(0.7)
        wait(10)

    else:
        if x_average < x_center: # max pixels are on the left, take a left turn
            l_steer.setPosition(-0.1)
            r_steer.setPosition(-0.1)
        elif x_average > x_center: # max pixels are on the right, take right turn
            l_steer.setPosition(0.1)
            r_steer.setPosition(0.1)

    # move forward
    left_wheel.setVelocity(10)
    right_wheel.setVelocity(10)

```

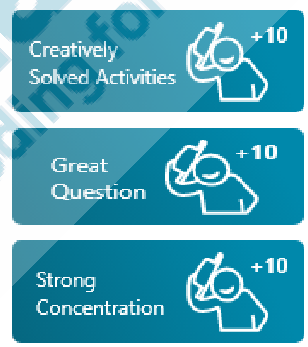
Reference Output:

 [Click here](#) to view the reference video.
Teacher Guides Student to Stop Screen Share
WRAP-UP SESSION - 5 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)

<u>FEEDBACK</u>

- **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! Congratulations for the completion of your 300th class with us! You have achieved a big milestone. We have come a long way since you started coding. We made games, mobile applications, AR/VR apps and much more. We also learnt to build electronic circuits and to design and program robots. This is not the end. You can learn and create much more by applying what you have learned so far. Our team will get in touch with you and share what you can build next. Enjoy learning and keep making new projects! All the best!	*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	Teacher Boilerplate Code	https://github.com/procodingclass/PRO-C300-Teacher-Boilerplate
Teacher Reference 1	Project	
Teacher Reference 2	Project Solution	
Teacher Reference 3	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/856bac10-0d12-4bc5-82e6-ff134328f62b.pdf
Teacher Reference 4	Reference code	https://github.com/procodingclass/PRO-C300-Reference-Code
Teacher Reference 5	Final output gif	https://s3-whjr-curriculum-uploads.whjr.online/d7cd3c0e-899a-4405-ab07-2401f529de02.mp4
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C300-Student-Boilerplate