Topic	SELF DRIVING CAR 1	
Class Description	**Students will learn how to create a car driving track environment in webots.**	
Class	**PRO C298**	
Class time	**50 mins**	
Goal	• Designing a car driving track. • Importing the car node.	

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**

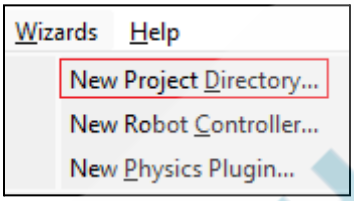
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show 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 plane node into the environment.**	
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 created a room cleaning robot. **ESR** : Varied

Let me ask you a simple question. Do you like to drive a vehicle or do any of your friends drive it? Great, then you might know that, although driving a vehicle is fun, if someone doesn't obey traffic rules, it might lead to a traffic accident. Can you tell me some of the reasons that might lead to a traffic accident?	**ESR** : Yes, some of the reasons are • **Over speeding.** • **Drunk driving.** • **Distractions to the driver.** • **Red light jumping.** • **Avoiding safety gears like seat belts, helmets etc.** • **Non adherence to lane driving and overtaking.**
Great, you are correct. If you observe carefully, you will see that the majority of the reasons are somewhat associated with the carelessness of the driver like **over speeding, drunk driving** etc. Can you tell me how we can avoid traffic accidents? What if we can create an **autonomous vehicle** or a **robotic car** or better known as a **self driving car**. Wouldn't it be cool?	**ESR :** Varied **ESR** : Yes
Great, let's open the webots application and start creating this project. But before creating the self-driving car, first we need to create a **track** or **roads,** where the car can run. For that, let's open the **webots application** and **pause** the	

last simulation if it's already **running**.

After that, from the top menu bar, click on the **Wizards** option.

A drop down menu will appear. Select the **New Project Directory** option.



A dialog box will appear which will ask you the following questions,

a) If you want to create a new project or not?

- Click on **Next**.

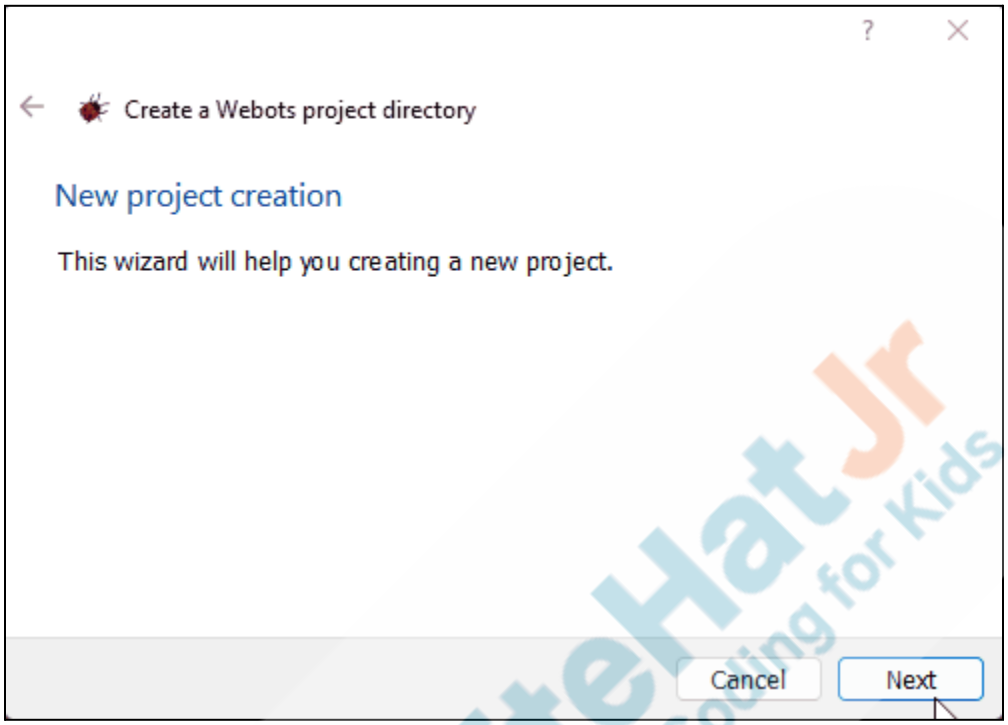
b) It will ask you to either **choose** an **existing directory** or **create** a **new directory** so that you can **store** all the **assets** of your project in a single directory.

- Create a new directory and name it as **sdc.**

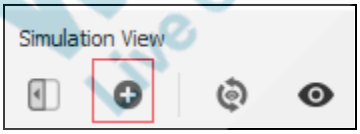
c) Next it will ask to **give a name** to the world and **choose the features** that you want in your project world.

- Write the name as **sdc.wbt** and don't check mark **Add a rectangular arena** option.

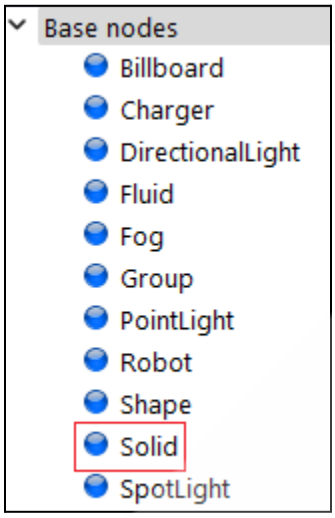
d) Finally, click on **finish.** Your new project world will be created.



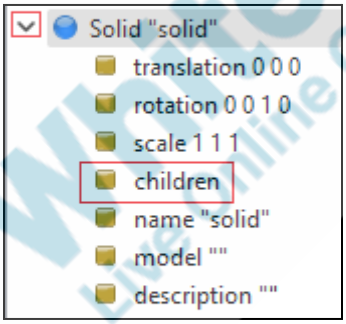
Once the new world is created, let's add a **plane shaped node** which will act as the **ground** for laying down the track. For that, click on the **Add object** button.



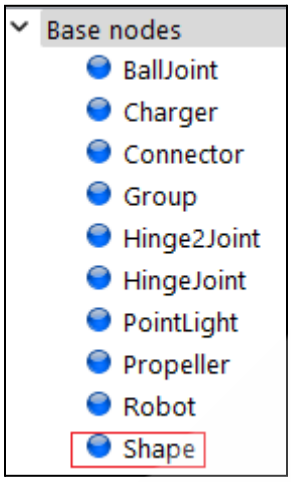
Expand the **Base nodes** and select the **Solid** node.



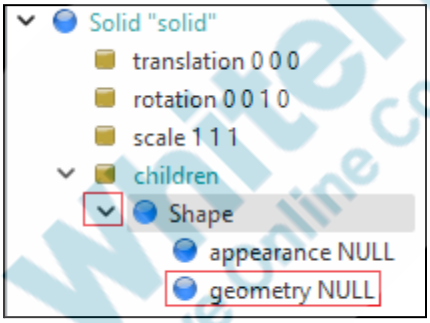
You will see that a solid node will be added. To add a plane shaped node, **expand** the **Solid node** and **double click** on the **children** node.



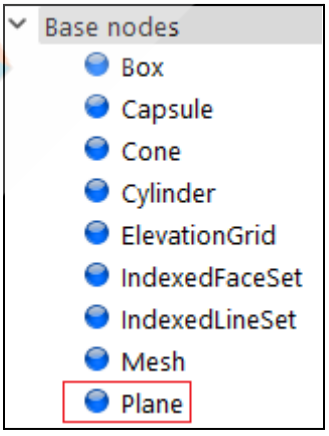
From the **Base nodes**, add the **Shape node**.

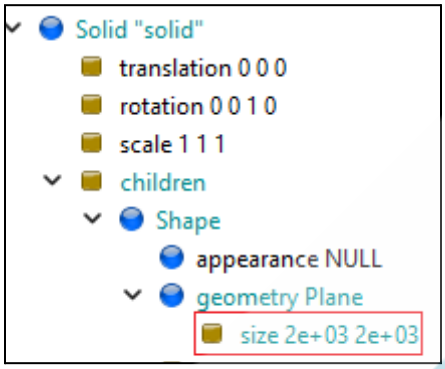
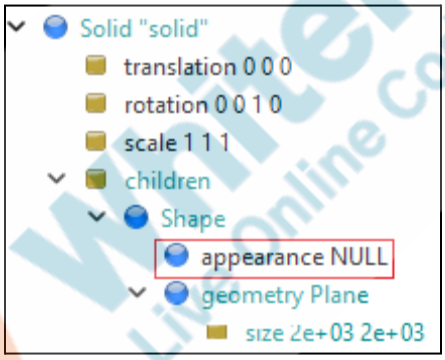
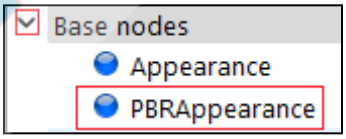


To give the solid a plane shaped geometry, **expand** the **shape** node and **double click** on the **geometry node**.

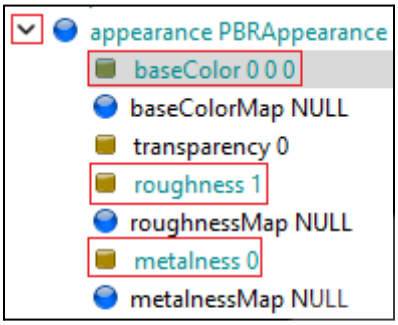


From the **Base nodes**, select **Plane**.

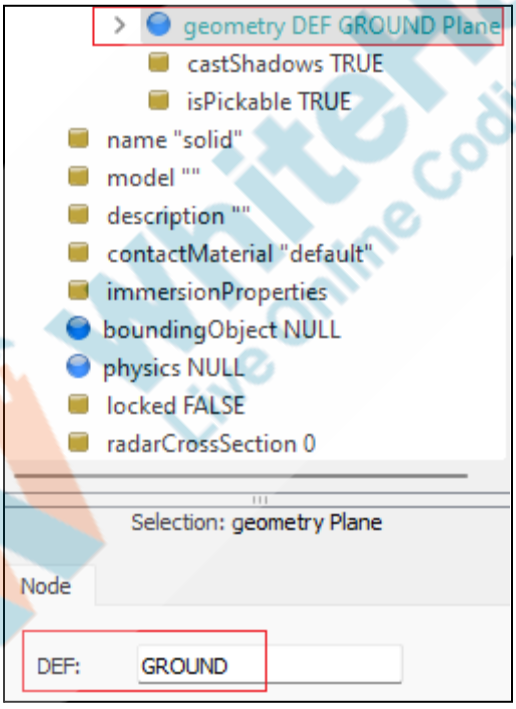


To give it a shape of specific dimensions, **expand** the **geometry** node, and **change** the **size** to **2000 m** in **X direction** and **2000 m** in **Y direction**.	
	
Once you have defined the size, **double click** on the **appearance** node.	
	
From **Base nodes**, select the **PBRAppearance node**.	
	
To define a proper appearance, • **Expand** the **PBRAppearance** node. • Specify the **base color** as **0 percent red, 0 percent green** and **0 percent blue**. • Specify the **roughness** a **1**.	

- Specify the **metalness** as **0**.

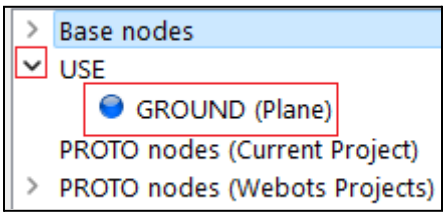


Finally, let's name the **Plane geometry node** as **GROUND** so that we can use it while defining the **bounding box** for the **Plane solid node**.

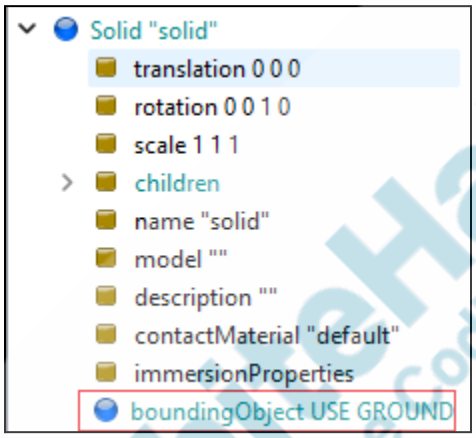


Once you have named the geometry node, let's define the bounding box for our solid. For that,

- **Double click** on the **bounding box** property.
- **Expand** the **USE** menu.
- Select the **GROUND** node.



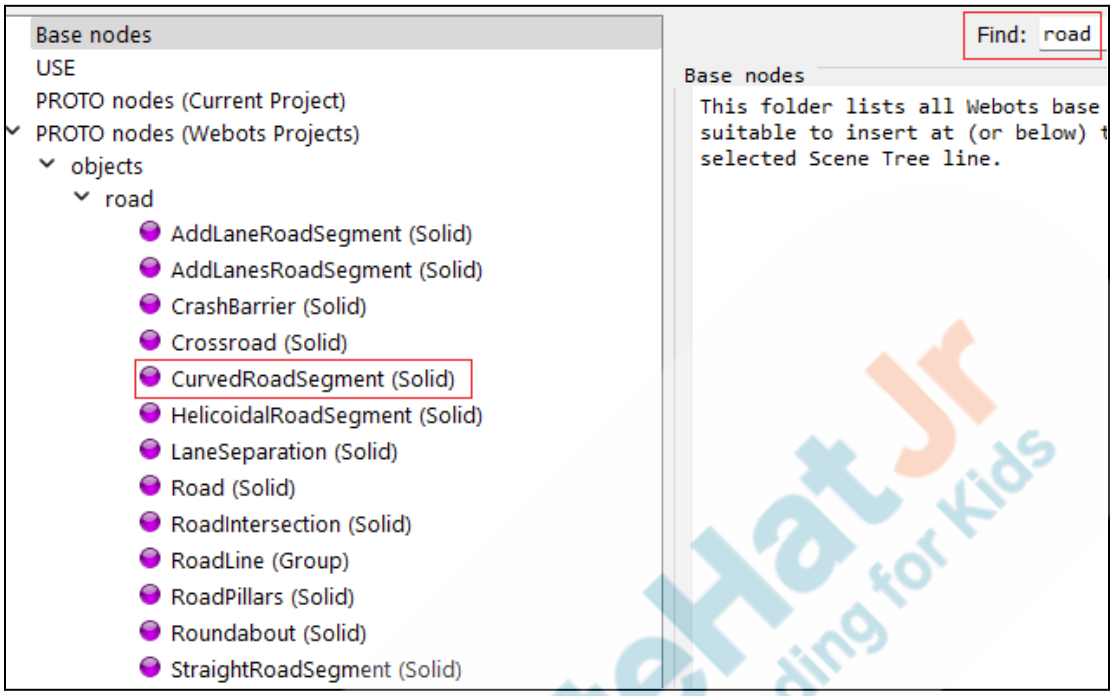
You will see that the bounding box property for the solid node will be defined.



Save your work till here and If you zoom out, you will be able to see the plane.

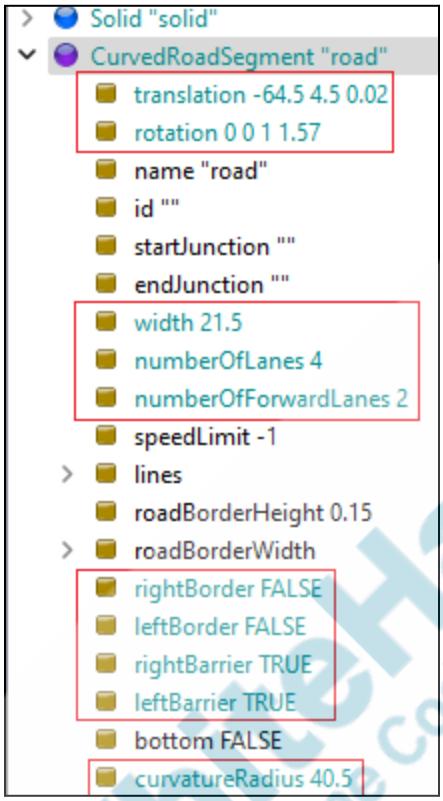
Teacher Stops Screen Share

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	
ACTIVITY • **Adding different road segment nodes.** • **Adding a car node.**	
Teacher Action	**Student Action**
Open the student boilerplate link, and download all the files. Open the downloaded files in the webots software. Now that we have added the ground structure, let's lay down the driving track over it. For that, click on the **Add object** button and **search** for the word **'road'** in the **'Find' textbox** and double click on the **curved road segment node**.	

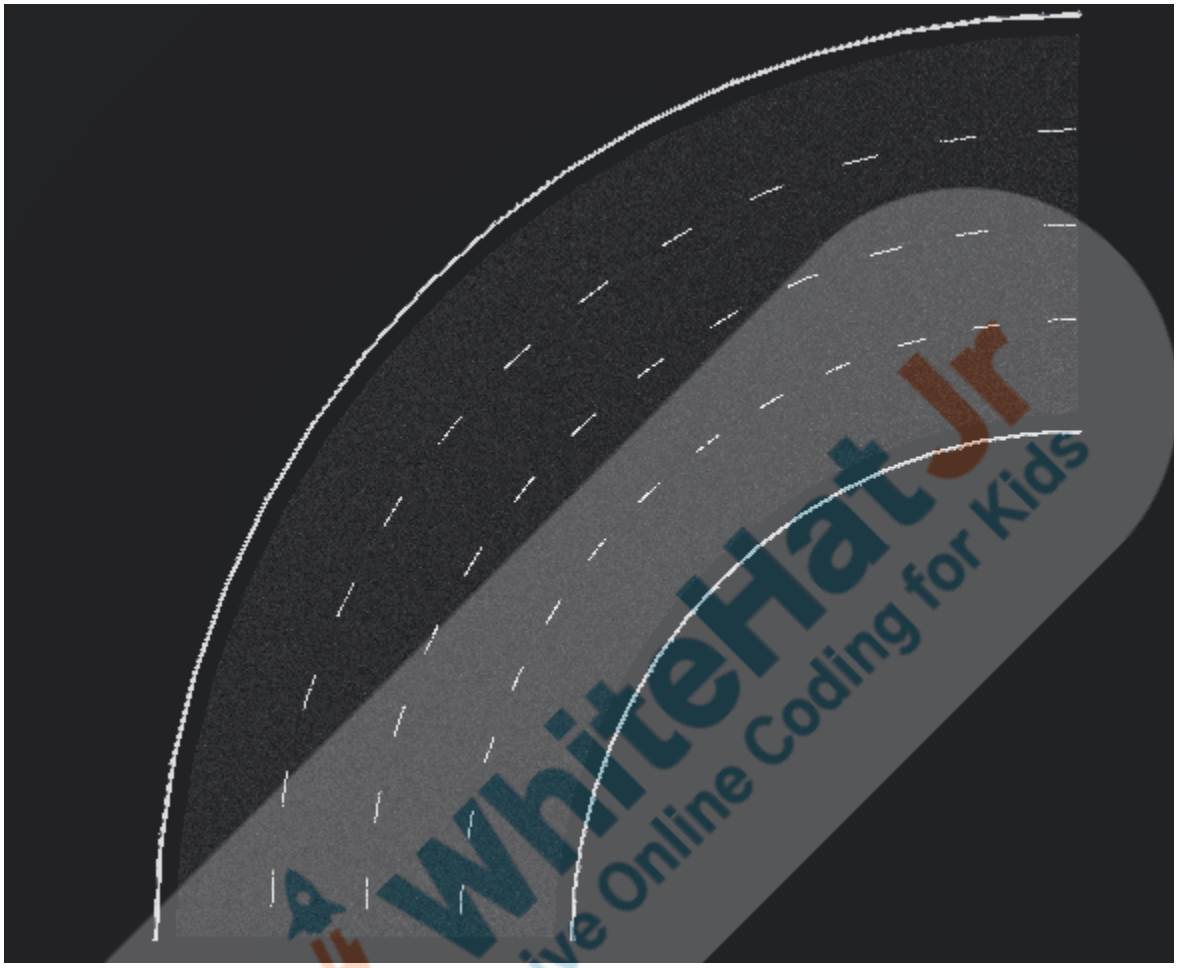


Let's **expand** the **curved road segment** node so that we can change its properties. Next, let's change its,

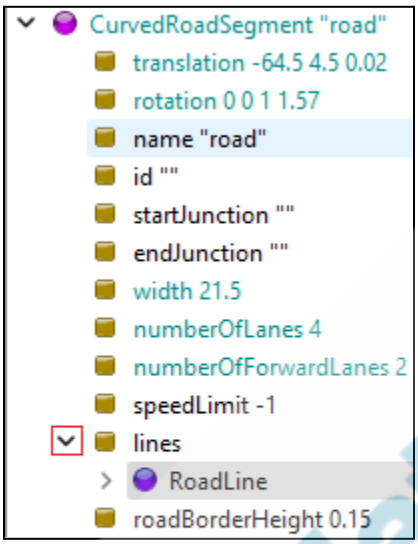
- **Translation** to **-64.5 m** in **X direction, 4.5 m** in **Y direction** and **0.02 m** in the **Z direction**.
- **Rotation** to **1.57 radians** in the **Z direction**.
- **Width** as **21.5 m**.
- **Number of lanes** to **4**.
- **Number of forward lanes** to **2**.
- **Right** and **left border** as **False**.
- **Right** and **Left barrier** as **True**.
- **Curvature radius** as **40.5 m**.



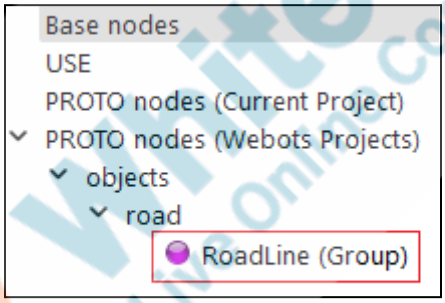
Save your work till here.



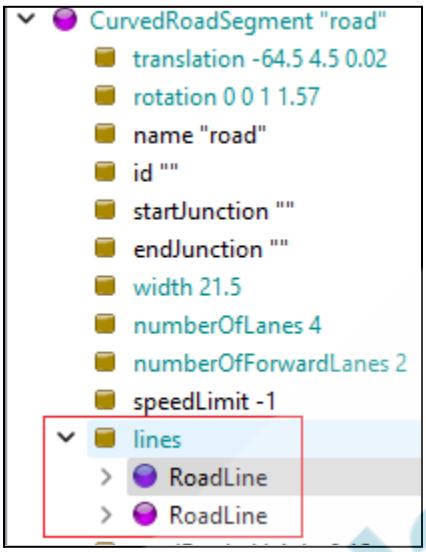
Once we have added the curved road segment, let's change the **color** off the **middle line**, so that it can be used for writing the **track following algorithm** later. For that, let's first **expand** the **lines** property.	



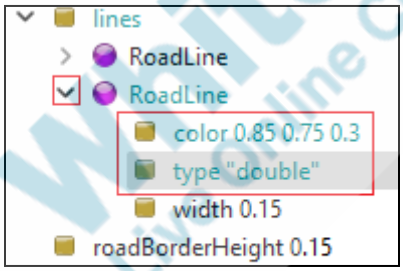
After expanding the lines property, click on the **add object button.** Select the **RoadLine** node.



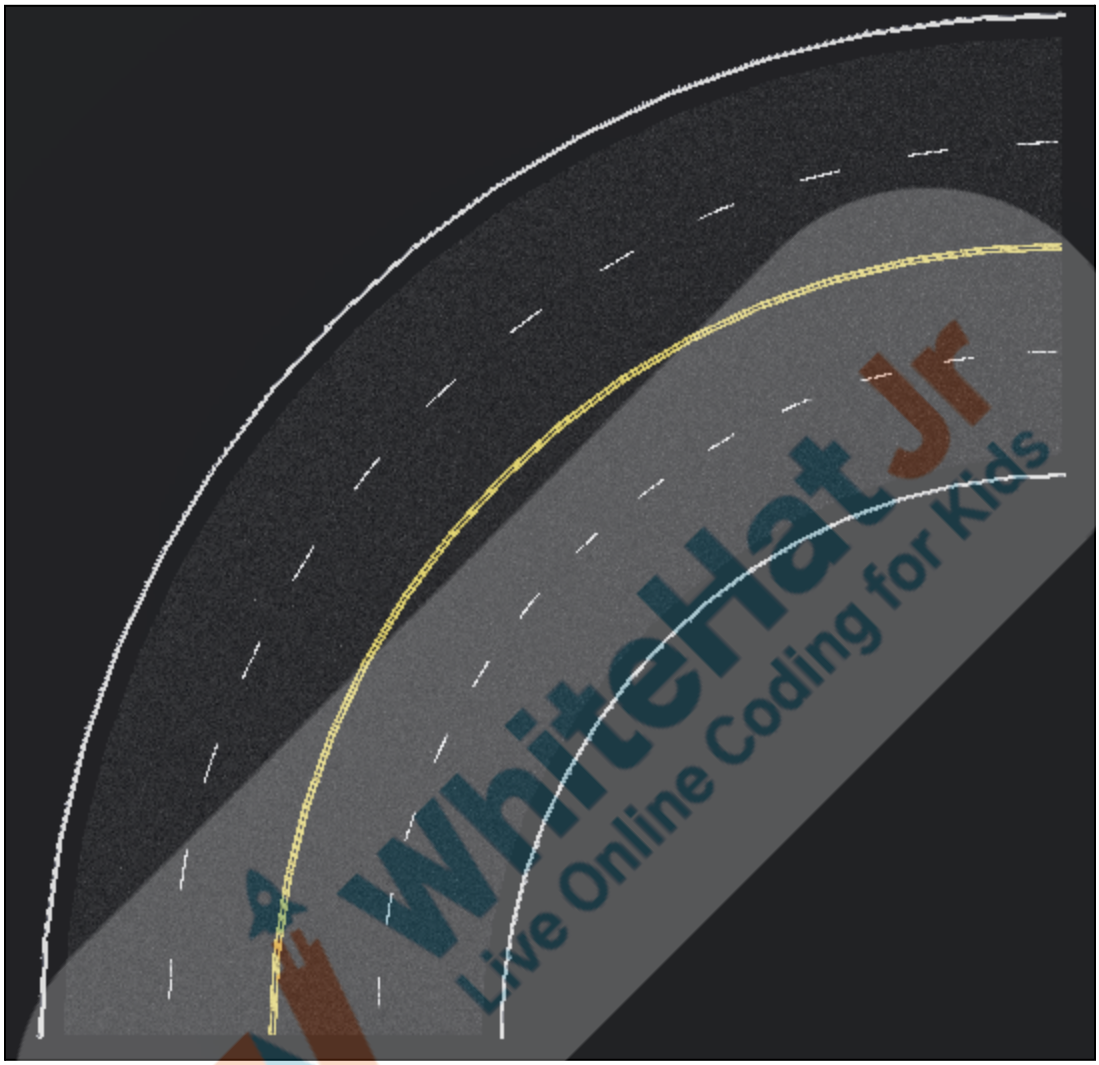
You will see that a new **RoadLine node** is added under the **lines** property.



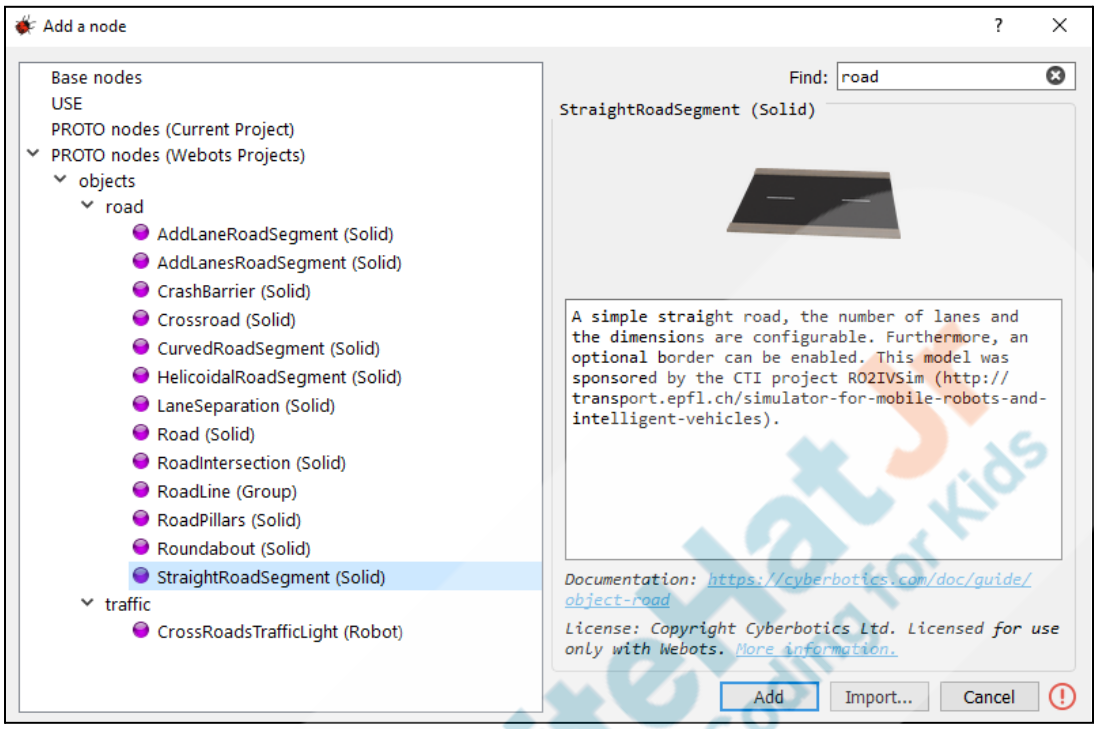
Expand the **second RoadLine node** and change the color to **0.85** in **red, 0.75** in **green** and **0.3** in **blue**. Also change the type to **"double".**



Save your work till here.

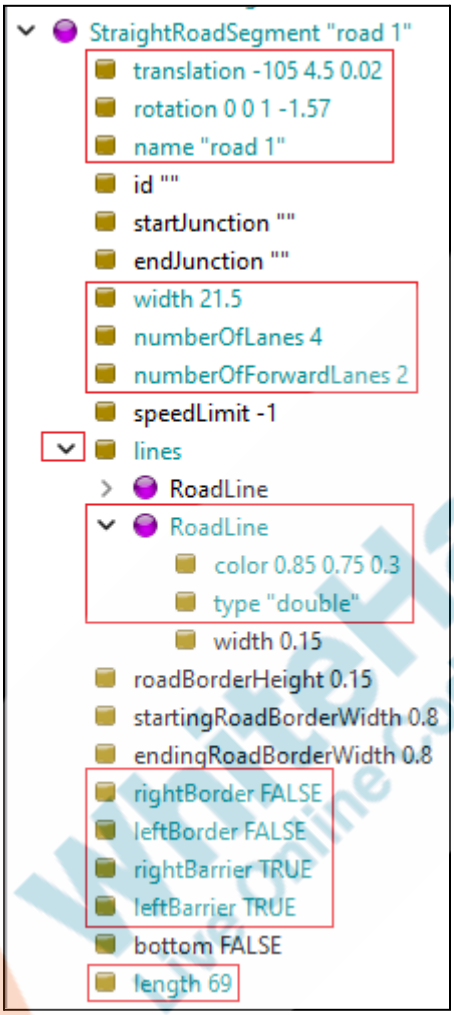


Next, we need to add a **straight road segment.** For that, click on the **add object button**, search for the word **“road”** and add a **straight road segment node**.	



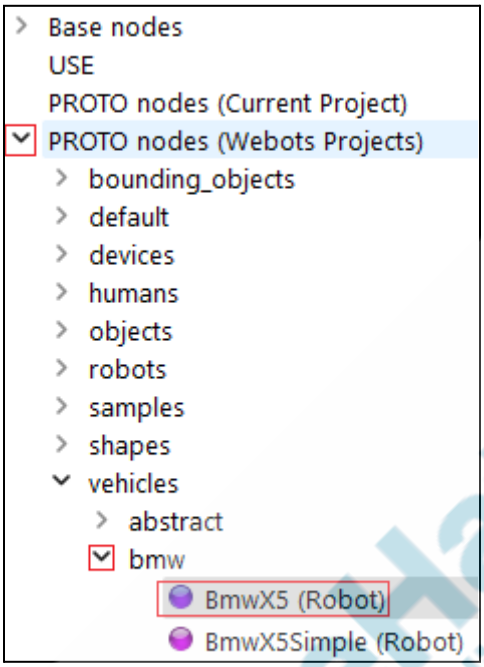
Expand its properties and change its,

- **Translation** to **-105 m** in the **X direction, 4.5** in the **Y direction** and **0.02 m** in the **Z direction**.
- **Rotation** to **1.57 radians** in the **Z direction**.
- **Name** as **“road 1”**.
- **Width** as **21.5 m**.
- **Number of lanes** to **4.**
- **Number of forward lanes** to **2**.
- **Add** a **RoadLine node** under **lines** property and change its **color** to **0.85** in **red, 0.75** in **green** and **0.3** in **blue**. Also change its **type** to **double**.
- **Right** and **left border** to **False**.
- **Right** and **left barrier** to **True**.
- **Length** to **69 m**.



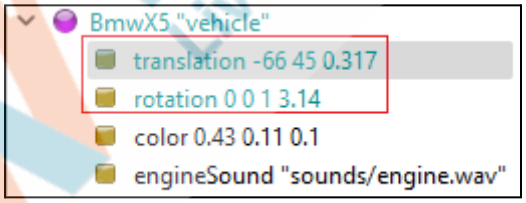
Once you have added the **curved and straight road segments**, let's add the car node. For that,

- Click on the **add object** button.
- **Expand PROTO** nodes.
- **Expand vehicles**.
- **Expand bmw**.
- Add a **BMW X5 (Robot) node**.

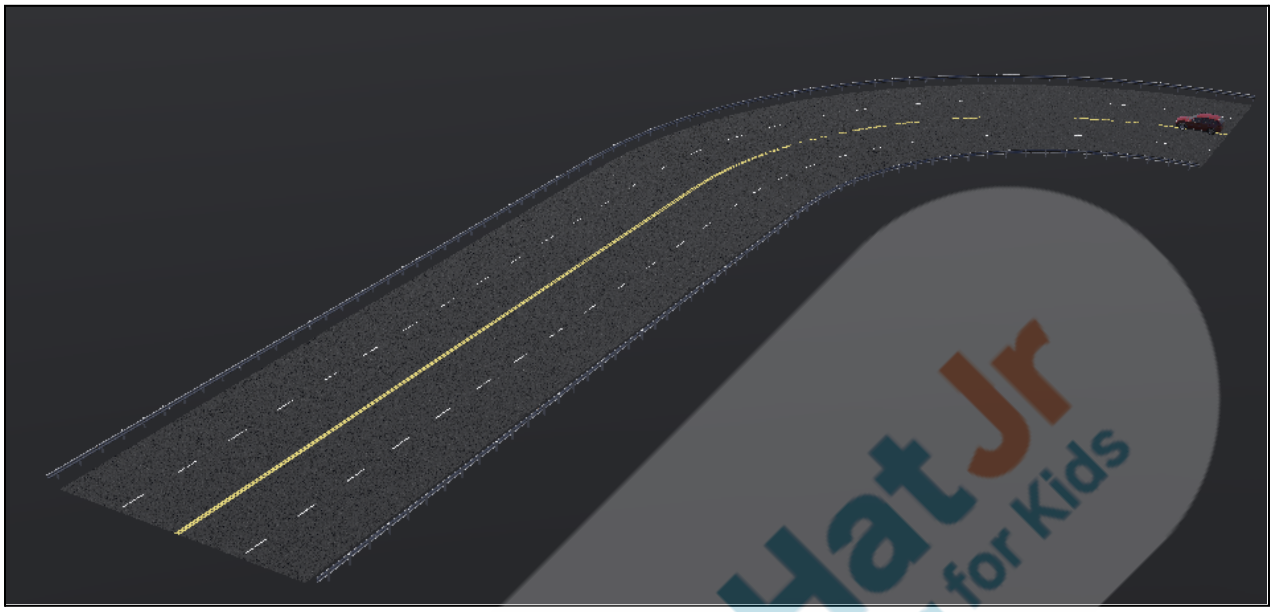


Expand it and change its **translation** to **-66 m** in the **X direction, 45 m** in the **Y direction** and **0.317 m** in the **Z direction**.

Also change its **rotation** to **3.14 radians** in the **Z direction**.



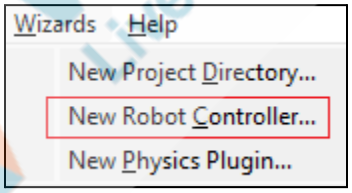
Save your work till here.



Once we are done with the designing part, let's write some **code** so that we can move our car.

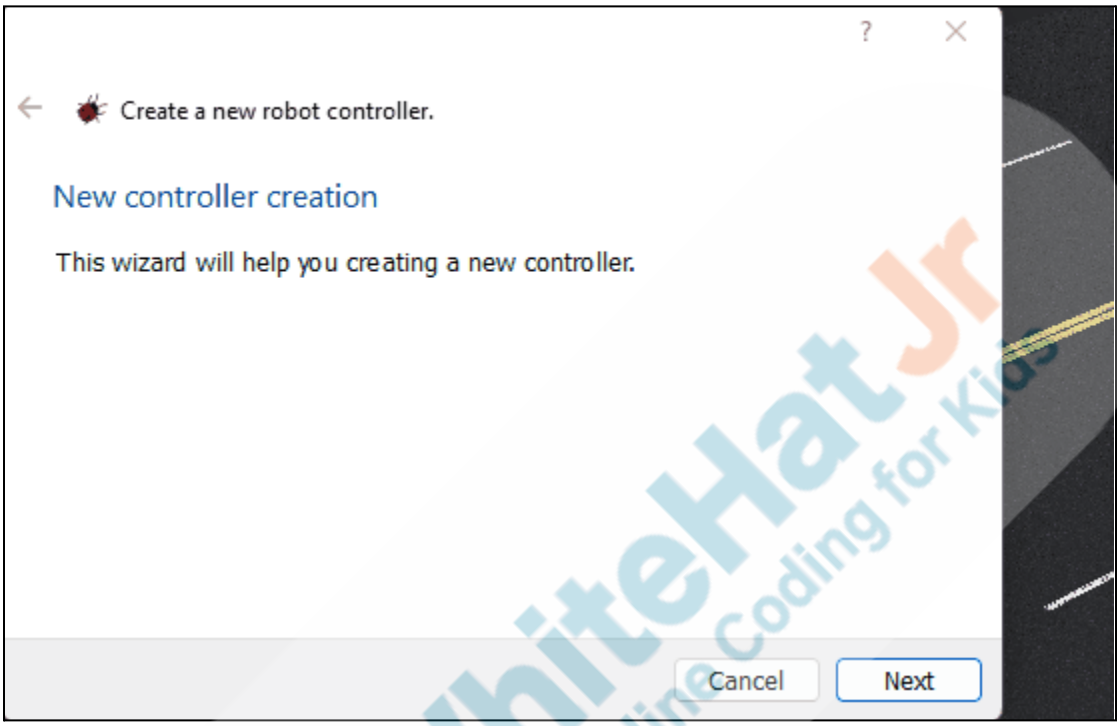
For that, let's create a **new python controller** as,

a) Click on the **Wizards** window on the top menu bar.
b) Select the **New Robot Controller** option.



A window will appear which will ask you,

a) If you want to create a new controller or not.
   - Click on **Next**.
b) It will then ask you the language in which you want to write the code.
   - Select **python** and click on **Next**.
c) Finally, it will ask you to name your controller file.
   - Write the file name as **sdc1** and click on **Next**. Although you can write any name you

want. d) Finally, click on **Finish**.	
	
You will see that a **python file** named as **sdc.py** will be opened with a standard template code in it. **Delete** all the code from it as we will write our own code.	

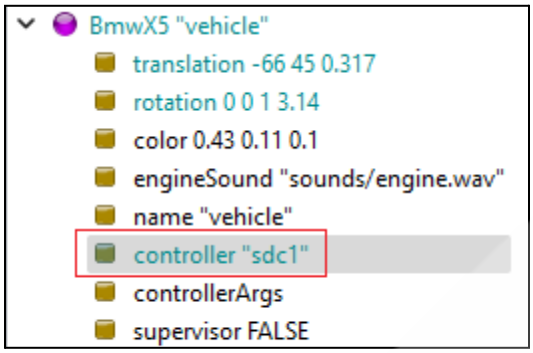
one_arm_robot.py

```
"""one_arm_robot controller."""

# You may need to import some classes of the controller module. Ex:
#  from controller import Robot, Motor, DistanceSensor
from controller import Robot

# create the Robot instance.
robot = Robot()

# get the time step of the current world.
timestep = int(robot.getBasicTimeStep())

# You should insert a getDevice-like function in order to get the
# instance of a device of the robot. Something like:
#  motor = robot.getDevice('motorname')
#  ds = robot.getDevice('dsname')
#  ds.enable(timestep)

# Main loop:
# - perform simulation steps until Webots is stopping the controller
while robot.step(timestep) != -1:
    # Read the sensors:
```

Before we start writing our code, let's specify that our **BMW X5 robot** will follow the code written in the **sdc1.py** file.

For that,

a) Expand the **BMW X5** node.
b) You will see tha **controller** property. Initially it shows **void**, which means no controller is selected yet.
c) **Double** click on the **controller** property.
d) Click on the **Select** button.
e) A window will appear, select **sdc1.py** from it.
f) The **controller** property will be **updated**.



Now go to the sdc1.py file and

- **Import** the **Robot** class from the **controller** module.
- Create an **instance** of the **Robot** class.
- Set the **time step** of the **controller** as **64 ms**.
- The motor on front left wheel and front right wheel, is named as **“left_front_wheel”** , **“right_front_wheel”** respectively. Using the **.getDevice(motor name)** method, fetch both the motors.
- Do the same for steering, which is named as **“left_steer”** and **“right_steer”**.

```
from controller import Robot

bot = Robot()

timestep = 64

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')
```

Once all the objects are created, set the positions for both the wheels as **‘infinity’** as we will control both of them using the **.setVelocity()** method.

```
left_wheel.setPosition(float('inf'))
right_wheel.setPosition(float('inf'))
```

Next, set the **initial positions** of the **steering** motors to **0** and set the **initial velocities** of both the **wheels** as **0**.

```
l_steer.setPosition(0)
r_steer.setPosition(0)
left_wheel.setVelocity(0)
right_wheel.setVelocity(0)
```

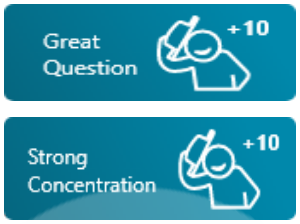
Finally, in the main loop, set the velocities of both the wheels as **10 rad/s**.

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

    left_wheel.setVelocity(10)
    right_wheel.setVelocity(10)
```

Save your work till here and run the simulation.

Click here to view the reference output.

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)	
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! In the next class, we are going to add the self-driving capabilities to this vehicle.	*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 Reference 1	Project	
Teacher Reference 2	Project Solution	
Teacher Reference 3	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/c4738ec0-4553-41a2-be9a-8b754a0b6ea4.pdf
Teacher Reference 4	Reference code	https://github.com/procodingclass/PRO-C298-Reference-Code.git
Teacher Reference 5	Final output gif	https://s3-whjr-curriculum-uploads.whjr.online/81b0c0b8-9b12-4342-8b0e-0fbfc2f2de25.gif
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C298-Student-Boilerplate.git