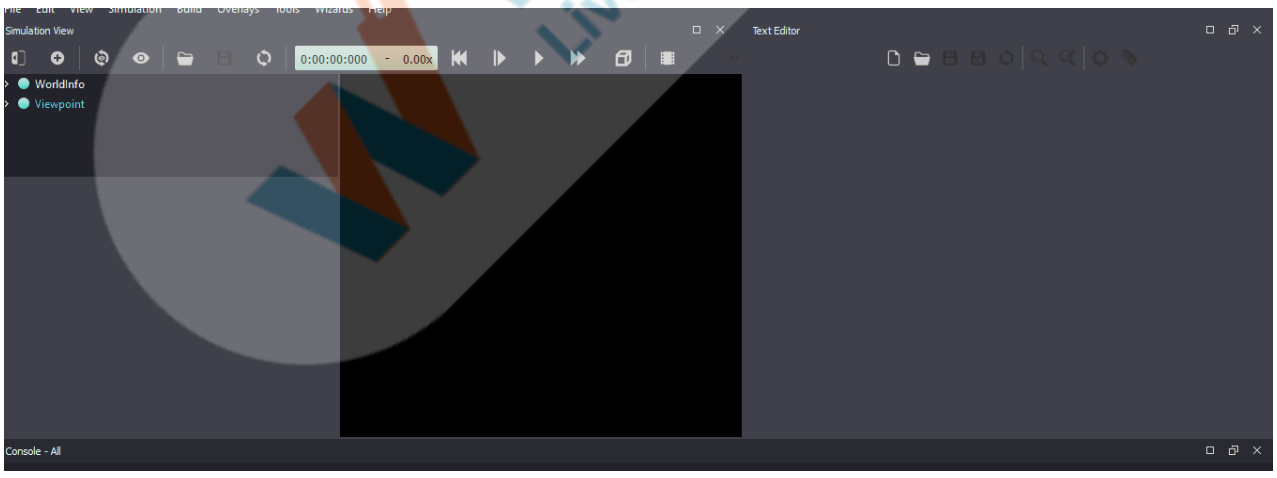
Topic	Exploring Webots Design	
Class Description	**Students will learn from scratch to design some geometrical figures on Robot.They will also learn about some design parameters and their functions to explore webots.**	
Class	**PRO C279**	
Class time	**45 mins**	
Goal	• Introduction to Webots Function • Design of Geometrical Figures	

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**	**05 mins** **15 mins** **15 mins** **10 mins**
Credit & Permissions:	**Code samples used for Firebase-Google Authentication are licensed under the https://cyberbotics.com/** **Expo documentation used from -https://cyberbotics.com/** **Note: Keep this row section only if applicable**	

WARM-UP SESSION - 05 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	
• **Introduction to Webots Function** • **Design of Geometrical Figures**	
Teacher Action	**Student Action**
Any doubts from the last class! *The teacher will clarify if there are any doubts!* So we installed and explored webots and we saw how inbuilt robots work. .	**ESR:** Varied!

So today we are exploring webots. We will learn how we can make some own functions in Webots We will focus on some design environment aspects. As designing will take usually a long time as compared to programming. Let's design some geometrical shapes on webots.	**ESR:** Yes!
Teacher will open the Webots	**Students will observe the teacher.**
A new Simulation: This simulation will contain a simple environment (a rectangle arena with floor and walls), one inbuilt Robot and a controller program that will make the robot move.	
Create a new world: **World:** A World defines the initial state of a simulation. A world is stored in a file having the **".wbt" extension.**	



https://s3-whjr-curriculum-uploads.whjr.online/63c3a92e-2585-4a86-b082-53fd0bf8f5ae.gif

So we set up our Rectangle arena.	
So today we will see how to design products from scratch. We will design some geometrical figures. Procedure for design will remain the same but we need to change some features to make different shapes. Practice is very important in this case as there are multiple steps to perform.	
Let's focus on some features and their uses first *Note: Teacher will explain these terms. All these terms will be used in the design part.* **World:** A world is a file defining a simulation. It includes nodes(Objects) organized hierarchically in a scene tree. **Scene tree**: The scene tree is a hierarchical tree structure containing nodes that can be saved as a world. **Base Nodes**: A base node is a Webots built-in shape. **Basic time step**: The basic time step is the time step increment used by Webots to advance the virtual time and perform physics simulation. **Node:** A node is a component of the scene tree. It defines a concept of the world. **Parent:** When referring to a node", the *a parent of a child is a node containing the child at a relative depth of one in the scene tree. Note that a parent is always an ancestor, but an ancestor is not necessarily a parent. This parent and child relation is very important to understand. To initiate anything we must focus on our parents. and then related to the parent anything will act as	

a child. **Children**: When referring to a node, the children of a parent is a node directly contained inside the parent (Base Nodes) , at a relative depth of one in the scene tree. Note that a child is always a descendant, but a descendant is not necessarily a children	
Transform : The Transform node is a grouping node that defines a coordinate system for its children that is relative to the coordinate systems of its parent. The **translation** field defines the translation from the parent coordinate system to the children's coordinate system.It will decide x, y, z position coordinates **Appearance:** The appearance field contains an Appearance or PBRAppearance node that specifies the visual attributes (e.g., material and texture) to be applied to the geometry. **PBR Appearance:** The PBRAppearance node specifies a physically-based visual appearance of a node. The acronym "PBR" refers to "Physically-Based Rendering", a term used to designate a class of shading models based on the physical properties of an object, **BaseColor:** The baseColor field specifies the base color or "albedo" of the material's surface, analogous to the diffuseColor field of the Material node. **Roughness** :The roughness field specifies the roughness of the material's surface, analogous to the inverse of the shininess field of the Material node. A roughness of 0 gives a perfectly smooth material, while a roughness of 1 gives a highly rough material. **Metalness:** The metalness field specifies how metallic the material's surface is. A metalness of 0 gives a completely non-metallic material, while a metalness of 1	

gives a completely metallic material.

Note: It is encouraged to use either 1 or 0 for this value as no real-world materials are partly metallic.

Geometry: The geometry field contains a Geometry node: Box, Capsule, Cone, Cylinder, ElevationGrid, IndexedFaceSet, IndexedLineSet, Mesh, Plane, PointSet or Sphere. The specified Geometry node is rendered with the specified appearance nodes applied.

So we got familiar with terms. I know it's not pretty clear but now by doing it we will understand better.

Procedure:

Geometrical Figure Cuboid

To make cuboid or any new thing we must click on + or

Add object sign

1. Click on **+**
2. Click on **Base nodes**
    a. Select **Transform**
    b. Write name of the Function DEF **BOX**

    (Note: DEF is already written)

    c. Click **ADD**. Function is created
    d. Now click on Arrow just before the function name

i. Double Click on **children**

ii. Click on **Base nodes**

1. Select **SHAPE**
2. Click on **ADD**
3. Now click on **Shape**
    a. Double Click on **Appearance**
    b. **Click on Base Nodes**
        i. Select **PBR Appearance**
        ii. Click **ADD**
4. **Click on Arrow PBR Appearance**
    a. Select **baseColor 0.8, 0.264, 0.264**
    b. Select **roughness 0.2**
    c. Select **metalness 0**
5. Double Click on **Geometry NULL**
    a. Select **Box**

b. Click **Add**

c. size **0.5, 0.3, 0.4**

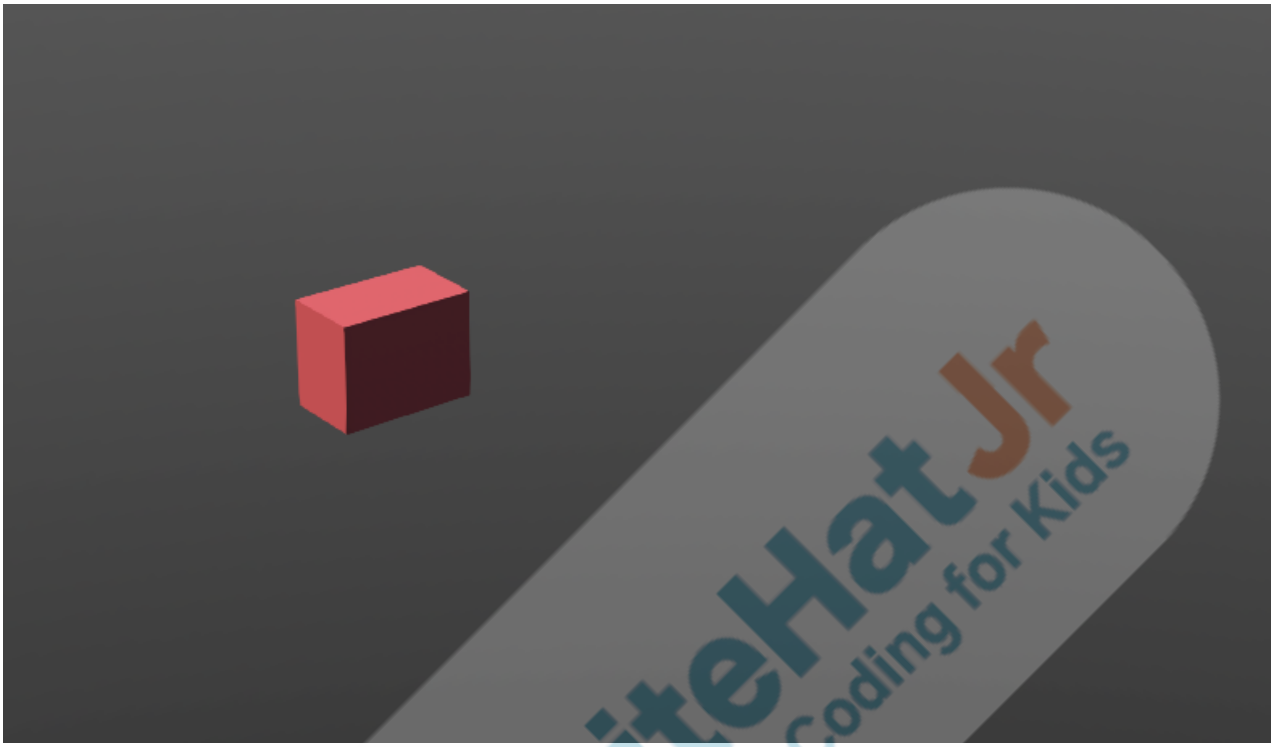
Save the simulation. 

Box code will look like this:

```
DEF BOX Transform {
  children [
    Shape {
      appearance PBRAppearance {
        baseColor 0.8 0.264 0.264
        roughness 0.2
        metalness 0
      }
      geometry Box {
        size 0.5 0.3 0.4
      }
    }
  ]
}
```

Reference Video Link to create a Box shape:
https://s3-whjr-curriculum-uploads.whjr.online/58f8b8c0-5be7-4651-a9c2-f3941e858a7c.mp4

Output will look like thisne



Geometrical Figure Cone

3. Click on **Base nodes**
    a. Select **Transform**
    b. Write name of the Function DEF **CONE**
    (Note: DEF is already written
    c. Click **ADD**. Function is created
    d. Now click on **Arrow** just before the function name
    e. Select **translation** and write value **2, 0, 0**
    f. Now click on **Arrow** just before the function

name

i. Double Click on **children**

ii. Click on **Base nodes**

1. Select **SHAPE**
2. Click on **ADD**
3. Now click on **Shape**
    a. Double Click on **Appearance**
    b. **Click on Base Nodes**
        i. Select **PBR Appearance**
        ii. Click **ADD**
    c. **Click on Arrow PBR Appearance**
    d. Select **baseColor 0.8, 0.62, 0.264**
    e. Select **roughness 0.2**
    f. Select **metalness 0**
4. Double Click on **Geometry NULL**

a. Select **Cone**

b. Click **Add**

c. **bottomRadius 0.3**

d. **height 0.5**

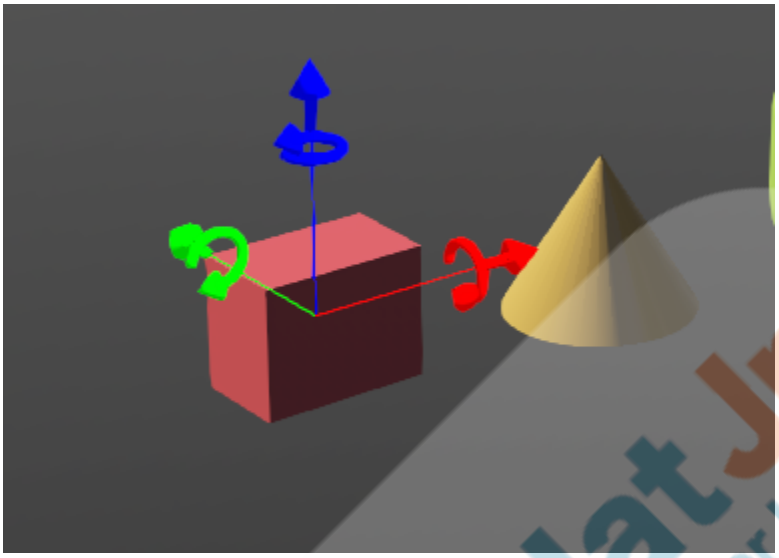
e. **subdivision 36**

Save the simulation. 

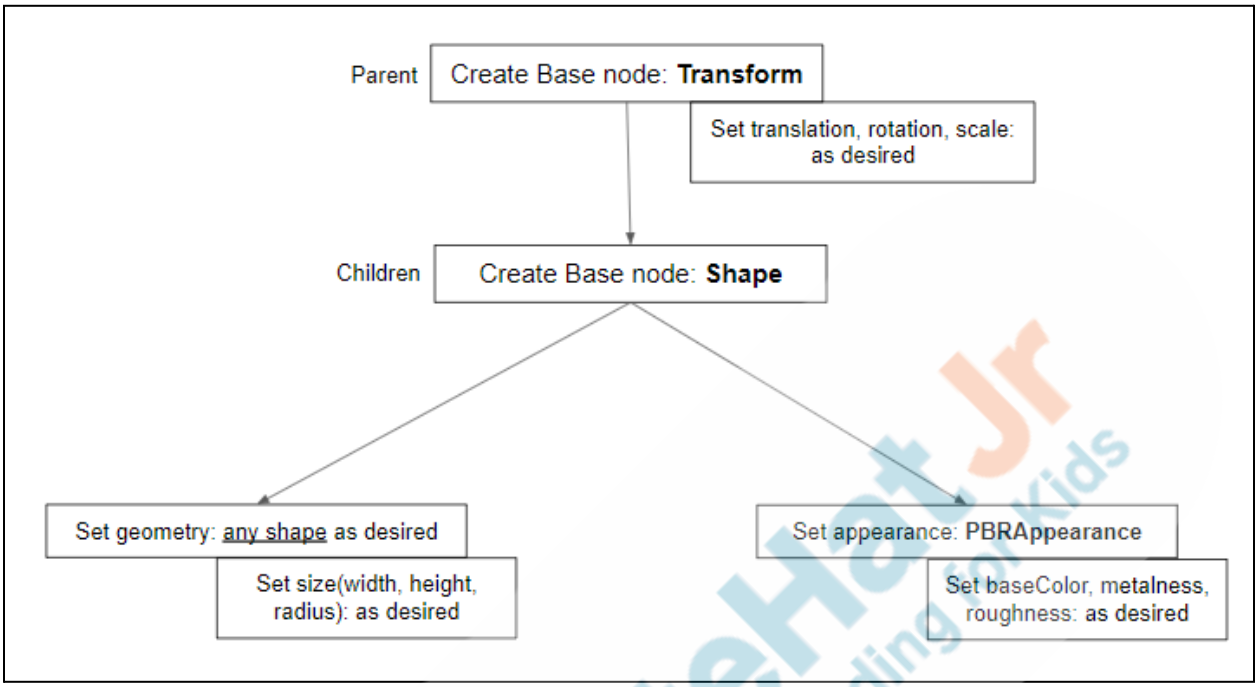
Cone code will look like this:

```
DEF CONE Transform {
  translation 1 0 0
  children [
    Shape {
      appearance PBRAppearance {
        baseColor 0.8 0.621333 0.264
        roughness 0.2
        metalness 0
      }
      geometry Cone {
        bottomRadius 0.3
        height 0.5
        subdivision 36
      }
    }
  ]
}
```

Output will look like this:



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 • **Design of Geometrical Figures**	
Teacher Action	**Student Action**
Now, it's your turn to make another shape. Let's start with a cylinder.	
Quick reference to draw any geometric shape using webots:	



https://s3-whjr-curriculum-uploads.whjr.online/ee78f62c-4410-447e-adc7-8cc6ed886bf9.png

Click on **+** to add new shape

1. Click on arrow of **Base nodes**
    a. Select **Transform**
    b. Write name of the Function DEF **CAPSULE**(Note: DEF is already written
    c. Click **ADD**. Function is created
    d. Now click on **Arrow** just before the function name
    e. Select **translation** and write value **2, 0, 0**

f. Now click on Arrow just before the function name
    i. Double Click on **children**
    ii. Click on Arrow before the **Base nodes**
        1. Select **SHAPE**
        2. Click on **ADD**
        3. Now click on **Shape**
            a. Double Click on **Appearance**
            b. **Click on Base Nodes**
                i. Select **PBR Appearance**
                ii. Click **ADD**
            c. **Click on Arrow PBR Appearance**
            d. Select **baseColor 0.6, 0.8, 0.24**
            e. Select **roughness 0.2**
            f. Select **metalness 0**
        4. Double Click on **Geometry**

NULL

a. Select **Capsule**

b. Click **Add**

c. **radius 0.25**

d. **height 0.3**

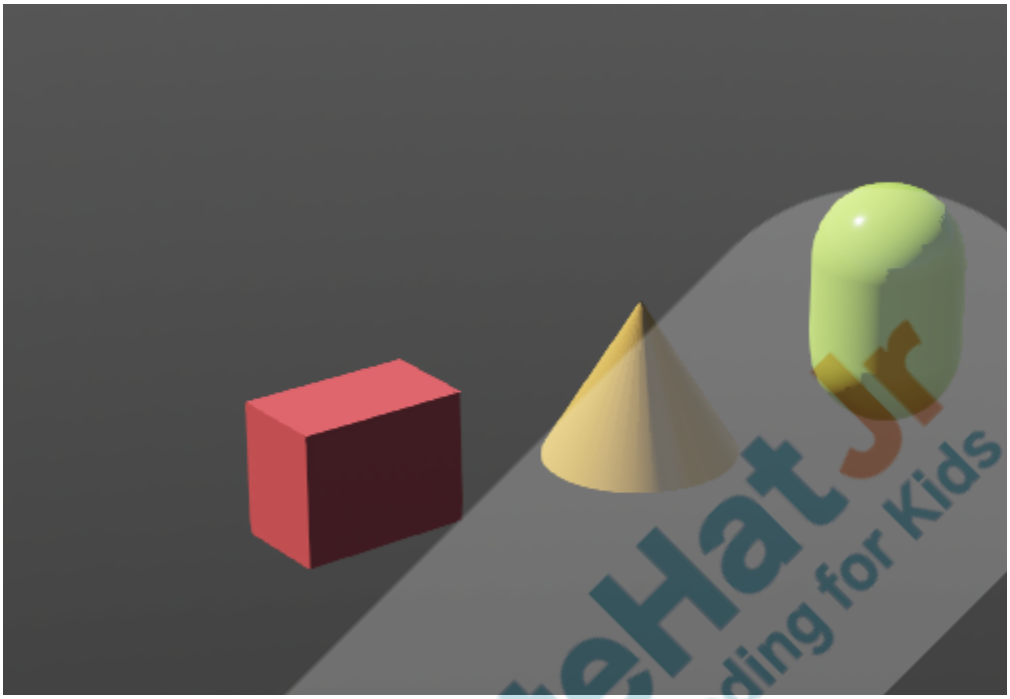
e. **subdivision 24**

Save the simulation. 

Code for Capsule Shape will look like below:

```
DEF CAPSULE Transform {
  translation 2 0 0
  children [
    Shape {
      appearance PBRAppearance {
        baseColor 0.621333 0.8 0.264
        roughness 0.2
        metalness 0
      }
      geometry Capsule {
        height 0.3
        radius 0.25
        subdivision 24
      }
    }
  ]
}
```

Output will look like this:



It's time to draw next shape:

Click on **+** to add new shape

2. Click on arrow of **Base nodes**
    a. Select **Transform**
    b. Write name of the Function DEF **CYLINDER**(Note: DEF is already written
    c. Click **ADD**. Function is created
    d. Now click on **Arrow** just before the function name
    e. Select **translation** and write value **3, 0, 0**

f. Click on Arrow just before the function name

i. Double Click on **children**

ii. Click on **Arrow** before the **Base nodes**

1. Select **SHAPE**
2. Click on **ADD**
3. Now click on **Shape**

a. Double Click on **Appearance**

b. **Click on Base Nodes**

i. Select **PBR Appearance**

ii. Click **ADD**

c. **Click on Arrow PBR Appearance**

i. Select **baseColor 0.2, 0.8, 0.24**

ii. Select **roughness 0.2**

iii. Select **metalness**

0

4. Double Click on **Geometry**

NULL

a. Select **Cylinder**

b. Click **Add**

c. height 0.6

d. radius 0.2

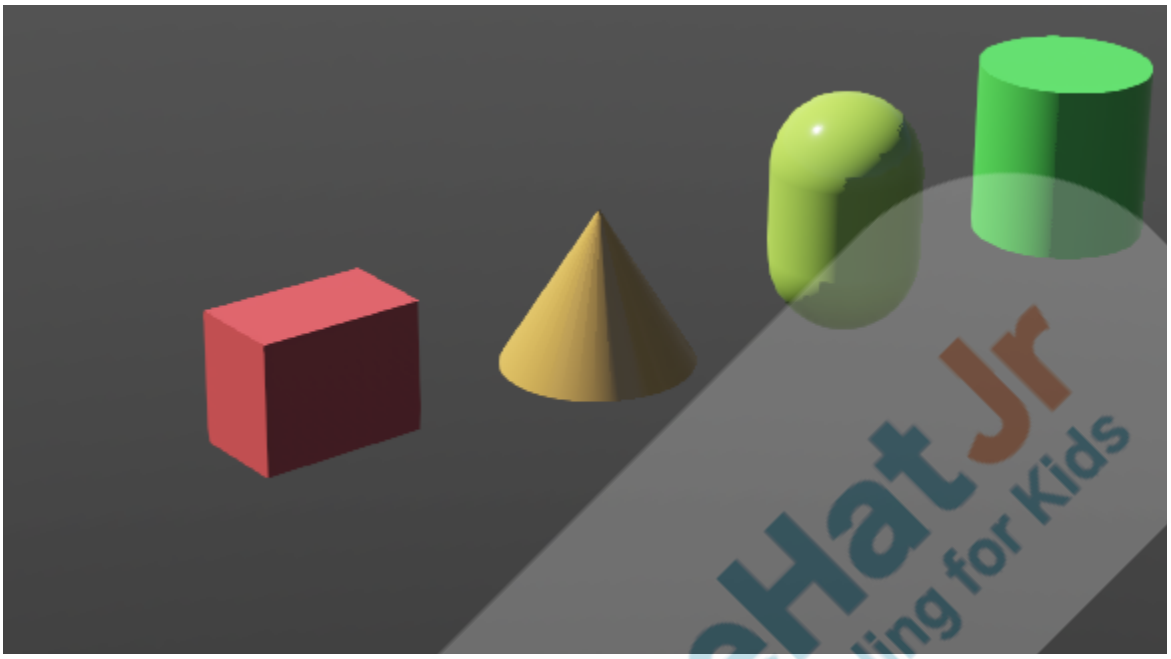
e. subdivision 24

Save the simulation. 

Reference Cylinder Shape Code:

```
DEF CYLINDER Transform {
  translation 3 0 0
  children [
    Shape {
      appearance PBRAppearance {
        baseColor 0.264 0.8 0.264
        roughness 0.2
        metalness 0
      }
      geometry Cylinder {
        height 0.6
        radius 0.3
      }
    }
  ]
}
```

Output will look like this:



So today we learned how to design simple elements using Webots. These designs will give a clear picture of how Scene tree works. Later on it will help to design our first Robot.	
Note: *If you get any errors in the design part.Follow the below procedure:* 1. *Save the webot file.Open the webot file with Notepad++ or any other word pad.* 2. *Open the reference file too with Notepad++.* 3. *Match the content if there's any design issue.* 4. *Basically wbt file is a program file written in C. Either we can do it using blocks or through a*	

program file.	
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! In the next class, we will learn to create a ramp follower robot.	*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 ✖ End Class

ADDITIONAL ACTIVITIES

(Optional)

Additional Activities:

Ask the student to make Sphere.

Reference Code for Sphere:

```
DEF SPHERE Transform {
  translation 8 0 0
  children [
    Shape {
      appearance PBRAppearance {
        baseColor 0.8 0.264 0.621333
        roughness 0.2
        metalness 0
      }
      geometry Sphere {
        radius 0.3
        subdivision 2
      }
    }
  ]
}
```

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Activity 1	Reference File Class Code	https://github.com/procodingclass/PRO-C279-Teacher-Activity
Teacher Activity 2	Reference Code	https://github.com/procodingclass/PRO-C279-Reference-Code
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/d7ff874d-c3a2-4ac2-a108-df0c87c19cdd.pdf
Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-C279-Project-Solution
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/b8806acd-fa79-452c-8642-9cb1dd5b0ba2.pdf
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C279-Teacher-Activity