Mini – Project Robotics

Roll no : 512

Name : Sejal Pawar

Class : M.S c CS P2

Subject : Robotics

Topic : Mini Project

Aim : simulate a Lego robot navigating around obstacles using touch sensors

Description :

Import Statements: The code starts by importing necessary classes from the ch.aplu.robotsim library.

Static Initialization Block : This block sets up the simulation environment with obstacles using the RobotContext class. It displays a navigation bar and adds four obstacles at specific positions on the simulation canvas.

1. obstaclee Class and Constructor:
    - The obstaclee class is defined.
    - The constructor method is defined within the class. This constructor will be executed when an instance of the obstaclee class is created.
2. Robot Setup in Constructor:
    - An instance of LegoRobot named robot is created to represent the simulated robot.
    - An instance of Gear named gear is created to control the robot's movement.
    - An instance of TouchSensor named ts is created and connected to sensor port S3 of the robot.
    - The gear and ts instances are added to the robot using the addPart method.
    - The movement speed of the robot is set to 30 using the setSpeed method of the gear instance.

- The robot starts moving forward using the forward method of the gear instance.

3. Main Loop:
    - The program enters an infinite loop using while (true).
    - Inside the loop, the program checks if the touch sensor (ts) is pressed using the isPressed method of the TouchSensor instance.
    - If the touch sensor is pressed (indicating an obstacle is encountered), the program executes the obstacle avoidance behavior:
        - The robot is commanded to move backward for 1200 milliseconds using the backward method of the Gear instance.
        - The robot is turned left for 750 milliseconds using the left method of the Gear instance.
        - The robot then continues moving forward using the forward method.
4. main Method:
    - The main method is defined, serving as the entry point of the program.
    - It creates an instance of the obstaclee class using new obstaclee(). This instantiation starts the robot simulation and its navigation behavior.

Conclusion :

The code sets up the robot simulation environment using the RobotContext class. It adds four obstacles at specific positions using the useObstacle method and specifies the obstacle images' filenames and positions.

1. The obstaclee class is defined. It contains the main logic for controlling the Lego robot.
2. Inside the obstaclee constructor:
    - An instance of LegoRobot is created to represent the robot.

- An instance of the Gear class is created to control the robot's movement.
- A TouchSensor is initialized and connected to sensor port S3 of the robot.
- The Gear instance is added to the robot, as well as the TouchSensor instance.
- The robot's movement speed is set to 30, and the robot starts moving forward.

3. A while loop runs indefinitely (while (true)):
    - The loop continuously checks if the touch sensor is pressed using the isPressed() method of the TouchSensor instance.
    - If the touch sensor is pressed (meaning the robot has encountered an obstacle):
        - The robot is commanded to move backward for a duration of 1200 milliseconds using the backward method of the Gear instance.
        - The robot is turned left for a duration of 750 milliseconds using the left method of the Gear instance.
        - The robot is then commanded to move forward again to continue its navigation.
4. The main method is defined:
    - It simply creates an instance of the obstaclee class, which starts the robot and its obstacle avoidance behavior.

This code essentially simulates a robot moving around the obstacles while avoiding them using touch sensors. If the touch sensor detects an obstacle, the robot reverses its direction, turns left, and then continues forward to navigate around the obstacleIn summary, this code creates a simulated Lego robot that navigates around obstacles using touch sensors. When an obstacle is detected, the robot backs up, turns left, and then resumes moving forward to avoid the obstacle. The code relies on the provided simulation library to visualize the robot's behavior.

Program Code :

```
import ch.aplu.robotsim.*;

public class obstaclee
{
  static
  {
    RobotContext.showNavigationBar();
    RobotContext.useObstacle("sprites/bar0.gif", 250, 200);
    RobotContext.useObstacle("sprites/bar1.gif", 400, 250);
    RobotContext.useObstacle("sprites/bar2.gif", 250, 400);
    RobotContext.useObstacle("sprites/bar3.gif", 100, 250);
  }

  public obstaclee()
  {
    LegoRobot robot = new LegoRobot();
    Gear gear = new Gear();
    TouchSensor ts = new TouchSensor(SensorPort.S3);
    robot.addPart(gear);
    robot.addPart(ts);
    gear.setSpeed(30);
    gear.forward();
    while (true)
    {
      if (ts.isPressed())
      {
        gear.backward(1200);
        gear.left(750);
        gear.forward();
      }
    }
  }

  public static void main(String[] args)
  {
    new obstaclee();
  }
}
```

Displaying obstaclee.java.

Output :